Linux-Firewalls mit iptables & Co.

Ralf Spenneberg

Linux-Firewalls mit iptables & Co.

ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

10 9 8 7 6 5 4 3 2 1

12 11 10

ISBN 3-8273-xxxx-z

ein Imprint der Pearson Education Deutschland GmbH,
Martin-Kollar-Straße 10–12, D-81829 München/Germany

Einbandgestaltung: Marco Lindenbeck, webwo GmbH (mlindenbeck@webwo.de)
Lektorat: Boris Karnikowski, bkarnikowski@pearson.de
Fachlektorat: Wilhelm Dolle, Berlin
Korrektorat: Friederike Daenecke, Zülpich
Herstellung: Monika Weiher, mweiher@pearson.de
Satz: le-tex publishing services GmbH, Leipzig
Druck und Verarbeitung: Kösel, Krugzell (www.KoeselBuch.de)
Printed in Germany

Für Claudia: Ich Dich auch ;-)

Inhaltsübersicht

Inhaltsverzeichnis

Vorwort zur 2. Auflage

Dieses Buch widmet sich dem Paketfilter Netfilter/Iptables der Linux-Kernel 2.4 und 2.6. Obwohl bereits einiges an Literatur zu diesem Thema existiert, hat mir persönlich kein Buch richtig gut gefallen. Daher habe ich bereits seit einigen Jahren überlegt, ein eigenes Buch zu diesem Thema zu schreiben, das auch die fortgeschrittenen Themen und neue Eigenschaften behandelt, die in den letzten Jahren hinzugekommen sind.

Diese jetzt vorliegende zweite Auflage wurde stark überarbeitet. Das Netfilter-Team (das das Framework im Kernel entwickelt, das mit Iptables gesteuert wird) hat seit der ersten Auflage große Teile des Codes[a], die im Kernel verborgen sind, ausgetauscht. Diese Änderungen bemerkt der Administrator nicht bei der Nutzung des `iptables`-Befehls. Jedoch benötigt er die Informationen für das Verständnis. Ich habe daher das Buch komplett auf den aktuellen Stand gebracht. An den Stellen, an denen ältere Linux-Kernel (vor 2.6.20) gewisse Einstellungen anders vorgenommen haben, habe ich versucht, dies entsprechend im Text zu erwähnen. Im Wesentlichen basiert dieses Buch jedoch auf den Linux-Kernel-Versionen ab 2.6.20.

Neben diesen Änderungen hat IPv6 endlich seinen prominenten Platz in diesem Buch gefunden. Dieses Protokoll wird in den nächsten Monaten und Jahren die tägliche Arbeit bereichern. Die Zeiten, in denen Administratoren dieses Protokoll einfach ignorieren konnten, sind nun endlich vorbei.

Ansonsten habe ich versucht, sowohl den Einsteiger zu berücksichtigen und ihn in einzelnen Kapiteln behutsam mit der Materie des Paketfilters vertraut zu machen als auch dem fortgeschrittenen Benutzer die Mittel an die Hand zu geben, die er für eine mächtige Firewall benötigt.

Ich hoffe, dass es mir mit diesem Buch gelungen ist, Ihren Geschmack zu treffen und Ihnen wertvolle Informationen zu liefern. Nach meinen Büchern „VPN mit Linux“ [4], „Intrusion Detection und Prevention mit Snort & Co.“ [3] und „SELinux und AppArmor“ [5] rundet dieses Buch das Thema Netzwerksicherheit mit Open Source ab.

Sollten Sie Fragen oder Kritik zu diesem Buch haben, zusätzliche Ideen haben oder Informationen vermissen, würde ich mich über Ihre E-Mail an *ralf@spenneberg.net* freuen.

Ich wünsche Ihnen viel Spaß beim Lesen und viel Erfolg bei der Konfiguration und Administration Ihrer Linux-Firewall.[b]

Einleitung

Angesichts täglicher Meldungen über neue Viren, Hackerangriffe und Sicherheitslücken ist eine Firewall ein wesentlicher Bestandteil eines sicheren Netzwerks. Eine Firewall ist besonders wichtig, wenn Sie Ihr Netzwerk mit anderen unbekannten und nicht vertrauenswürdigen Netzwerken wie dem Internet verbinden. Dann sollten Sie diese Verbindung mit einer Firewall überwachen. Die Firewall beschränkt den Netzwerkverkehr auf den von Ihnen gewünschten Verkehr und weist alle anderen Anfragen ab. Natürlich müssen Sie sicherstellen, dass es keine Möglichkeit gibt, die Firewall mithilfe eines Modems oder anderer Hilfsmittel zu umgehen, denn dann ist die Firewall fast nichts mehr wert. Es bleibt Ihnen dann nur noch der Wert der Hardware.

Aber selbst wenn Sie Ihr Netzwerk nicht mit einem anderen Netz verbinden oder hierzu eine kommerzielle Firewall verwenden, kann es sinnvoll sein, über interne Linux-Firewalls nachzudenken. Kennen Sie Ihre Benutzer und Anwender wirklich? Vertrauen Sie ihnen? Gibt es in Ihrem Netz Systeme, die Sie schützen möchten oder müssen? Dann ist es sinnvoll, auch innerhalb eines Netzes mit Firewalls den Schutz dieser Systeme zu implementieren.

Dieses Buch hilft Ihnen dabei, diese Funktionen mithilfe von Linux und Iptables zu realisieren.

Das Buch beginnt mit den Firewall-Grundlagen. Hier lernen Sie wichtige Begriffe, Architekturen und die Unterschiede der verschiedenen Firewall-Technologien kennen. Außerdem erhalten Sie einen kurzen Überblick, wie Firewalls entstanden sind.

Der zweite Teil beschäftigt sich mit einer Einführung in Iptables und zeigt Ihnen, wie Sie eine einfache Firewall auf einem Gateway mit Iptables realisieren. Außerdem erfahren Sie in diesem Kapitel, wie Sie eine Linux-Distribution härten und wie Ihnen Intrusion-Detection-Systeme helfen können, Angriffe zu erkennen, die Ihre Firewall nicht abgewehrt hat.

Der dritte Teil beschäftigt sich mit typischen Firewall-Architekturen und stellt Ihnen eine lokale Firewall und eine Firewall mit DMZ inklusive der Realisierung und möglicher Probleme vor.

Im vierten Teil werden verschiedene zusätzliche Werkzeuge vorgestellt, die Sie benötigen oder verwenden können, um erfolgreich eine Firewall aufzubauen und zu warten. Dies sind Protokollserver, Zeitserver, Protokollanalysewerkzeuge, Oberflächen für die Administration, eine Betrachtung der von den Distributionen mitgelieferten Werkzeuge und Werkzeuge für den Test Ihrer Firewall.

Der fünfte Teil ist für die fortgeschrittenen Anwender gedacht. Er beginnt mit einer kompletten Referenz aller Iptables-Funktionen (Tests und Ziele), den in Patch-O-Matic verfügbaren zusätzlichen Funktionen, einer Beschreibung des Connection-Tracking-Mechanismus und dessen Tuning und der Beschreibung der NAT-, Mangle- und Raw-Tabellen. Weiterhin finden Sie hier die Erklärung der Variablen im `/proc`-Verzeichnis. Sie lernen, wie Sie in Datenbanken protokollieren, IP-Adressen mit `ipset` gruppieren, hochverfügbare Firewalls aufbauen und welche Funktionen in der nächsten Zukunft in Iptables zu erwarten sind.

Der sechste Teil ist komplett transparenten Firewalls gewidmet. Dies sind Firewalls, die auf der IP-Ebene unsichtbar sind und wie auf einer Bridge arbeiten.

Der siebte Teil beschäftigt sich mit den Besonderheiten einzelner Protokolle und zeigt Ihnen, wie Sie DHCP, IPv6 oder IPsec filtern.CE[c]

Diese Kapitel sollten alle Fragen rund um eine Iptables-Firewall beantworten.

CE[a] Bitte Ergänzung prüfen.

TS[b] Bitte Ort und Name liefern.

CE[c] Bitte prüfen: Teil 8 behandelt IPv6. Bitte den Text anpassen.

Teil I

Firewall-Grundlagen

1. Firewall-Geschichte

Ich persönlich empfinde es immer als hilfreich, die geschichtliche Entwicklung der eingesetzten Produkte und Technologien zu kennen, um diese besser bewerten und einschätzen zu können. Dieses Kapitel versucht, skizzenhaft die Entwicklung der Firewalls aufzuzeigen.

1.1 Der Anfang des Internets

Das Internet begann als eine Reihe von Notizen von J.C.R. Licklider am MIT, in denen er 1962 ein „Galactic Network" beschreibt. Ein Jahr früher veröffentlichte Leonard Kleinrock (ebenfalls am MIT) eine erste Arbeit über paketvermittelnde Netzwerke. Aus diesen Ideen entstand 1965 das erste Wide-Area-Network (WAN), in dem Lawrence G. Roberts gemeinsam mit Thomas Merrill den TX-2-Computer des Lincoln Labs in Massachusetts mit dem Q-32-Computer des SDC in Kalifornien verband. 1966 ging Roberts dann zur Defense Advanced Research Projects Agency (DARPA), um dort ein Konzept für ein Computer-Netzwerk zu entwickeln. Angeblich versprach Charlie Hertzfeld (Direktor der DARPA) dem IPTO-Direktor Bob Taylor eine Million Dollar für die erfolgreiche Implementierung eines verteilten Kommunikationsnetzwerks. Die Entwicklung des ARPANET begann. Im Dezember 1970 wurde das erste Netzwerkprotokoll für das ARPANET offiziell verabschiedet: das Network Control Protocol (NCP). Die Entwicklung der Anwendungen konnte beginnen. 1972 wurde als erste Applikation E-Mail vorgestellt.

Das NCP-Protokoll erlaubte zwar die Kommunikation in dem Netz. Je größer das ARPANET wurde, desto deutlicher wurden jedoch seine Schwächen. So wurde mit der Entwicklung der Internetprotokollfamilie TCP/IP begonnen. TCP/IP wurde bereits 1980 vom Department of Defense als Standard anerkannt und 1982 in einem RFC beschrieben. Am 1.1.1983 wurde das ARPANET auf TCP/IP umgestellt. In den folgenden Jahren wuchs das ARPANET. Weitere Netzwerke wurden aufgebaut: BITNET, USENET, MFENET, HEPNET, SPAN, CSNET. Im kommerziellen Bereich entstanden XNS (Xerox), IBM SNA und das DECNET. Speziell für den akademischen Bereich wurde das NSFNET gestartet. Dieses wuchs in nur 8 Jahren von einem Backbone mit 6 Knoten (56 kBit/s) auf 21 Knoten mit bis zu 45 Mbit/s an. Das Internet schloss in dieser Zeit 50.000 Netze weltweit zusammen. Dennoch handelte es sich bei dem Internet lange Zeit um eine heile Welt. Jeder kannte und vertraute jedem anderen Teilnehmer. Als Sicherheitsproblem betrachtete man lediglich die Gefahr eines Paketverlustes, und man entwickelte Methoden und Protokolle, um den Verlust von Paketen zu verhindern und die Verfügbarkeit der Dienste und der Kommunikation zu garantieren.

1.2 Der Morris-Wurm

Im November 1988 löste der Morris-Wurm von Robert Tappan Morris eine Epidemie im Internet aus. Dieser Wurm drang über Sicherheitslücken in UNIX-Betriebssysteme TSd ein und nutzte Sendmail, Finger und die R-Dienste für seine Verbreitung aus. Die verwendeten Sicherheitslücken werden in der Bugtraq-Datenbank unter den Nummern 1 und 2 geführt (`http://www.securityfocus.com/bid/1`). Während seiner Verbreitung infizierte dieser Wurm mit 6000 Rechnern etwa 10% des damaligen Internets. Robert Morris wurde am 26. Juli 1989 als Freisetzer des Wurms identifiziert und am 22. Januar 1990 zu 400 Stunden gemeinnütziger Arbeit und 10.400 Dollar Geldstrafe verurteilt. Interessanterweise war sein Vater, Robert Morris Senior, zu diesem Zeitpunkt Chef-Wissenschaftler der National Security Agency (NSA).

Als direkte Reaktion auf den Wurm wurde noch im November 1988 das Computer Emergency Report Team Coordination Center (CERT/CC) von der Defense Advanced Research Projects Agency (DARPA) gegründet. Des Weiteren begannen die Unternehmen, über einen Schutz ihrer Netzwerke vor dem Internet nachzudenken. Bis zu diesem Zeitpunkt gab es keine Firewalls, kein NAT und auch keinen anderweitigen Schutz.

1.3 Die erste Firewall

Recht früh wurden bereits Router als Sicherheitssysteme für die Abschottung eines Netzes von den anderen Netzen eingesetzt. Zu Beginn nutzte man diese Router, um zu verhindern, dass Netzwerk-Fehlkonfigurationen Broadcast-Stürme auslösten, die sämtliche Netze in Mitleidenschaft zogen.[1]

Die ersten richtigen Firewalls wurden relativ gleichzeitig bei DEC und AT&T entwickelt. Bei DEC betrieb zunächst Brian Reid und dann Paul Vixie ein System mit dem Namen *gatekeeper.dec.com*. Für den Zugriff auf das Internet mussten die Anwender sich auf diesem System anmelden (per telnet und ftp) und konnten dann von diesem System aus auf Systeme im Internet zugreifen.

Gleichzeitig arbeitete Marcus J. Ranum für Frederick Avolio in einer anderen Abteilung bei DEC. Sie verfügten über eine Internetverbindung mit 9600 Baud und benötigten eine ähnliche Sicherheitsstruktur. Marcus Ranum wollte jedoch den Anwendern keine Konten auf dem System für eine Anmeldung zur Verfügung stellen und schrieb daher den ersten FTP-Proxy für seinen Gatekeeper. Kurze Zeit später (1991) wurde diese Firewall bereits als erste kommerzielle Firewall DEC-SEAL (Securing External Access Link) an einen großen Chemie-Konzern verkauft und dort im Juni 1991 auf mehreren Systemen eingerichtet. Ein Firewall-System bestand aus einem Gatekeeper, der als einziges System auf das Internet zugreifen durfte, und einem Gate, das den Zugriff des internen Netzes auf den Gatekeeper als Paketfilter über-

1 Ein Broadcast-Paket wird von jedem Rechner entgegengenommen. Viele der ersten Protokolle verwendeten Broadcast-Pakete für die Kommunikation. Ein Router leitet Broadcast-Pakete nicht weiter. So sind die Rechner hinter dem Router nicht von einem Broadcast-Paket betroffen. Ansonsten kann in einem großen Netzwerk der Broadcast-Verkehr einen großen Teil der Netzwerkkapazität belegen.

TSd Bitte diese Änderungen bestätigen.

Editor's or typesetter's annotations (will be removed before the final TEX run)

wachte. Der Gatekeeper verfügte über eine gewisse Anzahl von Proxies, die den Zugriff per telnet, ftp, E-Mail, News, Whois und X erlaubten. Die DEC SEAL wurde schließlich zur AltaVista Firewall weiterentwickelt.

INFO

In Kapitel 2 erläutere ich die Begriffe Paketfilter und Proxy genauer.

Frederick Avolio und Marcus Ranum wechselten zu Trusted Information Systems (TIS) und programmierten dort das TIS Firewall-Toolkit (TIS FWTK, `http://www.fwtk.org/`). Nach dem Kauf von TIS durch NAI 1998 wurde diese Firewall als Gauntlet kommerziell vermarktet. Gleichzeitig wurde bei AT&T von William R. Cheswick und Steven M. Bellovin in den Bell Laboratories eine Firewall entwickelt, die nicht über einzelne Proxies für jedes Protokoll verfügte, sondern als Curcuit-Relay-Proxy ausgelegt war. Dies war der Vorgänger des SOCKS-Protokolls und des SOCKS-Proxy. Kommerziell wurde diese Firewall als Raptor Eagle kurze Zeit später verkauft.

Die ersten Firewalls waren also immer Systeme, die für die Funktion auf Proxies zurückgriffen. Auch heute gibt es noch viele Firewall-Systeme, die für die Realisierung Proxies einsetzen. Bei einigen Protokollen ist dies auch sinnvoll, um eine Filterung von Viren oder unerwünschten Inhalten durchzuführen.

Im Jahr 1994 betrat Check Point den Markt der Firewalls und bot von Anfang an einen Paketfilter mit einer grafischen, leicht zu administrierenden Oberfläche an. Gleichzeitig war dieser Paketfilter in der Lage, den Zustand der TCP-Verbindungen zu überwachen und Verbindungen nur in bestimmten Richtungen zu erlauben.

1.4 Firewalls heute

Heute bestehen Firewall-Systeme meist aus einer Kombination aus Paketfilter und Proxy. Während einige Protokolle aufgrund ihrer Natur nicht mit einem Proxy gefiltert werden können (z.B. IPsec), ist es bei anderen Protokollen zum Schutz der eigenen Infrastruktur zwingend erforderlich, Proxies einzusetzen. Ein Paketfilter kann keinen Virus oder andere bösartige Inhalte in E-Mails oder in Webseiten erkennen und entfernen.

Dennoch gibt es auch häufig Szenarien, bei denen dieses gar nicht gewünscht wird oder eine Proxy-Firewall nicht den Datendurchsatz bewältigen könnte. Hier werden dann reine Paketfilter eingesetzt.

2. Firewall-Technologie

Es gibt viele verschiedene Technologien, die bei dem Aufbau einer Firewall eingesetzt werden können. Dieses Kapitel stellt die wichtigsten vor und beschreibt ihre Funktionsweise sowie ihre Vor- und Nachteile.

2.1 Paketfilter

Der Paketfilter ist die einfachste Technologie für die Implementierung einer Firewall. Die erste Implementierung geht auf Jeffrey Mogul im Jahr 1989 zurück (Simple and flexible datagram access controls for Unix-based gateways[1]). Mogul stellte auf der USENIX Summer Conference im März 1989 mit dem Programm Screend einen ersten Paketfilter vor. Der Screend ist ein Userspace-Programm, das die Entscheidung trifft, ob ein Paket passieren darf oder nicht.

Im Grunde ist ein Paketfilter ein intelligenter Router. Während ein normaler Router lediglich entscheidet, wohin ein Paket geschickt wird, analysiert ein Paketfilter zusätzlich das Paket und bestimmt, ob ein Paket überhaupt weitergesendet werden darf oder ob es verworfen werden muss.

Da diese beiden Aufgaben eng miteinander verwandt sind, werden seit damals fast alle Router mit zusätzlichen Paketfilterfunktionalitäten ausgestattet. Auch die meisten Netzwerkbetriebssysteme verfügen seit Jahren über derartige Paketfilter.

Die meisten Paketfilter sind IP-Paketfilter. Das bedeutet, dass sie ihre Betrachtung auf IP-Pakete beschränken. Dort analysieren sie den IP-Header. Die meisten Paketfilter können zusätzlich im Falle eines UDP-, TCP- oder ICMP-Paketes auch deren Header analysieren. Hierzu nutzen die Paketfilter üblicherweise ein Regelwerk, das sie Regel für Regel bei jedem Paket testen. Trifft eine Regel zu, so wird die mit der Regel verbundene Aktion ausgeführt. Die üblicherweise unterstützten Aktionen sind das Annehmen und Routen des Pakets, das Verwerfen des Pakets und das Ablehnen des Pakets (Senden einer Fehlermeldung an den Absender).

Ein klassischer zustandsloser Paketfilter kann nicht erkennen, ob ein Paket zu einer aufgebauten Verbindung gehört oder nicht. Jedes Paket wird einzeln für sich betrachtet und analysiert. Der Zusammenhang, in dem das Paket auftritt, ist belanglos. Der Administrator eines Paketfilters muss daher sowohl die Pakete des Clients als auch die Pakete des potenziellen Servers in seinem Regelwerk berücksichtigen.

1 http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-89-4.pdf

2.2 Zustandsorientierte Paketfilter

Die ersten zustandsorientierten kommerziellen Paketfilter wurden 1993 verfügbar. Das bekannteste, heute noch im Einsatz befindliche Produkt ist die Check Point Firewall-1. Dieses Produkt besaß eine einfache Verbindungstabelle, in der sich die Firewall sämtliche Verbindungen merkte, die von innen aufgebaut worden waren, und automatisch sämtliche Antworten zuließ.

Die erste frei verfügbare Variante eines zustandsorientierten Paketfilters war IP-Filter (IPF) 3.0 (1996) von Darren Reed. Ebenfalls 1996 erschien mit der SF Firewall (`http://www.ifi.unizh.ch/ikm/SINUS/sf-doc/impl.htm`) eine erste Implementierung für Linux. Diese wurde in den weiteren Jahren zur Sinus Firewall mit grafischer Oberfläche weiterentwickelt. Leider wurden die entsprechenden Seiten im Internet inzwischen gelöscht.

Heute besitzt Linux mit Iptables/Netfilter einen der mächtigsten zustandsorientierten Paketfilter, die verfügbar sind. Allen zustandsorientierten Paketfiltern ist gemeinsam, dass sie sich in einer Tabelle die aufgebauten und von dem Regelsatz erlaubten Verbindungen merken und alle weiteren Pakete dieser Verbindungen mehr oder weniger automatisch erkennen und zulassen. Die explizite Erkennung dieser Pakete durch einzelne Regeln ist nicht erforderlich. Dies vereinfacht zum einen die Definition und die Wartung der Regeln erheblich, denn die Zahl der Regeln schrumpft mindestens um die Hälfte. Zum anderen erhöht es auch sehr stark die Sicherheit der Firewall. Es ist nun nicht erforderlich, grundsätzlich jedes mögliche Antwortpaket eines jeden möglichen Kommunikationspartners durchzulassen. Die zustandsorientierte Firewall beschränkt dies auf die Pakete, die sie als Teil der Verbindung tatsächlich erkennt. Eine Weiterentwicklung ist die Inspection- oder auch Deep-Inspection-Firewall. Hierbei handelt es sich um Systeme, die die Pakete noch genauer analysieren und – besonders als Deep-Inspection-Systeme – eine Kombination aus Paketfilter und Intrusion-Detection-System oder Virenscanner darstellen. Hierzu prüfen die Systeme, ob über die Verbindung ein Angriff oder ein Virus übertragen wird. Aufgrund der Natur eines Paketfilters und der hohen Anforderung an die Geschwindigkeit sind die Deep-Inspection-Systeme jedoch meist nicht so gut wie eine Kombination aus einem reinen Paketfilter und einem Virusscanner oder einem IDS. Dennoch haben sich die Paketfilter in vielen Bereichen gegenüber den Proxies durchgesetzt, da die Implementierung preiswert und einfach möglich ist, der Paketfilter auch im Bereich der Gigabit-Netzwerke eine leistungsgerechte Filterung durchführen kann und als zustandsorientierter Paketfilter mit Inspection oder sogar Deep-Inspection-Analyse für viele Anwendungsfälle ausreichend Sicherheit bietet.

2.3 Curcuit Relay

Das Curcuit Relay ist ein Proxy auf der Transport-Schicht, der das verwendete Applikationsprotokoll nicht versteht, sondern die Daten auf der Transport-Schicht austauscht. Dieses Prinzip wurde 1996 in dem SOCKS-Protokoll weiterentwickelt und verbessert. Das SOCKSv5-Protokoll wurde in dem RFC 1928 definiert und festgelegt. Heute sind noch zwei Versionen des Protokolls im Einsatz: SOCKSv4 und SOCKSv5. SOCKSv5 unterstützt auch Applikationen, die das UDP-Protokoll nutzen.

Der Punkt, der für den Einsatz des SOCKS-Proxys spricht, ist die einfache Implementierung. Wenn der Quelltext einer Applikation verfügbar ist, kann diese sehr einfach mit der Fähigkeit ausgestattet werden, den SOCKS-Proxy zu nutzen. Der einfache Socket-Aufruf muss lediglich durch den SOCKS-Socket-Aufruf ersetzt werden. Viele kommerzielle Programme, wie der Microsoft Internet Explorer, enthalten bereits die hierfür notwendige Logik.

2.4 Application-Level-Gateway (Proxy)

Ein Application-Level-Gateway ist die klassische Firewall-Technologie, wie sie von Marcus Ranum (siehe Abschnitt 1.3) in der ersten Firewall implementiert wurde. Hierbei handelt es sich um Proxies, die auf der Schicht des Applikationsprotokolls arbeiten. Daher benötigt ein derartiges Gateway für jedes Applikationsprotokoll einen eigenen Proxy. Diese Application-Level-Gateways können daher meist nur eine beschränkte Anzahl von Protokollen filtern. Obwohl Sie dies als einen Nachteil dieser Technologie ansehen können, bietet diese Technologie auch Vorteile. Die Application-Level-Gateways implementieren das Applikationsprotokoll und können daher auch die korrekte Verwendung prüfen. Speziell bei komplizierten und gefährlichen Protokollen wie FTP oder HTTP können diese Proxies durch selektive Unterstützung des Protokolls die gefährlichen Befehle herausfiltern und deren Verwendung unterbinden. Ein reiner Paketfilter kann dies nicht. Hier sind zusätzliche Funktionen in Form eines Inspection- oder Deep-Inspection-Paketfilters erforderlich. Während jedoch das Application-Level-Gateway nur die erlaubten Funktionen implementiert (Positiv-Liste) und alle weiteren Verwendungen des Protokolls automatisch unterbindet, muss ein Deep-Inspection-Paketfilter die bösartige Verwendung des Protokolls erkennen und verhindern (Negativ-Liste). Der Deep-Inspection-Paketfilter kann daher nur bekannte Angriffe erkennen und verhindern, während ein Application-Level-Gateway weiterreichende Sicherheit bietet.

Dennoch haben sich in den letzten Jahren die Paketfilter in vielen Bereichen durchgesetzt. Dies hängt mit den vielen verschiedenen, teilweise sehr komplizierten modernen Protokollen zusammen. Für jedes dieser Protokolle ist ein eigener Proxy erforderlich. Deren Implementierung ist häufig so aufwendig und kompliziert, dass sie entweder gar nicht vorgenommen wird oder der Proxy anschließend selbst zu einem Sicherheitsrisiko wird, da die Implementierung Fehler aufweist.

2.5 Network-Address-Translation

Die Network-Address-Translation (NAT) wurde zunächst als Antwort auf die beschränkte Anzahl von IP-Adressen im Internet entwickelt. Mithilfe von NAT können viele Rechner über dieselbe IP-Adresse auf Dienste im Internet zugreifen. Hierfür wird NAT üblicherweise auf einem Router oder der Firewall implementiert. Die IP-Adresse aller in das Internet geschickten Pakete wird dann von dem Router oder der Firewall durch die IP-Adresse des Routers bzw. der Firewall selbst ersetzt. Dabei verwendet der Router bzw. die Firewall für jede Verbindung einen eindeutigen ungenutzten lokalen Port. Die Information über die Verbindung und die verwendeten IP-Adressen und Ports speichert das Gerät in einer Tabelle ab. Sobald ein Antwortpaket das Gerät erreicht, liest das Gerät das echte Ziel aus der Tabelle ab und lei-

TSe Bitte bestätigen Sie diese Formatierung.

Editor's or typesetter's annotations (will be removed before the final TEX run)

tet das Paket an den entsprechenden Rechner weiter. Nach Beendigung der Verbindung oder nach Ablauf einer vorgegebenen Zeit wird die Verbindung aus der Tabelle entfernt. Dies ist ein Source-NAT, bei dem die Quelle der Verbindung *genatteti*, also die Quelladresse ausgetauscht wird.

Das Source-NAT wird häufig als Sicherheitsfunktion eingesetzt, da es auch wirksam den Zugriff auf Systeme hinter dem Router bzw. der Firewall unterbindet. Da deren IP-Adresse unbekannt ist, ist kein Zugriff möglich, außer es existiert ein entsprechender Eintrag in der NAT-Tabelle, der Pakete, die von außen auf einer bestimmten IP/Port-Kombination ankommen, nach innen zu einem System weiterleitet. Dies ist das Destination-NAT. Dabei wird die Zieladresse einer Verbindung ausgetauscht.

Grundsätzlich ist ein Destination-NAT unter Sicherheitsaspekten kritischer zu betrachten, da es durch die Firewall den Zugriff auf interne Systeme erlaubt. Bei dem Einsatz des Destination-NAT sollte daher das interne System, auf das der Zugriff freigegeben wird, zusätzlich isoliert sein. Dies erfolgt in den meisten Umgebungen durch die Platzierung dieser Systeme in einer DMZ (siehe Abschnitt 3.2).

3. Firewall-Architekturen

Eine Firewall ist nicht ein einzelnes Gerät oder eine Gruppe von Geräten, sondern ein Konzept. Für die Implementierung eines Firewall-Konzepts haben sich in den vergangenen Jahren verschiedene Architekturen durchgesetzt. Diese Architekturen möchte ich Ihnen in diesem Kapitel vorstellen. Jede Architektur hat ihre Daseinsberechtigung. Es gibt keine absolut richtige Architektur. Wie bei allen anderen Betrachtungen sollten Sie immer Aufwand und Nutzen abwägen. Speziell bei der Wahl der Architektur müssen Sie den Aufwand in Form der benötigten Hardware analysieren.

Verabschieden Sie sich von dem Gedanken, eine absolut sichere Firewall aufzubauen, wenn Sie nicht das Netzwerkkabel durchschneiden wollen. Eine 100%ig sichere Firewall gibt es nicht. Daher muss Ihre Architektur mögliche Fehler in Ihrer Konfiguration und der Implementierung durch den Hersteller möglichst lange überleben, um Ihnen die Gelegenheit für eine Reaktion auf einen erfolgreichen Angriff auf Ihre Firewall einzuräumen. Ihre Firewall-Architektur sollte keinen Single-Point-of-Failure besitzen. Das ist ein Punkt, bei dessen Ausfall die Sicherheit komplett kompromittiert ist.

3.1 Screening-Router

Die erste Implementierung des Screening-Routers war der Screend von Jeffrey Mogul. Ein Screening-Router ist im Grunde nichts anderes als ein einfacher Paketfilter, wie er in diesem Buch beschrieben wird. Als Architektur basiert die Firewall lediglich auf diesem Screening-Router und verwendet keine andere zusätzliche Komponente. Abbildung 3.1 zeigt die Architektur.

Der Screening-Router erlaubt üblicherweise nur Verbindungen von innen nach außen. Eine Verbindungsaufnahme von außen nach innen wird von dem Screening-Router unterbunden.

In vielen Umgebungen, bei denen nur der Zugriff von innen nach außen erlaubt werden soll, ist ein Screening-Router ausreichend, um die Sicherheit eines Netzes zu gewährleisten. Dies hängt sicherlich auch stark davon ab, ob auch interne Dienste angeboten werden sollen. Dann ist in vielen Fällen ein Screening-Router nicht ausreichend, und Sie sollten sich Gedanken über die Verwendung einer demilitarisierten Zone machen (siehe Abschnitt 3.2).

Ein nicht zu unterschätzender Nachteil eines einzelnen Screening-Routers ist, dass bei einem Fehler in der Implementierung oder einem erfolgreichen Angriff das Netz offen steht. Wenn Ihre Firewall-Architektur nur aus einer Komponente besteht, kann ein Angreifer durch die

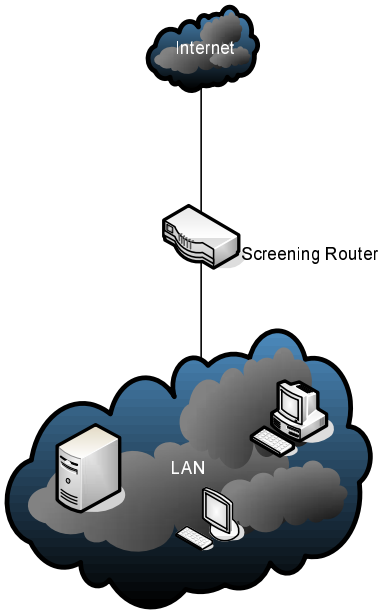


Abbildung 3.1: **Ein Screening-Router schützt das interne Netz.**

Überwindung dieser Komponente kompletten Zugriff auf Ihr Netzwerk erhalten. Um dieses zu verhindern, sollten Sie eine mehrstufige Firewall-Architektur wie die Multiple DMZ wählen (siehe Abschnitt 3.3).

3.2 DMZ

Sobald Sie Dienste anbieten möchten, die von anderen Anwendern im Internet oder von einem nicht vertrauenswürdigen Netz genutzt werden sollen, sollten Sie sich für die Implementierung Ihres Firewall-Konzepts mit der DMZ-Architektur beschäftigen. Hierbei handelt es sich um eine demilitarisierte Zone (häufig auch entmilitarisierte Zone genannt). Dies ist ein besonderes ausgelagertes Netz, in dem Sie die Systeme positionieren, auf die ein Zugriff aus dem Internet erforderlich ist. Auch ein Proxy wird üblicherweise in dieser DMZ aufgestellt. Ein zusätzlicher Screening-Router steuert den Paketfluss, sodass aus dem internen Netz nur ein Zugriff auf die Systeme der DMZ möglich ist (siehe Abbildung 3.2).

Ein direkter Zugriff auf das Internet wird so unterbunden.[1] Die Proxies greifen dann auf die Dienste im Internet zu und dienen dem internen Netz bei dem Zugriff auf das Internet als Mittelsmann. Alle Dienste, die Sie dem Internet anbieten möchten, wie zum Beispiel ein Web- und ein E-Mailserver, werden ebenfalls von Systemen in der DMZ implementiert. Da Sie diese

1 Da einige Protokolle nicht von Proxies unterstützt werden, ist für diese Protokolle häufig noch ein direkter Zugriff auf das Internet notwendig, wenn diese Protokolle genutzt werden sollen.

CE: Bitte bestätigen Sie diese Änderungen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

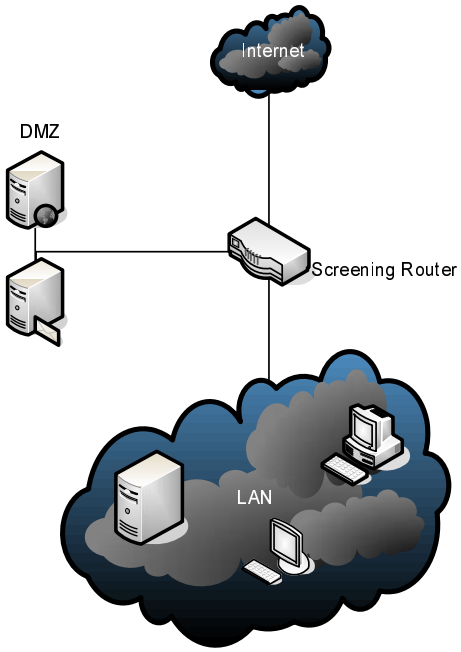


Abbildung 3.2: **Die DMZ wird am einfachsten durch ein drittes Bein am Screening-Router realisiert. Der Screening-Router regelt den Paketfluss.**

Dienste anbieten möchten, können Sie diese Systeme nicht so schützen wie die internen Systeme, auf die kein Zugriff erlaubt wird. Bei einem erfolgreichen Angriff und Einbruch in diese weniger geschützten Systeme sind die internen Systeme aber immer noch durch den Screening-Router geschützt, der hoffentlich nicht jeden Zugriff aus der DMZ in das interne Netz erlaubt.

Diese Kombination aus Proxies und einem Paketfilter[CE9] erhöht die Sicherheit der geschützten Systeme. Selbst wenn Sie keine Proxies einsetzen, erhöht die Auslagerung der angebotenen Dienste aus dem internen Netz die Sicherheit desselben, da nun bei einem Angriff kein direkter Zugriff auf die weiteren Rechner des internen Netzes möglich ist.

Dennoch basiert die Sicherheit der Architektur auf einem Single-Point-of-Failure, dem Screening-Router. Wenn der Administrator oder der Programmierer bei der Einrichtung oder Entwicklung des Screening-Routers einen Fehler macht, der erfolgreich von einem Angreifer ausgenutzt werden kann, ist der Angreifer in der Lage, auf das interne Netz zuzugreifen, da nur der Screening-Router dieses Netz schützt. Wenn dies nicht akzeptabel ist, können Sie die DMZ durch zwei Paketfilter realisieren: einen externen und einen internen Paketfilter. Der interne Paketfilter regelt den Verkehr zwischen der DMZ und dem internen Netz. Der externe Paketfilter regelt den Verkehr zwischen der DMZ und dem Rest. Dies ähnelt der multiplen DMZ (siehe den nächsten Abschnitt).

[CE9] War hier “ein” Paketfilter oder “Paketfiltern” gemeint?

Editor's or typesetter's annotations (will be removed before the final TeX run)

3.3 Multiple DMZ

Den Begriff der multiplen DMZ verwende ich, um eine mehrstufige Firewall zu beschreiben, die über mehrere DMZs verfügt. Abbildung 3.3 zeigt das Prinzip.

Bei der multiplen DMZ existiert kein Single-Point-of-Failure. Die in Abbildung 3.3 dargestellte Architektur wird durch zwei Screening-Router realisiert, die keine direkte Verbindung besitzen. Anstatt die beiden Screening-Router direkt miteinander zu verbinden, verfügen die Application-Level-Gateways über zwei Netzwerkkarten und stehen mit jeweils einem Bein in der inneren und der äußeren DMZ. Kann der Angreifer den äußeren Screening-Router überwinden, so ist das interne Netz immer noch durch die Application-Level-Gateways und den inneren Screening-Router geschützt. Selbst wenn der Angreifer direkt ein Application-Level-Gateway angreifen könnte, wird das interne Netz immer noch von dem inneren Screening-Router geschützt. Die angebotenen Dienste besitzen nur eine Netzwerkkarte und befinden sich in der äußeren DMZ. Falls diese Dienste auf interne Datenbanken zugreifen müssen, ist der Zugriff nur über die Application-Level-Gateways möglich, oder diese Systeme benötigen für diesen Zugriff ein weiteres Bein in der inneren DMZ.

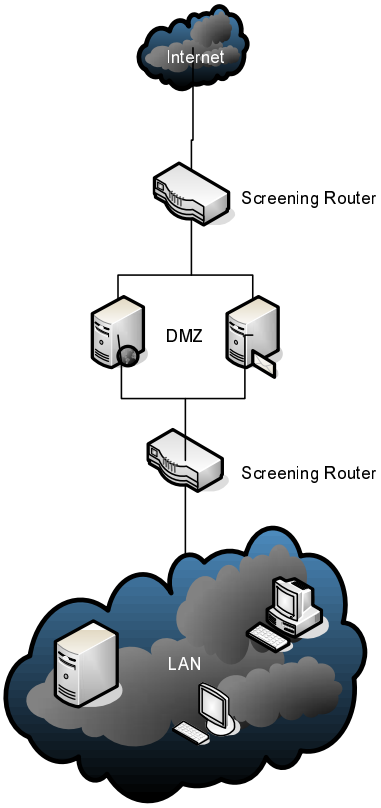


Abbildung 3.3: **Bei einer multiplen DMZ gibt es keinen Single-Point-of-Failure.**

Der schiere Hardware-Aufwand und der damit einhergehende Administrationsaufwand lassen diese Lösung natürlich nur für Netze mit einem hohen Schutzbedürfnis adäquat erscheinen. Natürlich sind auch zahlreiche Variationen der Architektur denkbar.

Der Vorteil dieser Architektur ist das Fehlen des Single-Point-of-Failure. Sie haben hoffentlich bei dem erfolgreichen Angriff auf die Firewall genügend Zeit, mit entsprechenden Maßnahmen (Unterbrechung der Netzwerkverbindung) zu reagieren.

3.4 Wahl der Architektur

Die Wahl der richtigen Architektur ist nicht einfach. Verabschieden Sie sich von dem Gedanken einer 100%ig sicheren Firewall. Diese mag in Hochglanzprospekten der Hersteller existieren. In der Realität habe ich sie noch nie gefunden. Falls Sie einen Hersteller finden, der Ihnen diese verspricht, fragen Sie ihn, warum er Ihnen dann auch noch ein Intrusion-Detection-System (IDS) und einen Virenscanner verkaufen möchte.

Die Wahl der Architektur ist entscheidend für die Standhaftigkeit der Firewall. Vergleichen Sie Ihr Netzwerk mit einer Stadt aus dem Mittelalter. Im Mittelalter war jede Stadt in Bezug auf ihren Schutz auf sich allein gestellt. Es gab keine Polizeigewalt, die den Austausch von Waren und das Reisen zwischen den Städten überwachte. Vielmehr regierten außerhalb der Städte meistens Raubritter und Diebe. Es herrschte Anarchie, wie es heute in vielen Bereichen des Internets der Fall ist. Sobald Sie Ihr Netz mit dem Internet verbinden, sind Sie daher bezüglich der Sicherheit Ihres Netzes auch auf sich allein gestellt. Sie müssen sich, wie die Städte im Mittelalter, gegen jeden nur möglichen Angriff verteidigen. Im Mittelalter bauten die Stadtbewohner daher Mauern und zogen Gräben. Meist wurde nicht nur eine Mauer oder ein Graben gezogen, damit die Verteidiger ausreichend Zeit für die Vorbereitung der Verteidigung erhielten. Musste der Angreifer nur eine Mauer überwinden, benötigte er nur genügend lange Leitern, um den Angriff erfolgreich durchzuführen. Gute Verteidigungsanlagen verfügten über mehrere Mauern und Gräben und deren Kombination, sodass die Mauer zwar mit einer Leiter überwunden werden konnte, aber der Angreifer für den Graben Boote benötigte und die Leitern nicht einsetzen konnte. Die Verteidiger hatten in der Zeit die Gelegenheit, ihr Öl zum Sieden zu bringen, um es über die Angreifer zu gießen.

Nutzen auch Sie daher in Ihrer Firewall-Architektur unterschiedliche Technologien zur Verteidigung, und bauen Sie mehrere Schutzwälle auf. Auch heute zeigt sich wie im Mittelalter, dass Angreifer häufig nur die schlecht geschützten Ziele angreifen. Diese werden auch als „low hanging fruits“ (niedrig hängende Früchte) bezeichnet.

Gegen derartige Angriffe sind Sie mit einer mehrstufigen Firewall gut geschützt. Falls ein Angreifer Sie tatsächlich gezielt angreift, bietet ein mehrstufiges System Ihnen hoffentlich die notwendige Zeit zur Vorbereitung der Verteidigung, bevor der Angreifer in Ihr Netz gelangt.

4. Bedrohungen bei der Vernetzung von Systemen

Solange die Rechner nicht vernetzt waren, war die Welt noch in Ordnung. Angriffe beschränkten sich auf physikalische Zugänge. Solange die Systeme für Unbefugte unzugänglich aufgestellt wurden, war ihre Sicherheit gewährleistet. Die physikalische Sicherheit des Rechnergehäuses und des Serverraums garantierte auch die Sicherheit der vorhandenen Daten. Sobald jedoch die Rechner eine größere Verbreitung fanden und Massenspeicher in Form von Disketten zum Austausch von Daten eingesetzt wurden, tauchte eine neue Bedrohung in Form von Viren auf. Der erste funktionsfähige Virus wurde von Fred Cohen in seiner Doktorarbeit 1983 beschrieben.[1] Die tatsächliche Verbreitung begann mit der Verbreitung des IBM-PCs und dem zunehmenden Informationsaustausch per Diskette und per Modem.

Durch die Vernetzung haben die Bedrohungen stark zugenommen. Der Angriff ist immer, zu jeder Zeit und von überall aus fast anonym durchführbar. Um sich für die Verteidigung zu rüsten, ist es wichtig, die Bedrohungen zu kennen und zu verstehen.

4.1 Angreifer und Motivation

Zunächst sollten Sie sich Gedanken über den potenziellen Angreifer und seine Motivation machen. Hieraus ergeben sich anschließend direkt Schlussfolgerungen zum erforderlichen Aufwand für die Sicherheit.

Die meisten Administratoren denken bei einem Angreifer zunächst an den klassischen Hacker. In den seltensten Fällen ist dieser jedoch für den Angriff verantwortlich.

4.1.1 Der klassische Hacker

Der klassische Hacker interessiert sich meist aus Neugierde für die interne Funktion eines Programms oder eines Betriebssystems. Der Hacker unterscheidet sich von dem Cracker, der – mit ähnlichem Wissen ausgestattet – versucht, in fremde Systeme einzudringen und dort Schaden anzurichten. Der Hacker verfügt über sehr spezialisiertes Wissen über die verwendeten Programmiersprachen und Algorithmen. Er analysiert die Softwarefunktionen und

1 Allerdings existierten auch schon früher Programme, die viele Anzeichen eines Virus aufwiesen. Diese Arbeit analysierte aber erstmalig wissenschaftlich das Phänomen.

findet dabei Programmier-, Logik- oder Designfehler. Anschließend versucht er, diese Fehler auszunutzen, um das Programm zu anderen Zwecken zu gebrauchen. Ist er erfolgreich, so hat er einen sogenannten Exploit gefunden. Seine Motivation ist meist Neugierde, Wissensdrang, Weiterbildung und „weil es möglich ist".

Der klassische Hacker ist normalerweise nicht daran interessiert, tatsächlich in fremde Systeme einzudringen. Er veröffentlicht meist die für den Angriff erforderlichen Informationen (teilweise gegen Bezahlung). Der Chaos Computer Club hat auf seiner Webpage (`http://www.ccc.de/hackerethics`) eine Hacker-Ethik veröffentlicht, mit der sich wahrscheinlich die meisten Hacker identifizieren können.

4.1.2 Skript-Kiddies

Der Begriff Skript-Kiddie bezeichnet Angreifer, die nicht selbst in der Lage sind, eine Sicherheitslücke zu finden, zu analysieren und einen Exploit zu entwickeln. Diese Angreifer sind darauf angewiesen, dass ein anderer Hacker einen Exploit entwickelt, den sie anschließend verwenden. Häufig wird dazu der Exploit in ein Skript eingebunden. Dieses Skript wird anschließend genutzt, um eine Vielzahl von Systemen anzugreifen und dort einzubrechen.

Die Tatsache, dass von „Kiddies" gesprochen wird, hat keinen Bezug auf das Alter der Personen. Skript-Kiddies gibt es in jedem Alter. Der Begriff soll eher abfällig zum Ausdruck bringen, dass das Skript-Kiddie nicht erfahren genug ist, den Exploit selbst zu entwickeln.

Skript-Kiddies werden meist durch Neugierde und die Suche nach Anerkennung in Hacker- und Möchtegern-Hackerkreisen motiviert. Hier versucht ein Skript-Kiddie dadurch auf sich aufmerksam zu machen, dass es in besonders viele Systeme eingedrungen ist.

4.1.3 Insider

Eine zweite sehr gefährliche Gruppe von potenziellen Angreifern sind die Insider. Hier handelt es sich zum Beispiel um entlassene Mitarbeiter, die über internes Wissen verfügen, oder aber auch um enttäuschte Mitarbeiter, die Rache üben möchten. Des Weiteren kann es sich um Zulieferer, Berater oder ähnliche Personen handeln. Diese Personen sind deshalb so besonders gefährlich, da sie über interne Informationen verfügen, die anderen Angreifern nicht zugänglich sind. Dadurch können sie unter Umständen Sicherheitsvorkehrungen umgehen oder besser angreifen. Meist müssen diese Personen sich keine Gedanken um die Firewall machen, da sie über Zugänge verfügen, die nicht von der Firewall überwacht werden. Dabei kann es sich um den lokalen Zugang im Netz oder um VPN-Zugänge handeln.

Auch wenn diese Gruppe häufig nicht über so hohes technisches Wissen in Bezug auf Sicherheitslücken wie ein Hacker verfügt, ist ihre kriminelle Energie meist höher einzustufen.

4.1.4 Mitbewerber

Auch Mitbewerber gehören zu den potenziellen Angreifern. Dabei reichen teilweise sicherlich auch Denial-of-Service-Angriffe aus, um einen wirtschaftlichen Vorteil zu erhalten. Stellen Sie sich vor, Sie suchen einen neuen Partner für das Outsourcing Ihrer E-Mail-Kommunikation

inklusive SPAM- und Virenschutz. In der heißen Phase der Verhandlung mit zwei potenziellen Partnern ist eines der beiden Unternehmen per E-Mail nicht mehr erreichbar. Welchem der beiden potenziellen Partner vertrauen Sie mehr? Unternehmen A, das gerade unter einem Denial-of-Service-Angriff (DoS) leidet, oder Unternehmen B, das diesen DoS durchführt oder in Auftrag gegeben hat?

Natürlich sind Ihre Mitbewerber aber auch an Ihren Daten interessiert. Wenn sich also die Gelegenheit ergibt, Zugang zu Ihrer Datenbank zu erhalten, den E-Mail-Verkehr mit Ihren Kunden mitzulesen oder Zugang zu neuen Produktentwicklungen zu erhalten, sind Ihre Mitbewerber sicherlich sehr interessiert.

Ihre Mitbewerber haben meist auch die finanziellen Möglichkeiten, hochqualifizierte Angreifer (Industriespione) zu bezahlen. So können Ihre Mitbewerber selbst dann zur Gefahr werden, wenn sie selbst über kein entsprechendes Wissen verfügen.

4.1.5 Geheimdienste

Die Geheimdienste waren schon immer an allen Informationen interessiert. Sie hören die Kommunikation zwischen Regierungen, aber auch zwischen Unternehmen ab. Häufig betreiben sie auch Wirtschaftsspionage mit nationalem Interesse, um Unternehmen aus dem eigenen Land einen Wettbewerbsvorteil zu verschaffen. Die Nachrichtendienste verfügen meist über ausreichend finanzielle Mittel, die erforderlichen Kenntnisse sowie die notwendigen Netzwerke und Computer, um Angriffe und Einbrüche durchzuführen. Sie sind auch in der Lage, schwach verschlüsselte Informationen sehr einfach und schnell zu entschlüsseln.

Schließlich besitzen sie häufig auch die Möglichkeit, sich physikalischen Zugang zu bestimmten Informationen zu verschaffen, wenn ein Computerangriff nicht erfolgreich ist. Häufig ist ein physikalischer Einbruch sogar einfacher.

4.1.6 Terroristen und organisierte Kriminalität

Auch Terroristen und das organisierte Verbrechen erkennen im Internet immer mehr ein Potenzial für ihre Geschäfte – sowohl, um eigene Geschäfte über das Internet abzuwickeln, als auch, um durch Angriffe und Einbrüche an sensitive Daten heranzukommen. So sind die Phishing-Angriffe, die in den Jahren 2004 und 2005 begannen, sicherlich noch nicht vorbei. Beim Phishing versuchen organisierte Banden, gezielt an Login-Informationen zu gelangen. Auch der Versand von SPAM ist immer mehr ein Geschäft von wenigen organisierten Banden. Dabei werden kompromittierte Rechner unwissender Anwender genutzt, um in großer Zahl E-Mails zu versenden, die den Empfänger zum Kauf von Viagra bewegen sollen.

2004 wurde auch bekannt, dass ein Erpresserring versucht hat, Online-Wettbüros mit Denial-of-Service-Angriffen zu erpressen. Der Erpresserring verfügte angeblich über Kontakte zur russischen Mafia (siehe c't 14/2004 und `http://www.heise.de/ct/04/14/048/default.shtml` und `http://www.heise.de/newsticker/meldung/49292`).

Viele Angriffe sind auch politisch motiviert. So wurden nach dem 11. September 2001 besonders viele Angriffe auf muslimische Websites unternommen.

Bitte bestätigen Sie die Änderungen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

Derartige Angriffe werden sicherlich in nächster Zukunft stark zunehmen und weitere Medien entdecken. Ich selbst befürchte schon für das Medium VoIP ähnliche Entwicklungen, wie wir sie im Moment bei E-Mail mit SPAM erleben. Der Begriff für VoIP-SPAM ist bereits gefunden worden: Diese Form wird als SPIT (SPAM via Internet-Telefonie) bezeichnet.

4.2 Tendenzen und Entwicklungen

Wenn man nun die potenziellen Angreifer, ihr Know-how und ihre kriminelle Energie betrachtet, kann man recht einfach die Grafik aus Abbildung 4.1 erzeugen. Diese Grafik zeigt, dass das größte Gefahrenpotenzial von den Geheimdiensten und den Industriespionen ausgeht, die von Mitbewerbern angeheuert werden.

Allerdings sollten Sie auch die Insider nicht aus den Augen verlieren. Ihre hohe kriminelle Energie macht sie besonders gefährlich. Die Skript-Kiddies sind meist zu vernachlässigen. Sie führen keine gezielten Angriffe auf Sie durch, sondern suchen meist allgemein nach verwundbaren Systemen. Sie suchen die niedriger hängenden Früchte, die leicht zu erreichen und anzugreifen sind. Ein gezielter Angriff gegen Ihre Systeme durch Skript-Kiddies ist unwahrscheinlich, wenn Sie Ihre Hausaufgaben gemacht haben, Ihre erreichbaren Systeme ordent-

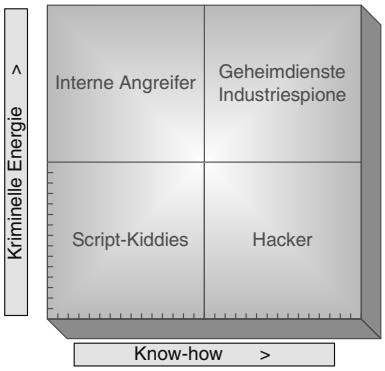


Abbildung 4.1: **Das Risiko eines erfolgreichen Angriffs steigt mit steigender krimineller Energie und steigendem Know-how.**

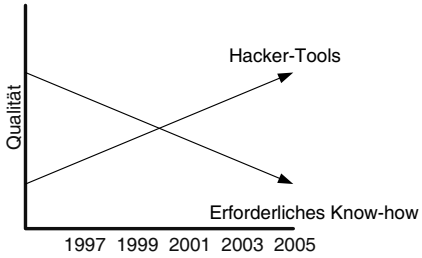


Abbildung 4.2: **Die Qualität der Angriffswerkzeuge nimmt seit vielen Jahren stark zu. Das für einen Angriff notwendige Know-how sinkt daher stetig.**

CE: Mir ist (ohne diese Änderung) der Zusammenhang im Satz nicht klar, zumal Sie oben sagen, dass das Risiko durch Skript-Kiddies (=Durchschnittseindringling?) abnimmt.

Editor's or typesetter's annotations (will be removed before the final TeX run)

lich gepatcht sind und Sie über eine Firewall verfügen. Dann zählen Ihre Systeme nicht zu den niedrig hängenden Früchten. Dennoch müssen Sie möglicherweise mit gezielten Angriffen durch Mitbewerber, Industriespione etc. rechnen. Hier kommt erschwerend hinzu, dass in den letzten Jahren die Angriffswerkzeuge immer besser geworden sind, sodass sich auch weniger technisch versierte Konkurrenten als Industriespione versuchen könnten. (siehe Abbildung 4.2).

4.3 Schutzziele

In diesem Abschnitt möchte ich ganz allgemein die zu schützenden Ziele beschreiben. Im Grunde gibt es drei Ziele bei dem Schutz von Netzen, Rechnern und Daten:

» Integrität
» Vertraulichkeit
» Verfügbarkeit

Datenintegrität bedeutet, dass eine Änderung der Daten durch Unbefugte nicht unbemerkt bleibt. Die Integrität von Daten wird üblicherweise durch kryptografische Prüfsummen erreicht.

Die Vertraulichkeit der Daten wird meist mithilfe von eingeschränkten Leserechten und zusätzlicher Verschlüsselung erreicht. Nur autorisierte Personen erhalten den Schlüssel für die Entschlüsselung. Das Abhören einer Kommunikation oder das Ausspähen von Daten wird so unmöglich.

Die Verfügbarkeit von Daten wird häufig vernachlässigt. Dieses Ziel verlangt, dass die gewünschten Daten jederzeit verfügbar sind. Hier werden häufig technische Hilfsmittel, wie Backups, unterbrechungsfreie Stromversorgungen, Cluster etc. genutzt, um die Verfügbarkeit zu sichern.

Sie mögen sich fragen, was diese Ziele nun mit einer Firewall zu tun haben. Eine Firewall ist auch ein Werkzeug, um diese Sicherheitsziele zu erreichen. Sie sichert die Integrität der lokal gespeicherten Daten, indem sie nichtautorisierte Kommunikationsanfragen, zum Beispiel an den Datenbankserver, ablehnt. Sie stellt die Vertraulichkeit von Daten sicher, indem sie verhindert, dass bestimmte Kommunikationsverbindungen von und nach außen aufgebaut werden können. Schließlich kann sie auch die Verfügbarkeit der Daten durch die Abwehr eines Denial-of-Service-Angriffs garantieren oder selbst hochverfügbar die Internetanbindung sicherstellen.

In größeren Unternehmen gibt es neben diesen sehr technischen Zielen auch formale Ziele. Hierzu gehören zum Beispiel die Nachvollziehbarkeit der Datenverarbeitung. Jeder Vorgang, zum Beispiel in einer Bank, muss nachvollziehbar sein. Auch die Revisionssicherheit spielt hier häufig eine große Rolle. Die Revisionssicherheit verlangt zusätzliche Protokolle, Erhebungen, Inventuren und spezielle Formen der Datenspeicherung und -haltung. Schließlich spielen auch gesetzliche Rahmenbedingungen eine große Rolle. Das können einfache Gesetze sein, wie zum Beispiel der Datenschutz personenbezogener Daten, aber auch sehr kom-

plexe gesetzliche Rahmenbedingungen, wie Basel-II und Basel-III, MaRisk, GOBS und der Sarbanes-Oxley-Act. Da viele Unternehmen heute global arbeiten, unterliegen sie auch international gültigen Gesetzen.

Auch bei der Einhaltung und Erfüllung dieser Rahmenbedingungen kann eine Firewall eine wichtige Aufgabe spielen. Gerade die Protokollierung durch eine Firewall wird hier häufig betroffen sein. Während der Datenschutz das Protokollieren personenbezogener Daten hier kritisch sehen wird, werden andere Personen in den Protokollen wertvolle Hilfsmittel zur Revisionssicherheit und Sicherstellung der Nachvollziehbarkeit sehen.

Falls Sie nicht eine Firewall für ein Unternehmen der Fortune500 aufbauen wollen, sind diese letzten Überlegungen möglicherweise für Sie uninteressant. Dennoch sollten Sie zum Beispiel den Datenschutz nicht aus dem Blickfeld verlieren. Über diesen CE müssen Sie sich bei einer Firewall möglicherweise ausführlich Gedanken machen.

4.4 Angriffsmethoden

Die meisten Angriffe lassen sich in wenigen Gruppen kategorisieren. Im Folgenden stelle ich Ihnen die wichtigsten Kategorien vor.

STOP

Ich konzentriere mich im Weiteren auf die sehr technische Seite der möglichen Angriffe über ein Netzwerk. Daneben gibt es natürlich auch noch:

Social Engineering. *Hier versucht der Angreifer, durch soziale Kompetenz vertrauliche Informationen, wie zum Beispiel Kennwörter, zu erhalten. Hierzu täuscht er einen legitimen Benutzer, einen Administrator oder andere vertrauenswürdige Personen vor. Kevin Mitnick hat ein sehr eindrucksvolles Buch zu diesem Thema geschrieben (*The Art of Deception*, Kevin Mitnick, 2003).*

War Dialing. *Anstatt den direkten Zugang durch eine Firewall zu suchen, bemüht sich der Angreifer, alternative Zugänge über unbewachte Modem- oder ISDN-Zugänge zu finden. Hierfür verwendet er Software, die jede erdenkliche Telefonnummer eines Unternehmens anwählt und die Antwort testet.*

Zunächst gibt es die große Menge der *Denial-of-Service-Angriffe*. Hier versucht der Angreifer nicht, die Kontrolle über das System zu erhalten, sondern die Funktion des Systems zu stören. Beim *Spoofing* täuscht der Angreifer eine falsche Identität als Client oder Server vor. Beim *Hijacking* versucht ein Angreifer, eine aufgebaute Verbindung zu kapern. Ein *Bufferoverflow* kann auch einen Denial-of-Service auslösen. Jedoch versucht der Angreifer, wenn möglich mit einem Bufferoverflow die Kontrolle über das System zu übernehmen. Der Bufferoverflow wird durch Programmierfehler in den betroffenen Applikationen möglich. Der *Formatstring-Angriff* ist dem Bufferoverflow sehr ähnlich und erlaubt dem Angreifer ebenfalls häufig die Übernahme der Kontrolle. Bei der *Race-Condition* handelt es sich um einen Fehler in der Programmlogik. Hierbei konkurrieren zwei Prozesse um eine Ressource. Wenn der Programmierer die Ressource nicht richtig vor dem Zugriff des zweiten Prozesses geschützt hat, kann der

CE Bitte bestätigen.

Angreifer dies ausnutzen. Die *SQL-Injektion* ist ein typischer Angriff auf Webserver mit dynamischen datenbankgestützten Webseiten. Hier versucht der Angreifer, die von der Webapplikation verwendeten SQL-Abfragen zu modifizieren. So kann er auf Daten zugreifen, die die Webapplikation normalerweise nicht ausgeben würde, oder sogar die Daten in der Datenbank modifizieren.

Im Folgenden werden die verschiedenen Angriffsmethoden ausführlicher dargestellt. Ich werde Ihnen dann auch verschiedene Möglichkeiten der Abwehr nennen.

4.4.1 Denial-of-Service (DoS)

Bei einem Denial-of-Service stört der Angreifer die Funktion eines Dienstes so, dass ein legitimer Nutzer nicht mehr auf diesen Dienst zugreifen kann. Im einfachsten Fall verübt bereits eine Putzfrau, die mit ihrem Staubsauger das Kabel eines Servers aus der Steckdose zieht, unwissentlich einen DoS.

Es gibt viele verschiedene Möglichkeiten, einen DoS durchzuführen. Diese entsprechen den verschiedenen Schichten, auf denen die Kommunikation erfolgt.

» Der Angreifer kann versuchen, die Netzwerkbandbreite des Servers komplett auszulasten, sodass weiterer Verkehr nur sehr langsam transportiert wird. Dieser Angriff erfolgt also auf der physikalischen Schicht und ist durch deren Bandbreite begrenzt. Es ist heute sehr schwierig geworden, diesen Angriff durchzuführen, da die Bandbreiten in den letzten Jahren stark zugenommen haben. Der Angreifer benötigt enorme Ressourcen, um sein Ziel zu erreichen.

  Allerdings ist es mit sogenannten Bot-Netzen sehr wohl möglich, den nötigen Netzwerkverkehr zu generieren. Bot-Netze sind gekaperte CE^k Rechner (meist unter Microsoft Windows), die zu Tausenden in einem Netz zusammengeschlossen wurden und gemeinsam von dem Angreifer gesteuert werden. Ein Bot-Netz wird häufig für den Versand von SPAM-E-Mails verwendet.

  Ein Schutz vor diesem Angriff ist nur mit Unterstützung des Providers möglich, der den Verkehr vielleicht vorher filtern kann. Werden jedoch die Absenderadressen der Pakete gefälscht und diese Pakete von vielen verschiedenen Systemen verschickt, ist das sehr schwer.

» Der nächste mögliche DoS-Angriff greift den TCP-Stack des Servers an und führt einen SYN-Flood durch. Der Angreifer generiert viele TCP-SYN-Pakete. Hierbei handelt es sich um Pakete, die einen Verbindungsaufbau anzeigen. Der Server antwortet auf dieses Paket mit einem SYN/ACK-Paket. Damit sich der Server später an diesen Verbindungsaufbau erinnern kann, speichert er die Verbindungsinformationen in einer Tabelle ab, in der alle schwebenden Verbindungen vorgehalten werden (Pending Connections). Sobald das dritte Paket, das TCP-ACK-Paket des Clients, erhalten wird, erhält die Verbindung den Status einer aufgebauten Verbindung und wird aus der Pending-Connections-Tabelle gelöscht und in der Tabelle der aufgebauten Verbindungen (Established Connections) eingetragen.

  Bei einem SYN-Flood überflutet der Angreifer den Server mit einer Vielzahl von TCP-SYN-Paketen. Hierbei fälscht der Angreifer zufällig die Absenderadresse der Pakete. Sinnvoll

CE^k Bitte bestätigen Sie diese Änderung.

Editor's or typesetter's annotations (will be removed before the final TEX run)

sind hier Absenderadressen von nicht existenten Rechnern. Der Server wird auf alle TCP-SYN-Pakete mit einem TCP-SYN/ACK-Paket antworten und diese Verbindungen in die Pending-Connections-Tabelle eintragen. Diese Tabelle weist jedoch (ähnlich allen anderen Tabellen in Computern) nur eine begrenzte Größe auf. Sendet der Angreifer mehr Pakete, als Verbindungen in dieser Tabelle gespeichert werden können, so muss der Server Verbindungen aus dieser Tabelle löschen.

Erfolgt gleichzeitig zum SYN-Flood ein korrekter Verbindungsaufbau durch einen echten Client, so kann der Server nicht zwischen dieser Verbindung und den Verbindungen des Angreifers unterscheiden. Sendet der Angreifer die TCP-SYN-Pakete so schnell, dass der Server auch die echte Verbindung aus der Tabelle der Pending Connections entfernen muss, bevor das dritte Paket des TCP-Handshakes vom Client empfangen wird, so wird der Server dieses Paket zurückweisen, da er keine Informationen über die Verbindung mehr besitzt. Die Verbindung wird abgebrochen und der Dienst steht nicht mehr zur Verfügung.

Viele TCP-Implementierungen brechen ab 100 TCP-SYN-Paketen pro Sekunde zusammen. Linux bietet die SYN-Cookies. Diese erlauben immer noch einen Verbindungsaufbau bei einem SYN-Flood mit bis zu 100.000 SYN-Paketen pro Sekunde. Hierbei wird die Tabelle der Pending Connections ignoriert und lediglich auf Basis der Sequenznummer über einen Verbindungsaufbau entschieden. Die Funktionsweise wird in Abschnitt 23.3.6 beschrieben.

4.4.2 Spoofing

Spoofing bezeichnet Angriffe, bei denen der Angreifer bestimmte Informationen fälscht. Meist erfolgt dies, um die Identität eines anderen Rechners anzunehmen. Drei verschiedene Arten des Spoofings werden heute durchgeführt. Hierbei handelt es sich um:

IP-Spoofing. Bei dem IP-Spoofing verändert der Angreifer den IP-Header seiner Pakete. Meist fälscht er seine Absender-IP-Adresse. Hiermit kann er die Identität eines anderen Rechners annehmen. Dies erfolgt, um entweder seine Spuren zu verwischen oder um Vertrauensstellungen zwischen gewissen Rechnern auszunutzen.

ARP-Spoofing. Diese Variante wird besonders von Angreifern in geswitchten Netzen eingesetzt, um trotz des Einsatzes eines Switches weiterhin sämtliche ausgetauschten Pakete zu protokollieren. Hierbei werden ARP-Antworten gefälscht. Außerdem wird es für ein TCP-Session-Hijacking verwendet (s.u.).

DNS-Spoofing. Hierbei fälscht ein Angreifer die Zuordnung eines DNS-Namens zu einer IP-Adresse, indem er die Antwort eines DNS-Servers fälscht.

IP-Spoofing

Dies ist die älteste Variante des Spoofings. Hierbei täuscht der Angreifer eine andere IP-Adresse als Absender vor. Dies wird zum Beispiel beim SYN-Flood verwendet, um die eigenen Spuren zu verwischen.

Das IP-Spoofing kann jedoch auch für einen direkten Angriff genutzt werden. Bei dem SMURF-Angriff sendet der Angreifer ein ICMP-Echo-Request-Paket an die Broadcast-Adresse eines Netzwerks. Sämtliche UNIX-Rechner dieses Netzwerks reagieren mit einem ICMP-Echo-Reply-Paket. Der Angreifer erhält hiermit also eine Multiplikation seiner Pakete. Fälscht der Angreifer nun die Absenderadresse so, dass sie die IP-Adresse des anzugreifenden Rechners darstellt, antworten alle UNIX-Rechner bei einem Broadcast-Ping an diesen Rechner. Erfolgt dies häufig genug, so besteht die Möglichkeit, die Netzwerkverbindung des angegriffenen Rechners zu überlasten. Jedoch ist die Antwort auf ein Broadcast-Echo-Request-Paket nur eine Eigenschaft von UNIX-Systemen, die heute meist von dem Hersteller oder dem Administrator abgestellt wird.

Schließlich kann das IP-Spoofing auch verwendet werden, um Vertrauensstellungen zwischen Rechnern auszunutzen. Das UDP-Protokoll bietet keinen Schutz vor gespooften Paketen, da es nicht verbindungsorientiert arbeitet. Bei einer TCP-Verbindung genügt nicht das Fälschen der IP-Adresse. Der Angreifer muss darüber hinaus auch die Sequenz- und Acknowledgement-Nummern spoofen. Erlaubt zum Beispiel ein Syslogd-Server die Protokollierung von Meldungen mit dem UDP-Protokoll nur bestimmten IP-Adressen, so genügt es in diesem Fall, wenn der Angreifer die IP-Adresse entsprechend fälscht. Das Paket wird dann akzeptiert und die Meldung dementsprechend protokolliert, als käme sie von dem korrekten Rechner.

Ein Schutz vor IP-Spoofing ist nur auf einer Firewall möglich. Hierbei sollten Sie darauf achten, dass die Firewall keine unzulässigen IP-Adressen akzeptiert. Wenn in Ihrer DMZ Adressen aus dem Netzwerk 192.168.0.0/24 verwendet werden, dann dürfen Anfragen aus diesem physikalischen Netz keine andere Absenderadresse verwenden.

ARP-Spoofing

Beim ARP-Spoofing verfolgt der Angreifer entweder das Ziel, ein TCP-Session-Hijacking durchzuführen (s.u.) oder in einer Umgebung, die durch einen Switch kontrolliert wird, dennoch einen Netzwerksniffer einzusetzen.

Der zweite Angriff soll hier ein wenig ausführlicher betrachtet werden.

Wenn zwei Netzwerkgeräte sich mit dem IP-Protokoll unterhalten möchten, so benötigen sie in einem Ethernet-Netzwerk für die eigentliche Kommunikation zunächst auch noch die Ethernet-MAC-Adressen. Hierfür ist das ARP-Protokoll zuständig. Der Rechner, der ein IP-Paket an die IP-Adresse des Zielsystems senden möchte, sendet zunächst einen ARP-Request, um die dazugehörige MAC-Adresse zu ermitteln. Anschließend sendet er das IP-Paket an die IP-Adresse des Zielsystems und den Ethernet-Rahmen an die MAC-Adresse des Zielsystems.

Wird aus Geschwindigkeitsgründen in diesem Netzwerk ein Switch eingesetzt, so wird dieser zunächst den ARP-Request, da dieser an die Broadcast-Ethernet-Adresse gerichtet ist, an alle angeschlossenen Geräte weiterleiten. Der anschließend versandte Ethernet-Rahmen mit dem IP-Paket wird jedoch vom Switch nur an das tatsächliche Ziel versandt. Um dies zu ermöglichen, besitzt der Switch eine Liste, in der MAC-Adressen der angeschlossenen Geräte abgespeichert werden.

Bei dem ARP-Spoofing-Angriff (siehe Abbildung 4.3) fälscht der Angreifer (Laptop) den ARP-Reply.

1. Der Rechner 192.168.0.5 möchte ein Paket an den Rechner 192.168.0.6 senden. Er sendet einen ARP-Request: Welche MAC-Adresse hat der Rechner 192.168.0.6?
2. Der Angreifer beantwortet diese Anfrage nun. Hierzu fälscht er die Antwort, indem er Folgendes sendet: Der Rechner mit der IP-Adresse 192.168.0.6 besitzt die MAC-Adresse 7.
3. Der Rechner 192.168.0.5 sendet nun sein IP-Paket an: Ziel-IP: 192.168.0.6; Ziel-MAC: 7. Der Switch ist nicht in der Lage, die IP-Adressen zu lesen. Er arbeitet lediglich auf der Ebene der MAC-Adressen. Er schlägt in seiner Tabelle nach und stellt fest, dass das Netzwerkgerät mit der MAC-Adresse 7 an Port 4 angeschlossen ist, und leitet das Paket an diesen Port weiter.
4. Der Angreifer muss auf seinem Rechner eine Routing-Software aktivieren. Der Rechner des Angreifers erhält nun das Paket und verarbeitet es, da es an die MAC-Adresse seiner Netzwerkkarte gerichtet ist. Es ist jedoch nicht an seine IP-Adresse gerichtet. Dies ist typisch für einen Router. Wurde das Routing aktiviert, so leitet dieser Rechner das Paket nun an die IP-Adresse 192.168.0.6 und die MAC-Adresse 6 weiter. Als Absender trägt er die IP-Adresse 192.168.0.5 und seine eigene MAC-Adresse 7 ein. Der Switch wird dieses Paket nun beim richtigen Empfänger zustellen. Dieser wird das Paket verarbeiten und umgekehrt antworten.

Es existieren fertige Werkzeuge, die in der Lage sind, unter Linux ein ARP-Spoofing durchzuführen. Ein Schutz vor ARP-Spoofing ist nur durch die folgenden Maßnahmen erreichbar:

Statische ARP-Tabellen. Die Pflege dieser Tabellen ist aber sehr aufwendig. Sie können unter Linux einen statischen Eintrag mit dem Befehl `arp -s "ip" "mac"` erzeugen.

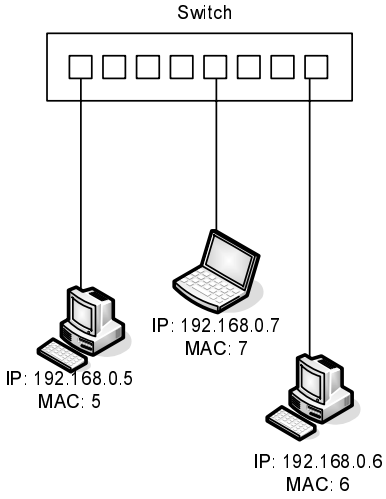


Abbildung 4.3: **ARP-Spoofing bei einem Switch**

ARPwatch. Der ARPwatch-Daemon ist in den meisten Linux-Distributionen enthalten und ist eine Art ARP-Intrusion-Detection-System. Er überwacht alle ARP-Pakete und meldet Änderungen und damit auch Angriffe.

Multilayer-Switches bieten die Möglichkeit, die Zuordnung der IP- und MAC-Adressen zu überwachen und bei dem Auftreten von ARP-Spoofing die entsprechenden Ports zu deaktivieren. Cisco nennt diese Funktion *Dynamic ARP Inspection*.

DNS-Spoofing

Das DNS-Spoofing führt einen Angriff auf DNS-Ebene durch, der dem Angriff auf ARP-Ebene gleicht. Ein Beispiel wird in Abbildung 4.4 dargestellt.

Beim DNS-Spoofing sind die folgenden Schritte erforderlich:

1. Der Client 192.168.0.5 möchte eine Netzwerkverbindung mit dem Rechner *www.sparkasse.de* aufbauen. Hierzu benötigt er zunächst die IP-Adresse dieses Rechners. Um diese zu ermitteln, kontaktiert er seinen DNS-Server und fragt ihn nach der IP-Adresse von *www.sparkasse.de*.
2. Da für diese DNS-Anfrage üblicherweise zunächst das UDP-Protokoll eingesetzt wird, existiert kein Schutz gegen gespoofte Pakete. Der Angreifer antwortet anstelle des echten DNS-Servers (schneller als der echte DNS-Server) und teilt dem Client mit, dass der Rechner *www.sparkasse.de* die IP-Adresse 192.168.0.7 aufweist.
3. Der Client wird sich nun mit diesem Rechner verbinden und in Wirklichkeit den Rechner *www.sparkasse.de* erwarten. Damit der Client diese Illusion erhält, richtet der Angreifer auf dem Rechner einen Proxy ein, der sämtliche Anfragen an den echten Rechner weiterleitet und die Anworten an den Client übermittelt.

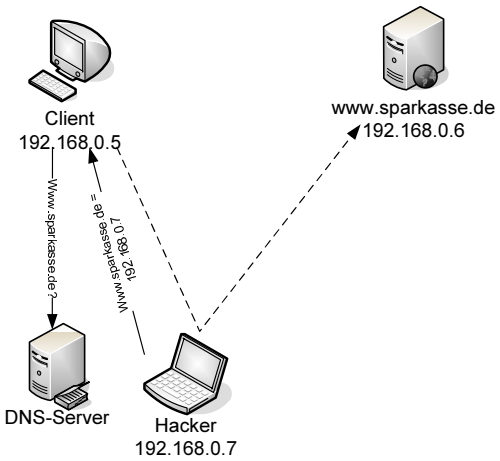


Abbildung 4.4: **DNS-Spoofing**

4. Dieser Angriff ist leicht durch die Verwendung von SSL zu vereiteln. Dazu ist aber zusätzlich erforderlich, dass der Client das SSL-Protokoll auch korrekt einsetzt und die Authentifizierung des Servers überprüft.

Es existieren fertige Werkzeuge, die in der Lage sind, unter Linux ein DNS-Spoofing durchzuführen. Auch existieren Werkzeuge, die dann als SSL-Proxy versuchen, eine SSL-Verbindung zu unterwandern. Wenn der Client die SSL-Zertifikate nicht richtig prüft, kann ein Angreifer so eine Verbindung abhören.

Gespoofter Portscan

Normalerweise ist ein gespoofter Portscan nicht möglich. Bei einem Portscan versucht der Angreifer, Daten über das untersuchte System zu ermitteln. Spooft er jedoch seine Absenderadresse, so erhält er nie das Ergebnis seines Scans. Verschiedene Werkzeuge wie zum Beispiel `nmap` (`http://www.nmap.org`) versuchen das Problem zu lösen, indem sie jedes Paket in einem Scan mehrfach senden. Alle zusätzlichen Pakete weisen eine gefälschte Absenderadresse (sogenannte Decoys) auf. Dies erschwert eine Analyse und Bestimmung des Ursprungs des Portscans sehr oder macht diese Bestimmung sogar unmöglich.

Jedoch gibt es auch die Möglichkeit, einen echten gespooften Portscan durchzuführen. Hierzu ist ein dritter Rechner und das Werkzeug `Hping` erforderlich. Dieser Rechner sollte gleichzeitig keine aktiven Netzwerkverbindungen besitzen. Daher wird er als *Silent Host* bezeichnet. Abbildung 4.5 demonstriert die Vorgehensweise.

Der Angreifer sucht zunächst einen sogenannten Silent Host. Dies ist ein Rechner, der gleichzeitig keine anderen Netzwerkverbindungen unterhält. Hierbei kann es sich zum Beispiel um einen Windows 98-Rechner einer asiatischen Universität handeln. Er beginnt nun, TCP-SYN-Flag-Pakete in regelmäßigen Abständen (1/Sekunde) an einen geschlossenen Port zu senden. Er erhält für jedes Paket ein TCP-RST/ACK-Paket zurück. Diese Pakete weisen eine steigende IP-Identifikationsnummer auf (siehe Abbildung 4.5)CE. Wenn der Rechner gleichzeitig keine weiteren Pakete versendet, so wird diese Nummer immer um 1 inkrementiert. (Achtung: Windows vertauscht die beiden Bytes der Identifikationsnummer. Unter Linux erscheint damit der Inkrementierungsschritt als 256.)

Nun beginnt der Angreifer mit dem Portscan. Er sendet mehrere TCP-SYN-Pakete in den gleichen regelmäßigen Abständen an das Opfer. Hierbei fälscht er die Absenderadresse so, dass

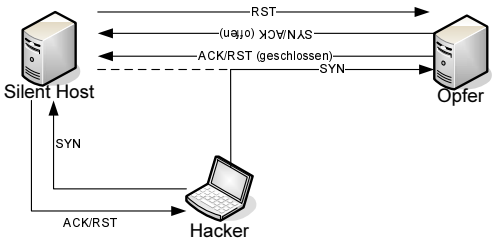


Abbildung 4.5: **Gespoofter Portscan**

CE Ist im Bild nicht dargestellt.

das Opfer seine Antworten an den Silent Host sendet. Es existieren nun zwei Möglichkeiten: Der Port ist offen oder der Port ist geschlossen.

1. **Offen:** Wenn der Port auf dem Opfer offen ist, so antwortet das Opfer mit einem TCP-SYN/ACK-Paket an den Silent Host. Dieser kann dieses Paket nicht zuordnen und sendet ein TCP-RST-Paket an das Opfer. Damit ist die Verbindung für alle beendet.
2. **Geschlossen:** Wenn der Port auf dem Opfer geschlossen ist, so antwortet das Opfer mit einem TCP-RST/ACK-Paket an den Silent Host. Dieser reagiert auf das Paket nicht mit einem weiteren Paket.

Da der Silent Host, wenn der Port auf dem Opfer offen war, nun weitere Pakete in regelmäßigen Abständen an das Opfer versendet, wird die IP-Identifikationsnummer der Pakete, die vom Silent Host an den Angreifer gesendet werden, immer um 2 inkrementiert. Dies ist ein Zeichen, dass der Port offen war. Ändert sich dies nicht, so war der Port geschlossen.

Das Opfer vermutet, dass es vom Silent Host gescannt wird. Wenn der Silent Host nicht durch eine Firewall überwacht wird, so kann die Herkunft des Portscans nicht festgestellt werden.

Nmap kann diesen Scan seit einigen Versionen auch durchführen. Hier heißt der Scan „Idle-Scan“ und der Silent-Host wird als „Zombie“ bezeichnet. Sie benötigen also nicht unbedingt `hping`, wenn Sie bereits Nmap installiert haben.

Vor diesem Scan können Sie sich nicht schützen. Sie sollten lediglich bei der Analyse der Protokolle Ihrer Firewall darauf achten, dass fast alle Informationen in den Protokollen falsch sein können.

4.4.3 Session Hijacking

Der Mitnick-Angriff hat die Verwundbarkeit des TCP-Protokolls vorgeführt, wenn die initialen Sequenznummern vorhergesagt werden können. Aber selbst dann, wenn das nicht der Fall ist, ist ein Übernehmen der Verbindung durch einen Angreifer (Session Hijacking) möglich. Hierzu ist es jedoch erforderlich, dass der Angreifer in der Lage ist, die Verbindung zu beobachten. Er kann dann die verwendeten TCP-Sequenznummern direkt von den ausgetauschten Paketen ablesen.

Durch eine sinnvolle Konstruktion von gespooften Paketen ist es dann möglich, eigene Daten in diese Verbindung zu injizieren. Dieser Angriff war in der Vergangenheit sehr kompliziert und schwer durchzuführen. Seit einigen Jahren existieren jedoch mehrere Werkzeuge, die dies stark vereinfachen. Die bekanntesten Werkzeuge sind `juggernaut` und `hunt`.

Im Folgenden soll schematisch die Funktionsweise von `hunt` erläutert werden.

1. Wenn `hunt` eine laufende Verbindung beobachtet, so ist es in der Lage, diese Verbindung zu übernehmen. Das Programm `hunt` ist hierbei für Rlogin- und Telnet-Verbindungen optimiert. Der Angriff erfolgt, indem `hunt` sowohl den Client als auch den Server davon überzeugt, dass sie die Pakete an andere MAC-Adressen versenden müssen (ARP-Spoofing). Anschließend senden sowohl der Client als auch der Server weiterhin korrekte

TCP/IP-Pakete. Diese werden jedoch nicht mehr von der Gegenseite gesehen, da sie an falsche und unter Umständen nicht existente MAC-Adressen gerichtet sind. `hunt` sieht jedoch weiterhin diese Pakete und ist in der Lage, die wichtigen Informationen zwischen Client und Server auszutauschen.

2. Nun kann `hunt` weitere Informationen in den Fluss dieser Verbindung injizieren. Da `hunt` die TCP-Pakete sieht, kann `hunt` die erforderlichen Sequenznummern berechnen.
3. Der Server wird den Empfang der zusätzlichen Daten bestätigen. Da er jedoch das ACK-Paket an die korrekte IP-Adresse des Clients sendet, aber die falsche MAC-Adresse nutzt, sieht der Client diese Bestätigung nicht und sendet keine Fehlermeldung an den Server. `hunt` ist weiterhin in der Lage, weitere Daten mit der Vertrauensstellung des Clients an den Server zu senden.
4. Nachdem die Injektion erfolgt ist, bestehen zwei grundsätzliche Möglichkeiten. `hunt` ist in der Lage, die Verbindung mit einem RST-Paket abzubrechen. `hunt` ist aber auch in der Lage, die Verbindung zu resynchronisieren. Hierbei werden einige Daten an den Client und den Server gesendet. Außerdem ist es erforderlich, dass der Benutzer auf dem Client weitere Daten eingibt. Dann kann eine Resynchronisation erfolgreich sein. Der Client hat den Eindruck, dass die Netzwerkverbindung lediglich vorübergehend ausgefallen ist.

Das Werkzeug `hunt` war früher auf der Homepage von Pavel Krauz erhältlich.[2] Leider kann man heute auf diese Seite nicht mehr zugreifen. Sie können das Werkzeug aber alternativ von `http://packetstorm.linuxsecurity.com/sniffers/hunt/` herunterladen.

Die TCP/IP-Protokolle bieten keinen Schutz vor diesen Angriffen. Ein Schutz ist hier nur durch eine zusätzliche Signatur der Pakete, durch höhere Protokolle oder ein VPN möglich. Leider bietet TCP/IP keine Möglichkeit, die Authentizität eines Pakets zu überprüfen. Viele Applikationen wie Telnet führen die Authentifizierung aber nur zu Beginn durch. Anschließend gilt die Verbindung als authentifiziert. Ein Angreifer kann die Verbindung übernehmen und ist dann ebenfalls authentifiziert.

4.4.4 Bufferoverflow

Ein Bufferoverflow ist das Ergebnis eines Programmierfehlers. Nicht jeder Programmierfehler führt zu einem Bufferoverflow, und nicht jeder mögliche Bufferoverflow kann von einem Angreifer ausgenutzt werden. Um den Bufferoverflow zu verstehen, müssen Sie sich in die Rolle eines Programmierers hineinversetzen.

In vielen Programmen kommen einzelne Aufgaben wiederkehrend vor. Diese Aufgaben werden dann gern in einem Unterprogramm realisiert, das von verschiedenen Stellen des Hauptprogramms aufgerufen werden kann. Damit das Unterprogramm später weiß, wohin es im Hauptprogramm zurückspringen muss, sichert der Prozessor vor dem Aufruf des Unterprogramms den aktuellen Stand des Befehlszeigers (Instruction Pointer, IP) auf dem Stapel (Stack). Der Stapel ist eine dynamische Struktur, auf der ein Programm vorübergehend Daten

2 `http://lin.fsid.cvut.cz/~kra/index.html`

ablegen kann. Jedes Programm besitzt einen eigenen Stapel. Ein Stapel erlaubt lediglich das Lesen und Schreiben der Daten auf dem höchsten Punkt. Dieser wird mit dem Stapelzeiger (Stackpointer) referenziert (siehe Abbildung 4.6). Benötigt nun das Unterprogramm vorübergehend Speicherplatz für eine Variable, so fordert es diesen Puffer (Buffer) auf dem Stapel an. Der Stapelzeiger wird um die entsprechende Anzahl Bytes verschoben und als Referenz an das Unterprogramm zurückgegeben (siehe Abbildung 4.7). Das bedeutet, das Unterprogramm kann nun Daten auf dem Stapel ablegen, wobei es beim Stapelzeiger beginnt und dann rückwärts geht.

Stellen wir uns vor, das Programm erwartet die Eingabe des Geburtsdatums des Benutzers. So genügen sicherlich 32 Bytes, um diese Eingabe entgegenzunehmen. Diese 32 Bytes werden nun auf dem Stapel reserviert. Aus irgendeinem Grund (Unwissenheit, Bösartigkeit) gibt der Benutzer jedoch 100 Zeichen ein. Überprüft der Programmierer vor der Kopie der Zeichenkette in den dynamisch allozierten Datenbereich auf dem Stapel ihre Länge nicht, so

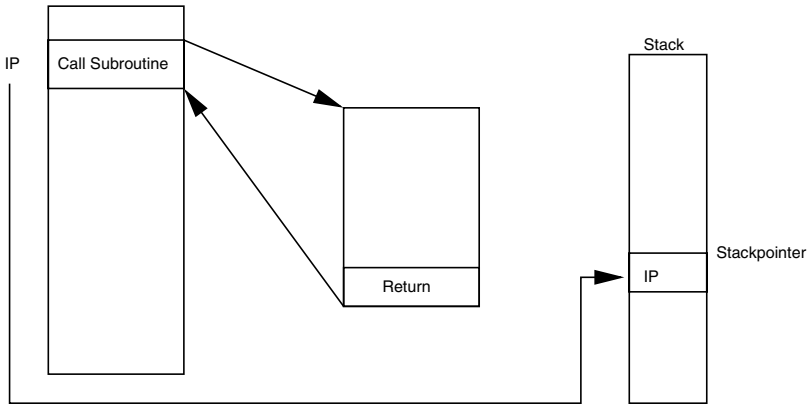


Abbildung 4.6: **Funktionsweise eines Bufferoverflows**

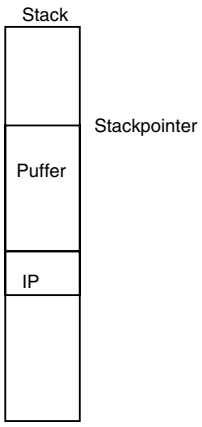


Abbildung 4.7: **Pufferreservierung auf dem Stapel**

werden alle 100 Bytes auf den Stapel kopiert. Dadurch werden auch Bereiche überschrieben, die andere gültige Daten enthielten. Es kommt zu einem Überlaufen (Overflow) des originalen Puffers (siehe Abbildung 4.8). Hierbei können auch zum Beispiel Rücksprungadressen der Unterprogramme überschrieben werden.

Es gibt einige Programmiersprachen, die dem Programmierer diese Arbeit (das sogenannte Boundary Checking) abnehmen, jedoch gehören die Programmiersprachen Assembler und C nicht dazu. In C werden die meisten Anwendungen für UNIX- und auch für Windows-Betriebssysteme programmiert.

Handelt es sich um willkürliche Daten, so stürzt das Programm ab, und es kommt zum Segmentation Fault (Linux) oder zu einer allgemeinen Schutzverletzung (General Protection Fault, Windows). Der Prozessor erkennt, dass es sich um eine unerlaubte Rücksprungadresse handelt. Ein Zugriff auf den Speicher an der Zieladresse ist dem Programm nicht erlaubt. Das Programm wird von dem Prozessor beendet (es stürzt ab) und steht nicht mehr zur Verfügung. Dies ist ein Denial-of-Service.

Unter Umständen besteht jedoch auch die Möglichkeit, den Ort einer Rücksprungadresse auf dem Stack vorherzusagen und so zu überschreiben, dass ein gezielter Rücksprung auf Code erfolgt, der sich im Vorfeld in den eingegebenen Daten befand. Dann wird in dem Benutzerkontext des missbrauchten Programms der gerade eingegebene Code ausgeführt. Üblicherweise handelt es sich hierbei um den Aufruf einer UNIX-Shell. Daher bezeichnet man diesen von dem Einbrecher verwendeten Code auch häufig als *Shell-Code*. Dies ist unter UNIX besonders brisant, da viele Netzwerkdienste über Rootprivilegien verfügen und Eingaben aus ungewisser Quelle entgegennehmen. Weisen diese Dienste derartige Mängel auf, so können sie ausgenutzt werden, um Rootprivilegien auf dem entsprechenden Rechner zu erlangen. In diesem Fall spricht man von *Root Exploits*.

Viele der ersten geschriebenen Bufferoverflows sind leicht an dem sogenannten NOP Sled zu erkennen. Ein gezielter Rücksprung, wie gerade beschrieben, ist meist nicht möglich. Der Angreifer kann nur selten genau den Zustand des Stacks vorhersagen. Daher kann er auch

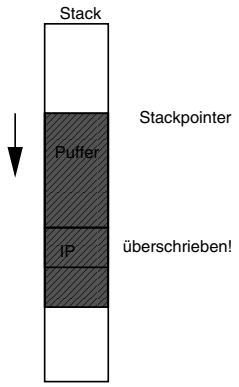


Abbildung 4.8: **Überlaufen des Puffers**

nicht die Rücksprungadresse berechnen. Nun versieht er seinen Code zu Beginn mit bis zu mehreren Hundert NOP-Befehlen. Ein NOP ist ein No-Operation-Befehl. Dieser Befehl hat, wenn er von dem Prozessor ausgeführt wird, keine Funktion, außer den Instruction-Pointer weiterzubewegen. Ist der Code für den Bufferoverflow derartig angepasst, so muss die Rücksprungadresse nur noch ungefähr in den Bereich der NOP-Befehle zeigen. Dies bezeichnet man als NOP-Schlitten (Sled), da der Befehlszeiger wie auf einem Schlitten über die NOPs zum Bufferoverflow rutscht und schließlich diesen Code ausführt.

Der injizierte Code wird wie gesagt als Shell-Code bezeichnet. Dieser Name leitet sich aus der Tatsache ab, dass klassische Bufferoverflows versuchen, eine Shell zu starten und so Kommandozeilenzugriff auf dem angegriffenen System zu erhalten.

Es gibt verschiedene Möglichkeiten, sich vor Bufferoverflows zu schützen. Der älteste Schutz erreicht dies, indem der gesamte Stapel für das Programm als nicht ausführbar gekennzeichnet wird.

STOP

Machen Sie nicht den Fehler anzunehmen, dass Sie mit dieser Methode aus dem Schneider sind. Erstens stürzt das Programm weiterhin ab. Der Prozessor erkennt lediglich, dass unerlaubter Code angesprungen werden soll. Außerdem gibt es auch Heap-Overflows. Hier hilft die Methode nicht.

Bei älteren Prozessoren der i386-Architektur ist dies CE[m] nur mit Tricks möglich. Aktuelle Prozessoren unterstützen diese Funktion in Hardware. Bei dem Athlon64 existiert das No-eXecute-Bit (NX), das von AMD als Enhanced-Virus-Protection-Technologie vermarktet wird. Intel hat mit den Pentium 4- und M-Modellen das eXecute-Disable-Bit (XD) eingeführt. PowerPC, SPARC und Alpha-Prozessoren verfügen über diese Funktion schon länger.

STOP

Bei dem NX- und XD-Bit ist es wichtig zu wissen, dass diese Bits nicht automatisch aktiviert werden. Sie müssen ein Betriebssystem einsetzen, das diese Bits auch nutzt (zum Beispiel RHEL und Fedora mit ExecShield).

Es existieren noch drei weitere Varianten, um sich vor den Gefahren des Bufferoverflows zu schützen. Der modifizierte GNU-C-Compiler Stackguard implementierte als Erster den Schutz mit dem sogenannten Canary (Kanarienvogel). Hierbei erzeugt der Compiler modifizierten Code, der vor jedem Sprung in ein Unterprogramm eine Zufallszahl auf dem Stapel abspeichert. Vor jedem Rücksprung überprüft das Programm, ob die Zufallszahl modifiziert wurde. Ist dies der Fall, wird ein Bufferoverflow angenommen und die Ausführung abgebrochen. Ansonsten fährt das Programm fort. Diese Vorgehensweise ist von IBM in ProPolice weiterentwickelt worden. ProPolice gibt es für einige Linux-Distributionen und OpenBSD.

Red Hat hat mit den Position-Independent-Executables (PIE) einen Schritt in eine andere Richtung gemacht. Auch hier wurde der GNU-C-Compiler so modifiziert, dass der entstehende Programmcode beliebig im Speicher arrangiert werden kann. Ein modifizierter Kernel (ExecShield-Patch) kann nun bei jedem Aufruf das Programm und alle Daten inklusive des Stapels zufällig an einer anderen Position abspeichern. Der Angreifer hat nun das Problem, dass er die Rücksprungadressen nicht mehr vorhersagen kann. Stellen Sie sich vor, Sie wären in einem Raum mit fünfhundert Türen. Eine ist offen. Sie haben nur einen Versuch. Es ist ein-

CE[m] Fehlt hier der Bezug...?

Editor's or typesetter's annotations (will be removed before the final TeX run)

fach, die richtige Tür zu finden, wenn sie immer an derselben Stelle ist und Sie dies vorher in einem anderen Raum üben können. Wenn die richtige Tür aber jedes Mal eine andere ist, ist dies vielfach schwerer.

STOP

Eine Firewall kann Sie nicht vor einem Bufferoverflow schützen.

4.4.5 Formatstring-Angriffe

Bei den Formatstring-Angriffen handelt es sich um eine noch rechte junge Gruppe von Angriffen. Der Formatstring-Angriff wurde zum ersten Mal im Juni 2000 für einen Angriff auf den WU-Ftpd-Server verwendet. Ähnlich wie der Bufferoverflow betrifft dieses Problem Programme, die in C geschrieben wurden. Die Programmiersprache C kennt verschiedene Funktionen, die formatierte Textausgaben erlauben: `printf`, `scanf` etc. Diese Funktionen nehmen einen Formatstring und mehrere Werte entgegen und geben die Werte formatiert aus, z. B. so:

```
printf("%s", buffer)
```

Leider geben nicht alle Programmierer immer den Formatstring an, sondern sparen sich die Mühe und schreiben `printf(buffer)`. Wenn nun der Inhalt des Puffers von dem Angreifer eingegeben wurde, kann er auch Formatstrings angeben. Die Formatstrings und der Puffer werden immer über den Stapel an die Funktion übergeben. Die `printf`-Funktion interpretiert dann die Formatstrings des Angreifers. Es gibt Formatstrings, mit denen der Angreifer sowohl Daten von dem Stapel lesen kann (zum Beispiel `%s` und `%x`) als auch Werte schreiben kann (`%n`). So kann der Angreifer zunächst den Ort der richtigen Rücksprungadresse eines Unterprogramms ermitteln und anschließend überschreiben. Die Auswirkung ähnelt also einem Bufferoverflow. Ein Formatstring-Angriff kann also auch verwendet werden, um die Kontrolle über einen Dienst oder ein System zu erlangen. Dabei besteht mit dem Formatstring `%n` die Möglichkeit, an beliebigen Adressen den Inhalt zu modifizieren. Der so von dem Angreifer injizierte Code muss also nicht auf dem Stapel liegen. Die Schutzmechanismen, die einen Überlauf erkennen (Stackguard, ProPolice) oder den Stapel als nicht ausführbar deklarieren, können uns daher nicht vor Formatstring-Angriffen wirksam schützen. Lediglich die Randomisierung der Adressen (ExecShield mit PIE, PaX etc.) bietet hier einen gewissen Schutz.

STOP

Eine Firewall bietet keinen Schutz vor Formatstring-Angriffen!

4.4.6 Race-Condition

Eine Race-Condition entsteht, wenn zwei Prozesse gleichzeitig auf eine Ressource zugreifen, dieser Zugriff nicht geordnet erfolgt und der Ausgang von der Reihenfolge des Zugriffs abhängig ist. Diesen langen komplizierten Satz möchte ich anhand eines einfachen Beispiels erläutern.

Ok?

Stellen Sie sich vor, Sie haben bei einer Bank ein Konto. Das Konto weist ein Guthaben von 500,00 EUR auf. Sie tätigen eine Überweisung und spenden 250,00 EUR für die Free-Software-Foundation-Europe (FSFE). Gleichzeitig geht Ihr Gehalt von Ihrem Arbeitgeber ein (2.500,00 EUR). Wenn diese beiden Vorgänge nun nicht geordnet durchgeführt werden, kann Folgendes passieren:

1. Für die Spendenüberweisung wird das Guthaben gelesen. Guthaben = 500,00 EUR.
2. Für die Gehaltsüberweisung wird das Guthaben gelesen. Guthaben = 500,00 EUR.
3. Das Gehalt wird addiert, und die Summe wird als neues Guthaben geführt. Guthaben = 500,00 EUR + 2.500,00 EUR = 3.000 EUR. Diese Summe wird als Guthaben gespeichert.
4. Für die Überweisung wird von dem Guthaben (500,00 EUR, es wurde ja früher gelesen) 250,00 EUR abgezogen. Guthaben = 500,00 EUR - 250,00 EUR = 250,00 EUR. Diese Summe wird nun als neues Guthaben gespeichert. Die 2.500,00 EUR sind verloren.

Natürlich hätte es auch genau andersherum passieren können. Dann hätten Sie trotz der Überweisung von 250,00 EUR an die FSFE anschließend immer noch 3.000,00 EUR besessen.

Eine Race-Condition führt also zu inkonsistenten Daten. Teilweise kann ein Angreifer diese aber auch ausnutzen, um erweiterte Rechte zu erlangen. Meist erfordert das Ausnutzen einer Race-Condition bereits Zugang zu einem System. Race-Conditions, die aus der Ferne über Netzwerkdienste ausgenutzt werden können, sind sehr selten.

Im Folgenden möchte ich Ihnen zwei typische Race-Conditions vorstellen: `/tmp`-Race-Conditions und die `ptrace`-Race-Condition.

Race-Conditions bei der Behandlung von temporären Dateien

Häufig treten bei der Behandlung von temporären Dateien Race-Conditions auf. Dies hängt damit zusammen, dass bei einem mehrfachen Zugriff auf temporäre Dateien über den Dateinamen das Programm nicht garantieren kann, dass es sich immer um dieselbe Datei handelt.

Stellen Sie sich vor, dass Sie ein Skript entwickeln möchten, das bei allen Systemkonten die Shell auf `/bin/false` ändern sollte. Ein Systemkonto benötigt keine Möglichkeit der interaktiven Anmeldung auf einem Linux-System. Ihr Skript könnte folgendermaßen aussehen:

Listing 4.1: **Ein per Race-Condition verwundbares Skript**

```
#!/bin/sh
TEMP=/tmp/mytemp
rm -f $TEMP
cat /etc/passwd | while read zeile
do
   echo $zeile | awk -F':' \
         '{if (($3>0) && ($3<500)) print $1":"$2":"$3":
                                   "$4":"$5":"$6":/bin/false";
           else print $0}'
   done >> $TEMP
mv $TEMP /etc/passwd chmod 644 /etc/passwd
```

Dieses Skript löscht zunächst die temporäre Datei, da es anschließend eine Zeile an die neue Datei anhängen möchte (`done » $TEMP`). Dann liest es die Datei `/etc/passwd` und ersetzt bei allen Konten mit den Nummern 1–499 die Shell in der Spalte 7 durch `/bin/false`. Anschließend benennt das Skript die Datei in `/etc/passwd` um und setzt die Rechte entsprechend.

Wo ist hier die Race-Condition? Stellen Sie sich vor, der Angreifer wüsste, dass Sie dieses Skript als root starten. Er könnte versuchen, ein eigenes Skript gleichzeitig laufen zu lassen.

Listing 4.2: **Der Exploit**

```
#!/bin/sh
TEMP=/tmp/mytemp
touch $TEMP
while test -f $TEMP
    do NOP=0 # do nothing
done echo "toor::0:0:toor owns your machine:/:/bin/sh" > $TEMP
```

Dieses Skript legt die temporäre Datei mit dem bekannten Namen an. Anschließend prüft das Skript, ob die Datei noch existiert oder bereits von dem verwundbaren Skript gelöscht wurde. Sobald das erkannt wurde, bricht die Schleife ab, und das Skript schreibt die angegebene Zeile in die Datei `/tmp/mytemp`, bevor das verwundbare Programm seine Daten an diese Datei anhängt.

INFO

Wenn Sie dieses Problem nachstellen möchten, dann sollten Sie, damit es nicht von Ihrem Glück abhängt, in dem verwundbaren Programm die Zeitspanne des möglichen Exploits vergrößern. Fügen Sie einfach ein `sleep 1` nach dem `rm`-Befehl ein.

Um sich vor diesen Race-Conditions bei der Verwendung von temporären Dateien zu schützen, ist es wichtig, dass der Name der temporären Datei möglichst nicht vorhersagbar ist. Viele Programme hängen daher ihre PID (Prozess-ID) an den Dateinamen an. Jedoch ist auch diese PID auf vielen Linux/UNIX-Systemen in gewissen Bereichen vorhersagbar. Die beste Lösung ist die Verwendung von benutzerspezifischen temporären Verzeichnissen, in denen

andere Benutzer keine Dateien erzeugen können, oder der Befehl `mktemp`, der eine temporäre Datei nach dem Muster `/tmp/tmp.XXXXXXXXXX` erzeugt. Hiermit sind 26^{10} verschiedene Dateinamen möglich. Eine Vorhersage durch den Angreifer ist unmöglich.

Bei der Beschreibung des Angriffs sollte Ihnen deutlich geworden sein, dass eine Firewall keinerlei Schutz bietet. Hier kann aber ein MAC-System wie SELinux durchaus Abhilfe schaffen.

Die ptrace-Race-Condition

Bei der `ptrace`-Race-Condition handelt es sich um eine Race-Condition im Linux-Kernel, die im Januar 2003 entdeckt wurde. Diese Sicherheitslücke war nur lokal ausnutzbar. Dennoch wurde sie bei vielen Angriffen genutzt, denn es gab gleichzeitig auch Sicherheitslücken in weiteren Netzwerkdiensten wie dem Apache Webserver. Die Angreifer konnten über den Netzwerkdienst Kommandozeilenzugriff auf das Zielsystem erhalten. Da die meisten Netzwerkdienste aber nicht über root-Privilegien verfügen, musste der Angreifer anschließend nach einer weiteren Sicherheitslücke suchen, um eine Privilege-Escalation, eine Erweiterung seiner Privilegien, durchzuführen. Dafür wurde dann die `ptrace`-Race-Condition genutzt. Linux-Kernel <2.2.25/2.4.20 weisen diese Sicherheitslücke auf.

Der Fehler tritt auf, wenn ein Prozess eine Funktion nutzen möchte, die über ein noch nicht geladenes Kernelmodul realisiert wird. Dies ist zum Beispiel häufig bei dem Öffnen eines Netzwerk-Sockets für eine untypische Protokoll-Familie (PF_IPX, PF_X25, PF_AX25, PF_APPLETALK etc.) der Fall. Um die Protokoll-Familie zu unterstützen, muss der Kernel ein Modul nachladen. Hierzu erzeugt der Kernel automatisch von dem anfordernden Prozess einen Kindprozess. Anschließend stellt der Kernel die User-ID des Kindprozesses auf 0 und lädt mit dem Befehl `modprobe` das Modul. Da der Kindprozess in dem Kontext des ursprünglichen Benutzers gestartet wird, kann dieser auf den Prozess Einfluss nehmen, bevor der Kernel die root-Privilegien überträgt. Hier ist die Race-Condition.

Wie nutzt man dies nun aus? Der Kernel bietet mit der `ptrace`-Funktion die Möglichkeit, Prozesse zu beobachten, Breakpoints einzufügen und den Code zu modifizieren. Diese Funktion wird von Debuggern zur Fehlersuche verwendet. Kann sich der Angreifer mit dieser Funktion an den Kindprozess binden, bevor der Kernel die root-Privilegien überträgt, so kann er anschließend einen Prozess mit root-Privilegien steuern, den Code modifizieren und so jede beliebige Tätigkeit auf dem System ausüben. Er hat die komplette Kontrolle über das System übernommen.

Ich hoffe, Sie haben erkannt, dass die Race-Condition weniger ein technischer Programmierfehler als vielmehr ein Fehler im logischen Design einer Applikation ist. Häufig sind es Seiteneffekte, die eine Race-Condition ermöglichen. Eine Verteidigung mit einer Firewall ist nicht möglich. Hier helfen nur Mandatory-Access-Control-Systeme wie SELinux oder AppArmor. Wenn diese Systeme auch die Race-Condition selbst nicht verhindern können, wenden sie jedoch weiteren Schaden von dem System ab.

4.4.7 SQL-Injektion

Die SQL-Injektion wird hier als ein Paradebeispiel eines Angriffs über einen Webserver beschrieben. Erwartungsgemäß erfolgen heute die meisten Angriffe über das Netzwerk entweder über das HTTP/HTTPS- oder das SMTP-Protokoll. Dies sind in vielen Fällen noch die einzigen Wege in das Netzwerk eines Unternehmens. Jeder andere Zugriff wird heute meist von Firewalls unterbunden. Mögliche Fernzugriffe werden über sichere VPN-Lösungen implementiert. Der Angreifer kann – vorausgesetzt, Sie haben Ihre Hausaufgaben als Firewall-Administrator gemacht – nur auf den Webserver und den Mailserver des Unternehmens zugreifen. Allerdings bestehen die meisten Websites heute aus dynamischen Seiten, die von einer Web-Applikation erzeugt werden. Diese Web-Applikation greift hierzu auf eine Datenbank im Hintergrund zu. Über diesen Zugriff erhält die Web-Applikation eines Internet-Shops zum Beispiel Informationen über den Preis und die Verfügbarkeit eines Artikels. Diese Datenbank befindet sich meist in dem internen Netz des Unternehmens (siehe Abbildung 4.9). Der Webserver, der sich in der DMZ befindet, benötigt daher Zugriff auf diese Datenbank. Ein Angreifer, der diese Logik ausnutzen kann, erhält so unter Umständen Zugriff auf interne Systeme!

Bei der SQL-Injektion versucht der Angreifer herauszufinden, wie eine Web-Applikation die von ihm eingegebenen Daten verwendet. Wenn die Applikation zum Beispiel über eine Login-Seite verfügt (siehe Abbildung 4.10), dann besteht die Möglichkeit, dass die Applikation zur Überprüfung der Daten folgende Anfrage an die Datenbank stellt:

```
SELECT name FROM users WHERE name="$name" AND password="$pass"
```

Wenn der Angreifer als Namen `Bob` und als Kennwort `password` eingibt, wird der folgende SQL-Befehl ausgeführt:

```
SELECT name FROM users WHERE name="Bob" AND password="password"
```

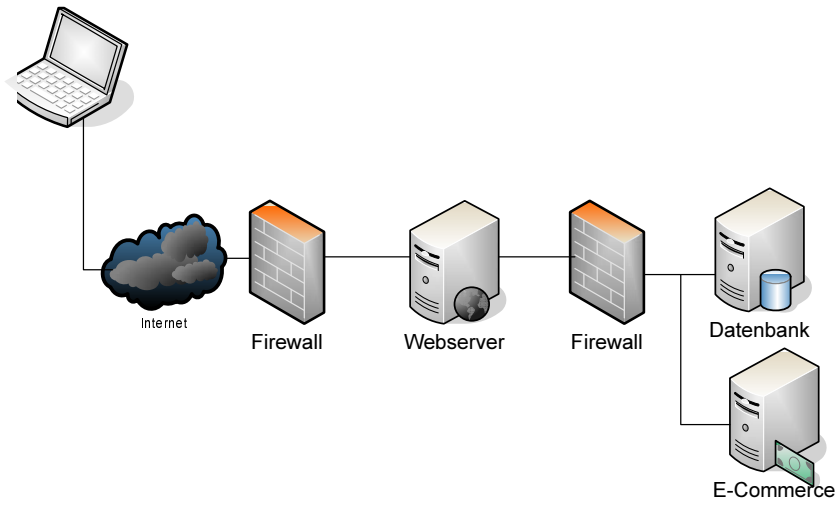


Abbildung 4.9: **Die Web-Applikation auf einem Webserver greift meist durch die Firewall auf die interne Datenbank zu.**

Abbildung 4.10: **Viele Applikationen sind durch eine Login-Seite geschützt.**

Existiert der Benutzer und ist das Kennwort korrekt, so liefert diese Anweisung den Namen des angemeldeten Benutzers. Vielleicht kennt der Angreifer den Namen, aber nicht das Kennwort eines Benutzers. Dann kann er folgende Eingabe versuchen:

```
name=Bob pass=weissnicht" OR 1=1
```

Wird dies nun in der SQL-Anweisung eingesetzt, ergibt sich die folgende Anweisung:

```
SELECT name FROM users WHERE name="Bob" AND password="weissnicht" OR 1=1
```

Da der Ausdruck 1=1 immer stimmt, ist die Anmeldung unabhängig von dem eingegebenen Kennwort gültig, da die Select-Anweisung ein Ergebnis liefert.

Diese Sicherheitslücke ist möglich, da der Programmierer die eingegebenen Daten nicht vor der Verwendung prüft. Es handelt sich um eine fehlende Eingabevalidierung ähnlich dem Bufferoverflow oder dem Formatstring-Angriff. Die Web-Applikation muss sämtliche Eingaben auf ihre Gültigkeit prüfen, bevor diese an die Datenbank weitergereicht werden. In diesem Beispiel wäre es zum Beispiel sinnvoll, bei dem Benutzernamen zu prüfen, ob außer den Zeichen A–Z, a–z weitere nicht erlaubte Zeichen in dem Namen oder außer A–Z, a–z, 0–9 weitere Zeichen in dem Kennwort vorkommen. Diese Aufgabe kann jedoch nur von der Web-Applikation selbst erledigt werden. Nur der Programmierer der Web-Applikation kann die gültigen Zeichen einer Eingabe festlegen. Eine klassische Firewall ist hier überfordert. Es gibt jedoch die Web-Application-Firewall (WAF). Produkte dieser Gattung, wie zum Beispiel Modsecurity[3], können derartige Angriffe erkennen und abwehren. Alternativ gibt es auch bereits Firewallprodukte für Datenbanken. Diese analysieren den SQL-Befehlsstrom und verbieten gefährliche Befehle. Ein Open-Source-Vertreter ist GreenSQL[4].

3 http://www.modsecurity.org
4 http://www.greensql.net

4.4.8 Welchen Schutz bietet eine Firewall?

Ich habe Ihnen auf den letzten Seiten verschiedene Bedrohungsszenarien bei der Anbindung Ihrer Rechner an das Internet vorgestellt. Bei fast allen Angriffen habe ich Sie darauf hingewiesen, dass sie von einer Firewall nur in geringem Maße oder gar nicht abgewehrt werden können.

Warum sollen Sie dann überhaupt den Aufwand einer Firewall betreiben?

Weil ohne Firewall alles noch schlimmer wird! Überlegen Sie sich, welche Dienste Sie in Ihrem Netz und auf Ihren Rechnern betreiben. Sicherlich möchten Sie nicht, dass jeder im Internet auf diese Dienste zugreifen kann. Wahrscheinlich ist es auch nicht in Ihrem Sinne, dass jeder Benutzer auf jeden Dienst im Internet zugreifen kann. Hier bietet die Firewall Schutz. Natürlich erkennt die Firewall keinen Bufferoverflow- oder Formatstring-Angriff. Ohne Firewall kann ein Angreifer mit diesen Methoden alle Ihre Systeme angreifen. Wenn Sie eine Firewall besitzen, ist dieser Angriff nur auf den Systemen möglich, auf die die Firewall den Zugriff erlaubt.

Eine Firewall bietet Schutz nach dem Prinzip „Single-Point-of-Defense“. Anstatt jeden Rechner einzeln zu verteidigen und die Dienste in allen Punkten auf Sicherheitslücken zu prüfen, regelt die Firewall, wer auf welche Dienste zugreifen darf. Natürlich sind diese Dienste dann immer noch besonders gefährdet. Diese Dienste können immer noch mit einem Bufferoverflow-Angriff getroffen werden. Die tatsächliche Zahl der verwundbaren Systeme und Dienste haben Sie aber durch den Einsatz einer Firewall reduziert.

Eine Firewall hilft auch bei der Schadensbegrenzung nach einem erfolgreichen Angriff, denn Angriffe erfolgen nur selten blind. Der Angreifer will nach dem Bufferoverflow auf der Kommandozeile des Zielsystems arbeiten, seine Privilegien erweitern und weiteren Schaden anrichten. Hierfür benötigt er Netzwerkverbindungen, Werkzeuge und die Möglichkeit, weitere Systeme zu kontaktieren.

Betrachten Sie noch einmal die Abbildung 4.9. Wenn die Firewall Verbindungen des Webservers in das Internet unterbindet, kann ein Angreifer keine weiteren Werkzeuge nachladen. Auch der Start einer Remote-Shell, die eine neue Netzwerkverbindung aufbaut, ist nicht ohne Weiteres möglich. Erlaubt die Firewall zusätzlich nur den Zugriff auf den Port des Datenbankservers in dem internen Netz, sind die weiteren Rechner vor dem Zugriff des Angreifers geschützt. Der Angreifer muss jetzt erst eine Sicherheitslücke in dem Datenbankserver finden, um diesen angreifen zu können.

Daher ist eine Firewall sehr wohl sinnvoll. Die Firewall schützt aber nicht vor allen Gefahren und Bedrohungen aus dem Internet. Sie müssen diese Bedrohungen kennen, um zusätzliche Maßnahmen zu implementieren, die Systeme regelmäßig zu patchen und so Sicherheit aufrechtzuerhalten.

Teil II

Firewalls mit Iptables – Einführung

5. Eine einfache Firewall mit Iptables

Dieses Kapitel spricht in erster Linie den Linux-Benutzer an, der sich noch nie ernsthaft mit den Linux-Paketfilterfunktionen beschäftigt hat. Es beginnt direkt mit praktischen Beispielen, aber erklärt auch sämtliche nötigen Grundlagen, ohne zu tief zu gehen. Nach diesem Kapitel können Sie bereits eine einfache Linux-Paketfilter-Firewall entwickeln und einsetzen. Spätere Kapitel beschreiben dann weitergehende Funktionen.

5.1 Netfilter und Iptables

Wenn Sie sich zum ersten Mal mit den Linux-Paketfilterfunktionen beschäftigen, werden Sie zwangsläufig auf die Begriffe Netfilter und Iptables treffen. Viele Anwender kennen den Unterschied nicht und nutzen diese Begriffe daher als Synonym. In Wirklichkeit sollten Sie diese Begriffe unterscheiden, daher beginnen wir zunächst mit einer Begriffsdefinition:

» **Netfilter**: Netfilter bietet Ankerpunkte (Hooks) im Netzwerkstapel von Linux an. Hier können sich dann weitere Kernelmodule anmelden. Sobald ein Paket an diesem Ankerpunkt vorbeikommt, wird das entsprechende Modul aufgerufen. Es handelt sich also um eine ganz allgemeine Struktur, die Filterfunktionen anbietet.

» **Iptables**: `iptables` erlaubt die Definition von Regeln in Tabellen (tables). Jede Tabelle besteht aus mehreren Ketten. Diese Ketten werden von Iptables an die verschiedenen Hooks von Netfilter gebunden. Es nutzt also die von Netfilter zur Verfügung gestellte Struktur.

Für die gesamte Funktionalität des Linux-Paketfilters ist zusätzlich auch noch das Connection-Tracking- und das NAT-Subsystem erforderlich (s. u.). Als Anwender werden Sie sich nie mit Netfilter beschäftigen müssen. Alle für Sie interessanten Funktionen werden Sie mit dem `iptables`-Befehl konfigurieren können.

5.2 Der Iptables-Befehl und die Filter-Tabelle

Wenn Sie bereits einige Erfahrungen mit dem Einsatz des `iptables`-Befehls gemacht haben, können Sie diesen Abschnitt überspringen. Vielleicht lesen Sie ihn aber doch. Es könnte sich der eine oder andere interessante Hinweis finden lassen.

Iptables verwaltet seine Regeln in mindestens drei Tabellen: `filter`, `mangle` und `nat`. Hier wird zunächst nur die `filter`-Tabelle besprochen. Die anderen Tabellen werden in weiteren Abschnitten (5.7, 5.9) erörtert.

Die `filter`-Tabelle ist für die Filterung der Pakete verantwortlich. Das heißt, hier entscheidet der Linux-Kernel, ob ein Paket erlaubt, abgelehnt oder verworfen wird. Da dies die wesentliche Funktion einer Firewall darstellt, werden wir uns zunächst hiermit beschäftigen. In der `filter`-Tabelle unterscheidet der Linux-Kernel drei Ketten: `INPUT`, `OUTPUT` und `FORWARD` (siehe Abbildung 5.1). Diese drei Ketten haben genau definierte Aufgaben. Alle Pakete, die an den Rechner selbst gerichtet sind, durchlaufen die `INPUT`-Kette. Alle Pakete, die der Rechner selbst erzeugt, werden vor dem Verlassen des Rechners durch die `OUTPUT`-Kette gefiltert. Die `FORWARD`-Kette kümmert sich schließlich um die Pakete, die der Rechner lediglich weiterleiten soll. Hierbei handelt es sich also um Pakete, die an die Firewall als Gateway zur Weiterleitung geschickt werden.

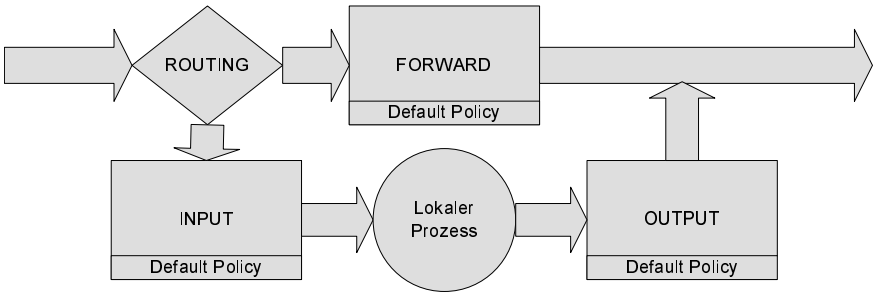


Abbildung 5.1: **Die Filtertabelle besteht aus drei Ketten.**

STOP

Diese Aufgabenteilung unter den Ketten ist bei Iptables neu. Wenn Sie sich bereits in grauer Vorzeit mit dem Paketfilter `ipchains` beschäftigt haben, kennen Sie dort noch eine andere Nutzung dieser Ketten.

Jede dieser Ketten besteht nun aus einer Liste von Regeln, die für jedes Paket nacheinander abgearbeitet werden (Abbildung 5.2). Sobald eine Regel auf das Paket zutrifft, wird die mit der Regel verbundene Aktion ausgeführt und mit wenigen Ausnahmen die Abarbeitung weiterer Regeln abgebrochen. Trifft keine Regel aus der Liste zu, so wird die Default Policy der Kette aktiv. Diese Default-Regel trifft auf alle Pakete zu, die nach Abarbeitung aller Regeln noch in der Kette verblieben sind.

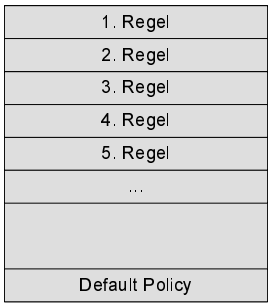


Abbildung 5.2: **Die Regeln in einer Kette werden der Reihe nach abgearbeitet. Trifft keine Regel auf das Paket zu, gilt die Default-Policy.**

Da diese Einführung bisher recht theoretisch und abstrakt war, macht es Sinn, dies gleich in die Praxis umzusetzen. Im Folgenden setze ich voraus, dass Sie über einen Rechner mit einer aktuellen Linux-Distribution Ihrer Wahl verfügen und als root angemeldet sind.

Prüfen Sie bitte zunächst Ihre Installation, indem Sie folgenden Befehl eingeben:

```
[root@bibo root]# iptables -V
iptables v1.4.5
```

INFO

Wundern Sie sich bitte nicht über die Namen meiner Rechner. Alle meine Systeme haben Namen aus der Sesamstraße. Neben Kermit, Oskar, Krümelmonster und Grobi gibt es auch einen Bibo.

Wenn der Befehl auf Ihrem System die Fehlermeldung „-bash: iptables: command not found" ausgibt, dann prüfen Sie bitte, ob Sie das entsprechende Paket noch installieren müssen. Gibt der Befehl so wie oben eine Versionsnummer aus, können Sie direkt fortfahren. Die Version selbst ist im Moment noch unerheblich.

Als Erstes sollten Sie sich eventuell bereits vorhandene Regeln anzeigen lassen:

```
[root@bibo root]# iptables -L
Chain INPUT (policy ACCEPT)
target              prot opt source               destination

Chain FORWARD (policy ACCEPT)
target              prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target              prot opt source               destination
```

Werden bei dieser Ausgabe bereits Regeln angezeigt, sollten Sie für die Dauer dieses Kapitels die Regeln deaktivieren. Wenn Sie diese Regeln nicht selbst gesetzt haben, ist hierfür wahrscheinlich Ihre Distribution verantwortlich. Auf einer Red Hat Linux-basierten Distribution (auch Fedora) können Sie mit folgendem Befehl die eingebaute Firewall abschalten:

```
[root@bibo root]# /etc/init.d/iptables stop
Firewall-Regeln löschen: [ OK ]
Setze Chains auf Policy ACCEPT: filter [ OK ]
iptables-Module beenden: [ OK ]
```

Auf einer SuSE-Distribution geben Sie bitte folgenden Befehl ein:

```
rcSuSEfirwall2 stop
```

Auf einem Gentoo-System sollte schließlich dieser Befehl die Firewall abschalten:

```
/etc/init.d/firewall stop
```

Es ist sicherlich nicht möglich, hier alle Eventualitäten und Distributionen zu berücksichtigen. Wenn die oben aufgeführten Befehle auf Ihren Systemen nicht funktionieren, lesen Sie bitte in der Dokumentation Ihrer Distribution nach.

Nun können Sie erste Regeln erzeugen. Dies erfolgt zunächst direkt auf der Kommandozeile mit dem Befehl `iptables`. Um direkt einen Effekt erkennen zu können, sollten Sie der Einfachheit halber auf einer grafischen Oberfläche arbeiten und dort zwei Fenster öffnen. In dem einen Fenster rufen Sie den `ping`-Befehl auf (siehe Abbildung 5.3):

```
ping 127.0.0.1
```

In dem anderen Fenster arbeiten Sie mit dem `iptables`-Befehl.

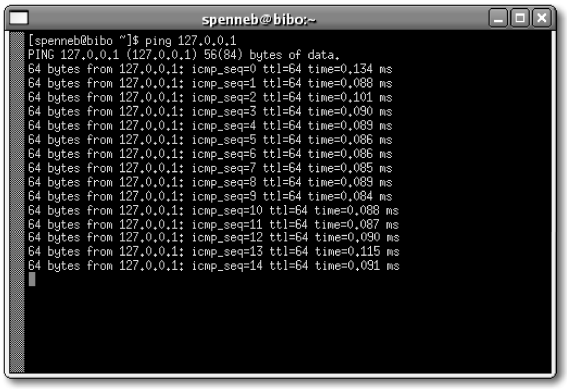


Abbildung 5.3: **Der Ping-Befehl kann die Erreichbarkeit eines Rechners prüfen.**

Beobachten Sie die Ausgabe des Ping, wenn Sie den folgenden Befehl absetzen:

```
[root@bibo root]# iptables -A INPUT -p icmp -j DROP
```

Der Ping sollte nun unbeantwortet bleiben. Dieser Befehl hängt eine Regel an die INPUT-Kette an (`-A`; `--append`). Diese Regel prüft in der INPUT-Kette, ob ein Paket das Protokoll ICMP (`-p`; `--protocol`) verwendet. Dieses Protokoll wird zum Beispiel von dem Ping genutzt. Ist das der Fall, wird das Paket verworfen (`-j`; `--jump`, `DROP`). Alle auf dem Rechner ankommenden Ping-Pakete werden also nun von der eingebauten Firewall verworfen. Alle weiteren Pakete können die Firewall weiterhin passieren.

Ein DROP verwirft die betroffenen Pakete, ohne den Absender davon zu unterrichten. Der Absender erhält keine Antwort und auch keine Fehlermeldung. Er wird es je nach Protokoll und Anwendung ein weiteres Mal versuchen. Nach mehreren Fehlversuchen wird der Absender üblicherweise den Verbindungaufbau mit einem Timeout abbrechen. Möchten Sie dem Absender direkt eine Fehlermeldung senden, um diesen Vorgang abzukürzen, können Sie das Ziel `REJECT` nutzen. Das `REJECT`-Ziel verwirft das Paket und sendet eine Fehlermeldung an den Absender, sodass dieser es nicht erneut versucht.

```
[root@bibo root]# iptables -A INPUT -p icmp -j REJECT
```

Sie können sich die Regeln mit `iptables -L` wieder ansehen. Mehr Informationen erhalten Sie aber mit folgendem Befehl:

```
[root@bibo root]# iptables -vnL --line-numbers
Chain INPUT (policy ACCEPT 142K packets, 209M bytes)

 num pkts bytes target prot opt in out source    destination
 1   2    168   DROP   icmp --  *  *   0.0.0.0/0 0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num pkts bytes target prot opt in out source destination

Chain OUTPUT (policy ACCEPT 112K packets, 6371K bytes)
num pkts bytes target prot opt in out source destination
```

Hier sehen Sie die ausführliche (`-v`; `--verbose`) Liste der Regeln (`-L`; `--list`) in numerischer Form (`-n`; `--numeric`) mit Zeilennummern (`--line-numbers`). Zunächst sehen Sie wieder jede Kette mit ihrer Default-Policy in Klammern. Hinter der Default-Policy sehen Sie, wie viele Pakete bereits von dieser Default-Policy bearbeitet wurden und wie viele Bytes das sind. Anschließend werden bei jeder Kette die Regeln aufgezählt.

In der ersten Spalte wird die Nummer der Regel angezeigt. Diese Nummer kann später verwendet werden, um diese Regel zu löschen. In der zweiten und dritten Spalte wird wieder die Anzahl der Pakete und die Anzahl der Bytes angegeben, auf die diese Regel zutraf. Hiermit können Sie auch Ihren Netzwerkverkehr auswerten.[1] Dann sehen Sie das IP-Protokoll (icmp) und mögliche Optionen. Die nächsten beiden Spalten geben die Netzwerkkarte an, die das Paket verwenden muss, damit diese Regel zutrifft. Es ist möglich, bei der Erzeugung einer Regel sowohl die Netzwerkkarte anzugeben, über die das Paket den Rechner erreichen muss (`-i`; `--in-interface`), als auch die Netzwerkkarte, über die es den Rechner verlassen muss (`-o`; `--out-interface`). In unserer Regel haben wir keine Netzwerkkarte angegeben, daher befinden sich hier Sternchen als Wildcard für alle verfügbaren Karten. Nun folgen noch die Quelladresse (`-s`; `--src`; `--source`) und die Zieladresse (`-d`; `--dst`; `--destination`) des zu filternden Pakets. Auch hier haben wir keine Angabe gemacht, daher hat Iptables hier 0.0.0.0/0 eingetragen. Dies entspricht einem Netzwerk mit allen möglichen IP-Adressen.

EXKURS

CIDR-Notation. *Mit der Einführung des Subnettings löste man sich von den starren Netzwerkklassen A, B und C. Um die Netzmasken einfacher zu schreiben, wurde die Classless Internet Domain Routing-Notation (CIDR) eingeführt. Anstelle von 255.255.255.0 nutzt diese Notation /24. Für die Umwandlung der Netzmaske in die CIDR-Notation stellen Sie sich die Netzmaske binär vor:*

255.255.255.0 = 11111111.11111111.11111111.00000000

1 Sie können auch Regeln erzeugen, die kein Ziel haben. Diese Regeln zählen nur die betroffenen Pakete!

Nun zählen Sie die Einsen in der Netzmaske von links ab. Es handelt sich um drei mal acht Einsen, also 24 Einsen. Diese werden nun als CIDR-Notation nach dem Schrägstrich geschrieben. Im Folgenden finden Sie die üblichen Netzmasken in CIDR-Notation:

/0.0.0.0 = /0 /255.0.0.0 = /8 /255.255.0.0 = /16 /255.255.255.0 = /24 /255.255.255.255 = /32

Wenn Sie den Iptables-Befehl mehrfach wiederholen, fügen Sie mehrere Regeln in der INPUT-Kette hinzu:

```
[root@bibo root]# iptables -A INPUT -p icmp -j DROP
[root@bibo root]# iptables -A INPUT -p icmp -j DROP
[root@bibo root]# iptables -A INPUT -p icmp -j DROP
[root@bibo root]# iptables -L INPUT
Chain INPUT (policy ACCEPT)
target               prot opt source               destination
DROP                 icmp --  anywhere             anywhere
DROP                 icmp --  anywhere             anywhere
DROP                 icmp --  anywhere             anywhere
DROP                 icmp --  anywhere             anywhere
```

Jede Regel wird „unten“ an der Kette angehängt. Wenn Sie nur eine Kette anzeigen möchten, können Sie, wie hier gezeigt, auch den Namen der Kette bei dem Kommando `Iptables -L` mit angeben. Um einzelne Regeln wieder zu löschen, können Sie die Option `-D` beziehungsweise `--delete` verwenden. Um die Regel Nummer Eins zu löschen, verwenden Sie:

```
[root@bibo root]# iptables -D INPUT 1
```

Existiert keine Regel mit dieser Nummer, erhalten Sie folgende Fehlermeldung:

```
[root@bibo root]# iptables -D INPUT 5
iptables: Index of deletion too big
```

Alle weiteren Regeln rücken automatisch nach dem Löschen auf. Löschen Sie die Regel Eins, so wird die Regel Zwei zu Eins, Drei zu Zwei usw.

Möchten Sie alle Regeln löschen, können Sie den Befehl `iptables -F` verwenden. Dieser Befehl löscht alle Regeln in allen Ketten der Filtertabelle. Möchten Sie nur sämtliche Regeln der INPUT-Kette löschen, verwenden Sie:

```
[root@bibo root]# iptables -F INPUT
[root@bibo root]# iptables -L

Chain INPUT (policy ACCEPT)
target               prot opt source               destination
```

```
Chain FORWARD (policy ACCEPT)
target              prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target              prot opt source               destination
```

Nun sind Sie wieder da, wo wir angefangen haben. Sämtliche Ketten sind leer. Die Default-Policies sind auf `ACCEPT` gestellt.

Nun bauen wir die Regeln anders auf. Während bis jetzt alles erlaubt war und nur das ICMP-Protokoll mit einer spezifischen Regel verboten wurde, möchten wir nun alles verbieten und nur ausgewählte Protokolle erlauben. Hierzu setzen Sie zunächst die Default-Policy in der `INPUT`-Kette auf `DROP`.

STOP

Stellen Sie sicher, dass Sie lokal angemeldet sind. Wenn Sie in diesem Moment über das Netzwerk arbeiten, wird der folgende Befehl Ihre Verbindung unterbrechen!

Dies erledigen Sie mit folgendem Befehl:

```
[root@bibo root]# iptables -P INPUT DROP
```

Nun sollte Ihr Rechner keine Pakete mehr entgegennehmen. Wenn Sie auf Ihrem Rechner nun auch keine neuen Fenster mehr öffnen können, dann liegt das am lokalen Netzwerkverkehr, der für die grafische Oberfläche benötigt wird. Solange noch Fenster geöffnet sind, sollten Sie in diesen aber ohne Probleme weiterarbeiten können.

Der Ping sollte auch nicht mehr möglich sein. Auch eine Secure-Shell-Verbindung von außen sollte keine Antwort liefern. Um nun ausgewählte Protokolle wieder nutzen zu können, müssen Sie diese durch einzelne Regeln erlauben.

Erlauben Sie als Erstes wieder den Ping, indem Sie spezifisch das Protokoll ICMP akzeptieren:

```
[root@bibo root]# iptables -A INPUT -p icmp -j ACCEPT
```

Wenn Sie auf Ihrem System einen SSH-Server betreiben, können Sie auch die SSH-Verbindungsaufnahmen wieder gestatten. Benutzen Sie dazu bitte folgenden Befehl:

```
[root@bibo root]# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
[root@bibo root]# iptables -vnL
Chain INPUT (policy DROP 51 packets, 5528 bytes)
 pkts bytes target prot opt in out source     destination
 4    372   ACCEPT icmp --  *  *   0.0.0.0/0  0.0.0.0/0
15    8121  ACCEPT tcp  --  *  *   0.0.0.0/0  0.0.0.0/0 tcp dpt:22
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target prot opt in out source    destination

Chain OUTPUT (policy ACCEPT 398K packets, 23M bytes)
 pkts bytes target prot opt in out source
```


Bitte Einfügung als "Stop-Icon" bestätigen.

Dieser Befehl fügt eine Regel in der INPUT-Kette hinzu, die alle Pakete akzeptiert, die das TCP-Protokoll verwenden (`-p tcp`) und an den Port 22 gerichtet sind (`--dport`; `--destination-port` 22). Falls Sie nun mit `ssh localhost` dies testen wollen, wird das noch nicht funktionieren. Sie benötigen hier noch zusätzlich folgende Regel:

```
[root@bibo root]# Iptables -A INPUT -p tcp --sport 22 -j ACCEPT
```

Dies ist erforderlich, da auch der Client von der Firewall betroffen ist. Wenn Sie Ihren SSH-Client auf einer zweiten Maschine aufrufen, ist das nicht der Fall. Dies setzt natürlich einen laufenden SSH-Server auf Ihrer Maschine voraus.

Um nun den Originalzustand wiederherzustellen, genügt es nicht, den Befehl `iptables -F` aufzurufen. Dieser Befehl löscht zwar sämtliche Regeln in den Ketten, verändert aber nicht die Default-Policies! Da die Default-Policy in Ihrer INPUT-Kette aber immer noch auf DROP gestellt ist, werden weiterhin alle Pakete verworfen. Sie müssen zusätzlich noch den folgenden Befehl eingeben:

```
[root@bibo root]# iptables -P INPUT ACCEPT
```

Damit setzen Sie die Default-Policy wieder auf den ursprünglichen Wert. Jetzt werden wieder alle Pakete in der INPUT-Kette akzeptiert.

5.3 Ihr erstes Firewall-Skript

Nachdem Sie in dem letzten Abschnitt ein wenig mit dem `Iptables`-Befehl herumgespielt haben, sollen Sie nun Ihr erstes Firewall-Skript entwickeln. Dieses Skript soll die Regeln für ein Firewall-Gateway nach Abbildung 5.4 implementieren. Das Skript soll sämtliche Verbindungen von innen nach außen zulassen, aber keine Verbindungen von außen nach innen. Hierzu benötigen Sie zwei Rechner. Einer dieser beiden Rechner sollte über zwei Netzwerkkarten verfügen und mit dem Internet verbunden sein. Der andere Rechner sollte eine Netzwerkkarte besitzen und die zweite Karte des ersten Rechners erreichen können.

Im Weiteren benutze ich die Worte „innen“ und „außen“, um jeweils das innere, von der Firewall geschützte Netz beziehungsweise das äußere Internet zu bezeichnen.

INFO

Virtualisierung

Falls Sie nicht über zwei Rechner verfügen, können Sie an dieser Stelle auch einen Rechner (den Client) virtuell betreiben. Hierzu befindet sich auf der CD ein kompletter virtueller Rechner, den Sie direkt von der CD aus starten können. Gehen Sie hierzu in das Verzeichnis `/VirtualClient`. Diese Virtualisierung nutzt KVM bzw. QEMU. Hierzu müssen mindestens das Qemu- und/oder das KVM-Paket und der Befehl `tunctl` installiert sein. Sie müssen das Skript als Benutzer Root aufrufen, damit es über die notwendigen Rechte verfügt. Das Root-Kennwort lautet „kennwort“.

Nun rufen Sie das Skript `./start_client` auf. Es bootet dann eine virtuelle Linux-Maschine. Diese wird anschließend die Aufgabe des Clients aus Abbildung 5.4 übernehmen. Die IP-Adressen sind schon entsprechend eingestellt. Lediglich die Namen der Netzwerkkarten werden sich noch ändern. Dies wird noch an der entsprechenden Stelle beschrieben.

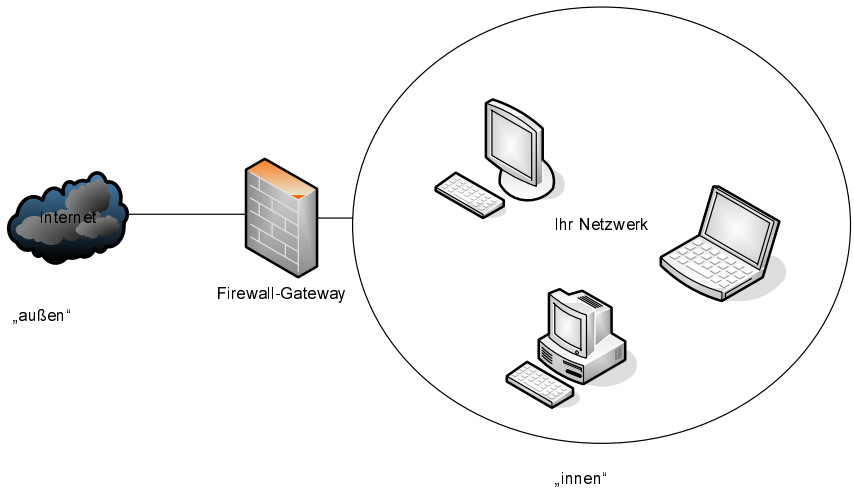


Abbildung 5.4: **Das Firewall-Gateway schützt Ihr internes Netz vor den Zugriffen aus dem Internet.**

Die Rechner sollten über folgende IP-Adressen verfügen:

- externe Karte der Firewall: beliebige IP-Adresse außer 10.x.x.x
- interne Karte der Firewall: IP-Adresse 10.0.0.1, Netzmaske 255.0.0.0
- Karte des Clients: IP-Adresse 10.0.0.100, Netzmaske 255.0.0.0, Gateway 10.0.0.1

Diese IP-Adressen sind bei dem virtuellen KVM-Client bereits voreingestellt. Wenn Sie alles richtig konfiguriert haben, sollten Sie in der Lage sein, von dem Firewall-System Ihren Client anzupingen (`ping 10.0.0.100`) und umgekehrt (`ping 10.0.0.1`). Funktioniert so weit alles, beginnen Sie mit Ihrem Firewall-Skript. Warum sollten Sie ein Skript schreiben?

- Nun, Änderungen sind in einem Skript mit einem Editor möglich.
- Sie können sämtliche Regeln in das Skript aufnehmen und dann auf einmal ausführen.
- Nach einem Reboot vergisst der Linux-Rechner alle Regeln. Mit einem Skript können Sie diese wieder laden.
- In einem Skript können Sie Kommentare einfügen, die später (auch nach einem Jahr) erläutern, warum Sie diese oder jene Regel für besonders wichtig hielten.
- Außerdem können Sie in einem Skript Variablen nutzen und so die Wartung des Skripts stark vereinfachen.

Ein ordentliches Shell-Skript sollte immer mit etwa den folgenden Zeilen beginnen:

```
#!/bin/bash
# Autor  : Ralf Spenneberg
# Version: 0.1
# Datum  : 17. Dez 2004
#
# Dieses Skript lädt die Regeln für die Firewall
```

Die erste Zeile ist zwingend erforderlich. Durch diese Zeile erkennt das Betriebssystem, mit welchem Interpreter das Skript ausgeführt werden muss. Unter Linux gibt es viele Skriptsprachen. Jede besitzt ihren eigenen Interpreter. Da diese nicht kompatibel sind, sollten Sie den zu wählenden Interpreter in der ersten Zeile mit „#¡' angeben. Diese Zeichenfolge wird auch als „She-Bang" bezeichnet. Die anderen Angaben können Sie entsprechend Ihren Wünschen anpassen. Für die dauerhafte Pflege Ihrer Skripte beschäftigen Sie sich am besten mit einem Revision-Control-System (RCS). Es gibt RCS und CVS. RCS ist auf fast jedem Linux-System vorhanden und verfügt über eine einleitende Manpage (`rcsintro(1)`).

Zunächst sollte Ihr Skript sicherstellen, dass der Befehl `Iptables` immer gefunden wird. Da sich möglicherweise auch der Pfad in einer zukünftigen Distributionsversion ändern kann, sollten Sie hierfür eine Variable verwenden. Fügen Sie die folgende Zeile zu Ihrem Skript hinzu, und Sie können den Pfad dann bei Bedarf anpassen:

```
IPTABLES="/sbin/iptables"
```

Nun sollten Sie alle Ketten zunächst durch das Skript sperren lassen. Dazu setzen Sie die Default-Policies in allen Ketten auf `DROP`:

```
#!/bin/bash
# Autor  : Ralf Spenneberg
# Version: 0.1
# Datum  : 17. Dez 2004
#
# Dieses Skript lädt die Regeln für die Firewall

IPTABLES="/sbin/iptables"

# Verwerfe erst mal alles, indem
# die Default-Policy auf DROP gesetzt wird
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
```

Der Zugriff auf die Variable erfolgt durch Voranstellen des Dollarzeichens. Bei der Zuweisung des Wertes ist es wichtig, dass um das Gleichheitszeichen keine Leerzeichen vorkommen.

Nun müssen die Regeln hinzugefügt werden. Diese sollen nur Verbindungen von innen nach außen zulassen. Für die Firewall handelt es sich um weitergeleitete Pakete. Daher müssen diese in der FORWARD-Kette gesetzt werden. Die Regeln müssen die Richtung unterscheiden können. Diese Unterscheidung ist am einfachsten über die Netzwerkkarten möglich. Sinnvollerweise erzeugen Sie für die Namen Ihrer Netzwerkkarten auch wieder Variablen. Vielleicht ändert sich mal eine Netzwerkkarte. Dann können Sie durch eine Modifikation Ihres Skripts dieses an die neue Situation anpassen.

```
INTDEV="eth0" EXTDEV="eth1"
```

Wenn Sie den virtuellen KVM-Client einsetzen, heißt die interne Netzwerkkarte `tap0`. Das bedeutet, Sie müssen die Zeile wie folgt abändern:

```
INTDEV="tap0"
```

Diese Variablen fügen Sie sinnvollerweise am Anfang direkt hinter der Iptables-Variable ein. Um nun die Verbindung von innen zu erlauben, fügen Sie die folgenden Regeln am Ende Ihres Skripts an:

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Die erste Regel prüft, ob ein Paket über die interne Netzwerkkarte die Firewall erreicht (`-i $INTDEV`; `--in-interface`) und über die externe Netzwerkkarte die Firewall verlässt (`-o $EXTDEV`; `--out-interface`).[2] Zusätzlich prüft die Regel den Zustand des Pakets. Hierzu pflegt das Linux-System eine Tabelle, in der jede erlaubte Verbindung eingetragen wird. Ein Paket besitzt den Zustand `NEW`, wenn es zu keiner bisher eingetragenen Verbindung passt. Es besitzt den Zustand `ESTABLISHED`, wenn es bereits eine Verbindung in der Tabelle gibt, der dieses Paket zugeordnet werden kann. Der Zustand `RELATED` wird verwendet, wenn das Paket nicht Teil der Verbindung ist, sondern in einer anderen Beziehung zu dieser Verbindung steht. Hierbei kann es sich zum Beispiel um eine ICMP-Fehlermeldung handeln. Eine ausführlichere Erklärung der Verbindungsüberwachung (engl. Connection Tracking) finden Sie in Abschnitt 5.5 und in Kapitel 19. Die erste Regel akzeptiert so jedes Paket, das eine neue Verbindung aufbauen möchte und von innen nach außen geschickt wird. Gleichzeitig wird diese Verbindung dann in der Tabelle eingetragen.

Die zweite Regel prüft dann lediglich, ob ein Paket den Zustand `ESTABLISHED` oder `RELATED` aufweist. Alle diese Pakete werden von der zweiten Regel akzeptiert. So ist ein Verbindungsaufbau nur von innen nach außen möglich.

Warum ist das sicher? Wenn nun ein Paket von außen eine neue Verbindung aufbauen wollte, würde es von der Firewall nicht akzeptiert werden. Dieses Paket besitzt den Zustand NEW. In der ersten Regel werden zwar solche Pakete akzeptiert, aber dieses Paket würde nicht die Bedingung erfüllen, über die interne Netzwerkkarte die Firewall erreicht zu haben. Damit ein Paket von außen zugelassen wird, ist erst ein Paket von innen erforderlich, das dieses Paket anfordert und die Verbindung in der Zustandstabelle einträgt. Sobald dies erfolgt ist, erhalten alle weiteren Pakete der Verbindung den Zustand ESTABLISHED und werden von der zweiten Regel unabhängig von ihrer Richtung akzeptiert.

2 Noch sicherer können Sie die Regel definieren, wenn Sie auch noch prüfen, ob die Absender-Adresse des Pakets aus Ihrem Netzwerk stammt. Hierzu können Sie eine Variable `INTNET=10.0.0.0/8` anlegen und diese zusätzlich mit der Option `-s $INTNET` der Regel hinzufügen.

CE[P] In Abb. 7-4 sind keine Ketten tzu sehen. Bitte Abbildungsverweis prüfen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

INFO

Warum werden die Regeln in der FORWARD-Kette hinzugefügt? Wieso nicht in der INPUT- oder OUTPUT-Kette? Wenn Sie sich noch mal die Abbildung 5.4 ansehen, wird Ihnen auffallen, dass die Pakete, die durch das Gateway geroutet werden, nur von der FORWARD-Kette bearbeitet werden. Die INPUT- und die OUTPUT-Kette sind hier gar nicht beteiligt. Diese Ketten werden nur aktiv, wenn das Gateway selbst Kommunikationspartner wäre. *Dies wird in Kapitel 7 genauer besprochen!*

Damit ist das Skript aber noch nicht ganz fertig. Zwei wesentliche Punkte fehlen noch, damit es funktioniert:

1. Forwarding: Selbst wenn die Firewall momentan von innen Pakete für die Weiterleitung nach außen erhält, wird sie diese im Moment noch nicht weiterleiten.

2. Masquerading: Selbst wenn die Weiterleitung aktiviert wurde und ein interner Client Netzwerkpakete nach außen sendet, werden die Systeme außerhalb der Firewall nicht wissen, wohin sie antworten sollen, da den Systemen die IP-Adresse und das Netzwerk des Clients unbekannt ist. Die IP-Adresse der Firewall ist bekannt. Daher kann die Firewall dieses Problem beheben, indem sie eine Network Address Translation durchführt. Hierbei wird die originale Absenderadresse des Clients durch die Adresse der Firewall ausgetauscht. Dies wird auch als Masquerading bezeichnet.

Zunächst soll das Forwarding aktiviert werden. Derartige Funktionen werden bei Linux über das virtuelle `/proc`-Dateisystem angeschaltet. Dieses Dateisystem erlaubt die Konfiguration des Linux-Kernels zur Laufzeit. Für das Forwarding ist die Datei `/proc/sys/net/ipv4/ip_forward` zuständig. Enthält diese Datei eine 0, so ist das Forwarding nicht aktiv. Bei einer 1 leitet der Linux-Kernel zwischen allen Netzwerkkarten die Pakete als Router weiter.

Es gibt grundsätzlich zwei Möglichkeiten, um nun das Forwarding einzuschalten:

» Sie verwenden den `/bin/echo`-Befehl. Mit diesem Befehl können Sie in der Datei folgendermaßen eine 1 speichern:

```
ECHO="/bin/echo"
$ECHO "1" > /proc/sys/net/ipv4/ip_forward
```

» Die zweite Möglichkeit ist der Befehl `sysctl`. Dieser Befehl kann die Werte im `/proc`-Dateisystem lesen und setzen. Die folgenden Zeilen erreichen dasselbe wie der Echo-Befehl weiter oben.

```
SYSCTL="/bin/sysctl" $SYSCTL -w net.ipv4.ip_forward=1
```

Für welchen Befehl Sie sich hier entscheiden, bleibt Ihnen überlassen. Weitere Informationen über den Befehl `sysctl` und das `/proc`-Dateisystem erhalten Sie in Kapitel 23. Network Address Translation (NAT) und die NAT-Tabelle werden in Abschnitt 5.7 ausführlich besprochen. Hier soll nur die erforderliche Regel vorgestellt und erklärt werden. Damit das Masquerading, das eine spezielle Form des NAT ist, funktioniert, benötigen Sie die folgende Regel:

```
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -j MASQUERADE
```

Das Masquerading erfolgt in der POSTROUTING-Kette der NAT-Tabelle. Diese Kette ist die letzte Kette, die ein Paket durchläuft, das den Rechner verlässt. In dieser Kette tauscht diese Regel die Absenderadresse jedes Pakets gegen die Absenderadresse der Firewall aus und merkt sich die Originaladresse. So kann die Firewall bei einem Antwortpaket die originale Adresse wieder einsetzen. Dieser Vorgang soll nur für Pakete aktiviert werden, die die Firewall nach außen verlassen (`-o $EXTDEV`). Die NAT-Tabelle enthält auch noch die Ketten PREROUTING und OUTPUT. Diese Ketten und die weiteren Funktionen der NAT-Tabelle werden in Abschnitt 5.7 besprochen.

Sinnvoll sind in dem Skript zu Beginn nun noch einige Befehle, die möglicherweise vorhandene Regeln aufräumen und entfernen, ehe neue Regeln hinzugefügt werden. Dies kann mit den folgenden beiden Zeilen erreicht werden:

```
$IPTABLES -F
$IPTABLES -t nat -F
```

Das komplette Skript sieht nun wie folgt aus:

Listing 5.1: **Das erste Firewall-Skript erlaubt nur Verbindungen von innen.**

```
#!/bin/bash
# Autor  : Ralf Spenneberg
# Version: 0.1
# Datum  : 17. Dez 2004
# Dieses Skript lädt die Regeln für die Firewall

INTDEV="eth1" # Achtung: Bei Einsatz des virtuellen Client lautet diese ↵
        Variable
              # INTDEV="tap0"
EXTDEV="eth0"

# Definiere einige Befehle
ECHO="/bin/echo"
SYSCTL="/bin/sysctl"
IPTABLES="/sbin/iptables"

# Verwerfe erst mal alles
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Leere die Ketten
$IPTABLES -F

# Die NAT-Tabelle muss extra geleert werden
$IPTABLES -t nat -F
```

```
# Akzeptiere Verbindungsaufbauten von innen
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -m state --state NEW -j ACCEPT

# Akzeptiere alle Pakete, die Teil einer aufgebauten Verbindung sind
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# Maskiere alle Pakete bei der Weiterleitung nach außen
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -j MASQUERADE

# Aktiviere das Forwarding
$ECHO "1" > /proc/sys/net/ipv4/ip_forward

# Alternativ:
$SYSCTL -w net.ipv4.ip_forward=1
```

INFO

Möglicherweise verhält sich Ihre Firewall anders, als Sie es erwarten würden. Es kann sein, dass bei einem Zugriff auf einen E-Mail- oder FTP-Server scheinbar die Verbindung erst verzögert zustande kommt. Schuld daran ist die Tatsache, dass Sie alle Verbindungen, die von außen aufgebaut werden, ohne Fehlermeldung verwerfen. Leider bauen gerade UNIX-E-Mailserver und FTP-Server bei einem Verbindungsaufbau Ihrerseits eine Verbindung zu Ihnen auf dem Port 113/tcp auf. Hier horcht bei UNIX-Systemen normalerweise der Identd. Der E-Mailserver und der FTP-Server fragen diesen Dienst, welcher Benutzer die ursprüngliche Verbindung aufgebaut hat. Da dieser Verbindungsversuch verworfen wird, wiederholen Sie den Versuch mehrfach. So lange müssen Sie mit Ihrer Verbindung warten. Am einfachsten lösen Sie das Problem, indem Sie Verbindungen auf dem Port 113/tcp nicht verwerfen, sondern mit einer Fehlermeldung ablehnen:

```
$IPTABLES -A INPUT -i $EXTDEV -p tcp --dport 113 -j REJECT
```

Nun ist es an der Zeit, dieses Skript zu testen, falls Sie es nicht schon vorher getan haben. Hierfür speichern Sie dieses Skript unter dem Namen `firewall` ab und machen es anschließend ausführbar: `chmod 755 firewall`. Leider schleichen sich bei der Eingabe häufig Tippfehler ein, die nur schwer gefunden werden können, da die Fehlermeldungen der Shell nicht besonders aussagekräftig sind – so zum Beispiel:

```
[root@bibo root]# ./firewall
iptables: No chain/target/match by that name
```

Erfreulicherweise bietet die Bash-Shell eine Option, mit der die Zeile, auf der sich der Fehler eingeschlichen hat, leicht ermittelt werden kann. Die Option `-x` gibt jede Zeile vor der Ausführung auf dem Bildschirm aus. Rufen Sie daher das Skript mit folgendem Befehl auf: `bash -x firewall`. Sie müssen dann nur die Fehlermeldung wiederfinden und die Zeile davor genau analysieren:

```
... gekürzt ...
+ /sbin/iptables -P INPUT DROP
```

```
+ /sbin/iptables -P OUTPUT DROP
+ /sbin/iptables -P FORWARD DROP
+ /sbin/iptables -A FORWARD -i eth0 -o eth1 -m state --state NEW -j ↵
        ACCEPT
+ /sbin/iptables -A FORARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables: No chain/target/match by that name
+ /sbin/iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
 ... gekürzt ...
```

Hier kann der Übeltäter sehr schnell erkannt werden. Es handelt sich um einen Tippfehler in dem Namen der FORWARD-Kette. Anstelle von FORWARD wurde FORARD verwendet. Herzlichen Glückwunsch, wenn keine Fehler auftreten. Sie sollten jetzt schon über einen funktionsfähigen Firewall-Regelsatz verfügen. Dieser erlaubt im Moment der Firewall selbst keinerlei Teilnahme am Netzwerkverkehr. Die Firewall kann also zum Beispiel nicht pingen und keinen Ping beantworten. Jedoch sollte ein Client durch die Firewall auf andere Rechner zugreifen können. Versuchen Sie doch einfach mal Ihr Glück mit einem Ping vom Client ins Internet. Wenn es nicht funktioniert, finden Sie am Ende dieses Abschnitts einige weitere Hinweise zur Fehlersuche.

INFO

Warum kann die Firewall nicht pingen?

Die INPUT- und die OUTPUT-Kette enthalten keine Regeln. Da die Default-Policies dieser Ketten auf DROP gesetzt sind, werden in diesen Ketten daher keine Pakete akzeptiert, sondern sämtliche Pakete verworfen. Sowohl die Pakete, die von außen an die Firewall selbst gerichtet sind, als auch die Pakete, die die Firewall versenden möchte, sind davon betroffen. Nur in der FORWARD-Kette akzeptiert die Firewall Netzwerkpakete. Daher ist ein Ping durch die Firewall möglich! In Kapitel 7 werden die INPUT- und die OUTPUT-Kette genauer betrachtet. Dort lernen Sie, wie Sie auch hier sinnvoll filtern. Wenn Sie dieses Verhalten für den Moment abstellen wollen, ändern Sie in Ihrem Skript die Default-Policy der INPUT- und der OUTPUT-Kette auf ACCEPT. Dann schützt sich die Firewall aber selbst nicht mehr!

Da Sie gerade in Übung sind, was das Schreiben von Firewall-Skripten betrifft, sollten Sie auch ein Skript erzeugen, das die Firewall wieder beendet. Hierzu können Sie das folgende Skript verwenden:

Listing 5.2: **Dieses Skript (firewall_stop) entfernt alle Regeln und setzt die Default-Policies wieder auf ACCEPT zurück.**

```
#!/bin/bash
# Autor  : Ralf Spenneberg
# Version: 0.1
# Datum  : 17. Dez 2004
# Dieses Skript löscht die Regeln für die Firewall

ECHO="/bin/echo" IPTABLES="/sbin/iptables"

# Deaktiviere das Forwarding
$ECHO "0" > /proc/sys/net/ipv4/ip_forward
```

```
# Lösche alle vorhandenen Regeln
$IPTABLES -F $IPTABLES -t nat -F

# Stelle die Default-Policies wieder her
$IPTABLES -P INPUT ACCEPT $IPTABLES -P OUTPUT ACCEPT
$IPTABLES -P FORWARD ACCEPT
```

Mit diesem Skript können Sie Ihre Firewall immer wieder zu Testzwecken beenden.

5.3.1 Fehlersuche

Ich hoffe, dass die meisten Leser mit dem Nachvollziehen des Kapitels keine Probleme[3] haben werden, da gerade die ersten Versuche häufig darüber entscheiden, ob man von einem System begeistert ist oder nicht. Daher habe ich mich bemüht, jeden Schritt genau zu erklären und an alle Eventualitäten zu denken. Dennoch kann es zu weiteren Fehlern kommen, die ich hier zu erklären versuche. Prüfen Sie aber bitte als Erstes Ihr Skript auf syntaktische Richtigkeit. Sobald bei dem Aufruf Ihres Firewall-Skripts eine Fehlermeldung ausgegeben wird, kann es nicht funktionieren! Suchen Sie den Fehler, und beheben Sie ihn, wie oben erläutert.

Wenn Ihr Skript erfolgreich lädt, Sie aber dennoch keinen Rechner außerhalb Ihrer Firewall erreichen können, gehen Sie bitte bei der Fehlersuche folgendermaßen vor:

1. Beenden Sie Ihre Firewall mit dem `firewall_stop`-Skript.
2. Prüfen Sie nun, ob Sie von Ihrem Client aus Ihre Firewall erreichen können: `ping 10.0.0.1`. Klappt das nicht, überprüfen Sie die Netzwerkverbindung. Ansonsten fahren Sie in der Fehlersuche fort.
3. Prüfen Sie nun, ob Sie von der Firewall einen Rechner außerhalb erreichen können. Klappt dies nicht, überprüfen Sie bitte die Netzwerkverbindung Ihrer Firewall.
4. Prüfen Sie nun die Routing-Konfiguration des Clients. Ist die interne IP-Adresse der Firewall das Standard-Gateway des Clients? Dies können Sie mit dem Befehl `/sbin/route` erledigen:

   ```
   [root@bibo root]# /sbin/route
   Kernel IP Routentabelle
   Ziel        Router        Genmask         Flags Metric Ref Use Iface
   10.0.0.1    *             255.255.255.255 UH    0      0   0   eth0
   192.168.0.0 *             255.255.255.0   U     0      0   0   eth1
   default     192.168.0.254 0.0.0.0         UG    0      0   0   eth1
   ```

   Hier hat das Default-Gateway die IP-Adresse 192.168.0.254. Sie können mit `route del default` vorübergehend das Default-Gateway entfernen und es mit `route add default gw <ip-adresse>` neu setzen.

3 Die Beispiele wurden auf den handelsüblichen Distributionen und Kerneln getestet. Wenn Sie einen selbst gebauten Kernel und/oder Iptables-Befehl verwenden, kann ich nicht dafür garantieren. Dann sollten Sie aber über die Erfahrung verfügen, diesbezügliche Fehler selbst zu lösen.

5. Wenn Sie DNS-Namen verwenden möchten, prüfen Sie bitte, ob der Client die richtigen DNS-Server kennt. Wenn Sie keinen direkten Zugriff auf das Internet haben, können die DNS-Server speziell des virtuellen KVM-Clients falsch eingerichtet sein. Die DNS-Server werden in der Datei `/etc/resolv.conf` gepflegt. Tragen Sie hier die DNS-Server ein, deren Funktion Sie sicher kennen.
6. Aktivieren Sie testweise auf der Firewall das Forwarding und das Masquerading:

```
[root@bibo root]# iptables -t nat -A POSTROUTING -o $EXTDEV -j ↩
        MASQUERADE
[root@bibo root]# echo "1" > /proc/sys/net/ipv4/ip_forward
```

   Ersetzen Sie hierbei `$EXTDEV` durch den Namen Ihrer externen Netzwerkkarte. Prüfen Sie anschließend, ob Sie nun von Ihrem Client aus den zuvor getesteten Rechner außerhalb Ihrer Firewall erreichen können.
7. Starten Sie nun Ihr Firewall-Skript erneut. Prüfen Sie, ob Sie immer noch den Rechner erreichen können.

5.4 Einbindung in den Bootprozess

Im letzten Abschnitt haben Sie erfolgreich Ihr erstes Firewall-Skript geschrieben. Vielleicht ist Ihnen bereits aufgefallen, dass nach einem Neustart des Systems alle Regeln, die durch das Skript gesetzt wurden, wieder verloren sind. Für eine richtige Firewall ist es daher erforderlich, dass bei jedem Neustart diese Regeln neu geladen werden. Bei den meisten Linux-Distributionen kümmert sich der SysV-Init-Prozess um den Bootvorgang, die Initialisierung des Systems und das Starten der Dienste.

Welcher Dienst bei dem Bootvorgang des Rechners gestartet wird, hängt von dem Betriebszustand des Rechners ab. Dieser Betriebszustand wird als Runlevel bezeichnet. Die Distributionen OpenSUSE[TS q] und Fedora unterscheiden die Runlevel aus Tabelle 5.1[TS r].

Tabelle 5.1: **Runlevel und ihre Bedeutung**

Runlevel	Bedeutung
0	Halt-Zustand
1	Single-User
2	Multi-User ohne Netzwerkfunktionen
3	Multi-User mit allen Netzwerkdiensten
4	Vorkonfiguriert wie 3, aber ohne Verwendung
5	Multi-User wie 3, aber mit grafischer Oberfläche
6	Reboot
7–9	Nicht konfiguriert

TS q Bitte bestätigen: Schreibung generell nach „OpenSUSE“ angepasst.

TS r Bitte bestätigen Sie diesen Verweis.

In welchem Zustand sich Ihr System befindet, können Sie mit dem Befehl `runlevel` oder `who -r` ermitteln:

```
[root@bibo root]# runlevel
N 5
[root@bibo root]# who -r
Runlevel 5          Dec 20 06:37          last=S
```

Beide Befehle sind sich einig, dass sich mein System im Moment in Runlevel 5 befindet. Welcher Runlevel bei dem nächsten Reboot automatisch von dem System gewählt wird, können Sie in der Datei `/etc/inittab` festlegen:

Listing 5.3: **Der Default-Runlevel wird in der Datei /etc/inittab definiert.**

```
#
# inittab  This file describes how the INIT process should set up the ↵
        system in a certain run-level.
# Author:  Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#          Modified for RHS Linux by Marc Ewing and Donnie Barnes

# Default runlevel. The runlevels used by RHS are:
#  0 - halt (Do NOT set initdefault to this)
#  1 - Single user mode
#  2 - Multiuser, without NFS (The same as 3, if you do not have ↵
        networking)
#  3 - Full multiuser mode
#  4 - unused
#  5 - X11
#  6 - reboot (Do NOT set initdefault to this)
id:5:initdefault:
... gekürzt ...
```

Bei einer Firewall ist es sinnvoll, dafür zu sorgen, dass die Regeln bei dem Bootvorgang unabhängig von dem gewählten Runlevel geladen werden. Daher sollten Sie Ihr Skript in den Bootvorgang des Runlevels 3, 4 und 5 einbinden. In den Runleveln 1 und 2 stehen normalerweise keine Netzwerkdienste zur Verfügung, daher ist die Einbindung hier nicht zwingend erforderlich. Allerdings kann sie auch nicht schaden. Daher können Sie auch in diesen Runleveln Ihr Skript einbinden.

Die Einbindung erfolgt über Verknüpfungen in den Verzeichnissen `/etc/rcX.d`. Hier ist die Anleitung für die Einbindung in aller Kürze:

1. Kopieren Sie Ihr Skript in das Verzeichnis `/etc/init.d/`:
   `cp firewall /etc/init.d/MyFirewall`.
2. Machen Sie das Skript ausführbar: `chmod 755 /etc/init.d/MyFirewall`

3. Gehen Sie in das Verzeichnis `/etc/rc.d/rc3.d/`, und erzeugen Sie einen Startlink:

```
[root@bibo root]#  cd /etc/rc.d/rc3.d/
[root@bibo rc3.d]# ln -s /etc/init.d/MyFirewall S05MyFirewall
```

4. Wiederholen Sie den letzten Schritt für die Verzeichnisse mit den Nummern 4 und 5. Wenn Sie möchten, können Sie dies auch für die Verzeichnisse 1 und 2 durchführen.

Diese vier Schritte genügen üblicherweise für die Einbindung des Skripts. Wenn Sie nun Ihren Rechner neu starten, sollte Ihr Skript automatisch geladen werden.

Möglicherweise fragen Sie sich nun, warum ich die Nummer 05 für die Verknüpfung `S05MyFirewall` gewählt habe. Die S-Nummer definiert die Reihenfolge der zu startenden Dienste. Die K-Nummer definiert die Reihenfolge, in der die Dienste beendet werden. Dies ist erforderlich, da es durchaus Abhängigkeiten unter den Diensten gibt. So ist zum Beispiel der Start des Netzwerks erforderlich, bevor ein Netzwerkdienst gestartet werden kann.

Im Falle der Firewall-Regeln sollten diese geladen werden, bevor das Netzwerk gestartet wurde und die Netzwerkkarten initialisiert wurden. Dies erfolgt bei Red Hat an der Position 10. Dadurch sind die Regeln schon aktiv, wenn die Netzwerkkarten aktiviert werden. So entsteht keine Lücke, die ein Angreifer ausnutzen könnte.

Erfreulicherweise erlaubt der `Iptables`-Befehl das Laden der Regeln für Netzwerkkarten und IP-Adressen, die noch nicht existieren!

Die hier vorgenommene Einbindung ist sicherlich nicht distributionskonform. Das werden Sie daran erkennen, dass Sie mit den Werkzeugen Ihrer Distribution nun diesen Dienst nicht konfigurieren können. In dem Handbuch Ihrer Distribution werden Sie weitere Hinweise dazu finden, wie Sie das Skript distributionskonform einbinden, sodass Sie es auch mit den Werkzeugen Ihrer Distribution verwalten können.

Damit Ihnen nun die distributionseigene Firewall die Arbeit nicht wieder zunichte macht, sollte diese deaktiviert werden. Auf einer Red Hat-basierten Linux-Distribution erreichen Sie das mit folgendem Befehl:

```
[root@bibo /]# chkconfig iptables off
```

Bei SuSE verwenden Sie Yast und deaktivieren die Firewall.

Nun sollten Sie Ihren Rechner neu starten und prüfen, ob anschließend auch Ihre Firewall-Regeln so geladen wurden, wie Sie sie eingestellt haben.

5.5 Connection Tracking und Netzwerkverbindungen

Die wesentliche Eigenschaft, die Iptables von älteren Paketfiltern unterscheidet, ist die Verbindungsüberwachung. Während ein einfacher Paketfilter wirklich nur Pakete analysieren und filtern kann, ist Iptables in der Lage, Verbindungen zu filtern. Dies erreicht Iptables, indem es eine Liste der bekannten Verbindungen pflegt und so Pakete zuordnen kann. Um dies bes-

ser zu verstehen, beschreibe ich zunächst kurz die Funktionsweise eines einfachen Paketfilters wie ipchains. Die Entwicklung der notwendigen Regeln soll sowohl anhand einer TCP- als auch einer UDP-Verbindung durchgesprochen werden. Den Abschluss machen ICMP-Pakete. Im Anschluss erläutere ich die Verbindungsüberwachung (Connection Tracking) durch Iptables. Dann sollte Ihnen deutlich werden, dass dieses Connection Tracking den Aufbau wesentlich sichererer Firewalls ermöglicht. Gleichzeitig wird das Regelwerk einfacher und übersichtlicher.

5.5.1 Der zustandslose Paketfilter IPchains

Wenn Sie bereits mit dem Paketfilter IPchains (`ipchains`) des Linux-Kernels 2.2 Erfahrungen gesammelt haben, wird Ihnen dieses Kapitel sicherlich bekannt vorkommen. Bitte überspringen Sie es dennoch nicht, denn viele Iptables-Anwender haben bis heute nicht den wesentlichen Funktionsunterschied zwischen IPchains und Iptables richtig verstanden. IPchains ist ein klassischer Paketfilter, der in der Lage ist, die Protokoll-Header eines Pakets zu analysieren und anschließend das Paket zu akzeptieren oder zu verwerfen. Dabei muss IPchains sich jedes Paket einzeln ansehen und ist als Paketfilter nicht in der Lage, Abhängigkeiten der Pakete zu erkennen. Im Folgenden sollen Beispielregeln für typische Netzwerkverbindungen in IPchains entwickelt werden. Sie müssen die IPchains-Syntax hierfür nicht erlernen. Ich werde sie an der entsprechenden Stelle ausführlich erläutern. Sie ähnelt auch stark der Iptables-Syntax. Ich möchte Sie lediglich auf einige Sicherheitsprobleme im Zusammenhang mit Paketfiltern hinweisen.

5.5.2 IPchains und UDP

Beginnen wir mit einer typischen UDP-Verbindung. Die DNS-Namensauflösung ist eine typische UDP-Verbindung und sicherlich für einen Großteil des UDP-Verkehrs im Internet verantwortlich. Hier fragt ein DNS-Client einen DNS-Server nach der IP-Adresse für einen DNS-Namen oder umgekehrt. Die Anfrage ist schematisch in Abbildung 5.5 dargestellt, und ein TCPDump-Mitschnitt ist in Listing 5.4 zu sehen.

Listing 5.4: **TCPDump-Mitschnitt der DNS-Anfrage „`www.spenneberg.com`“**

```
[root@bibo tmp]# tcpdump -nvv port 53
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 96 ↵
        bytes
```

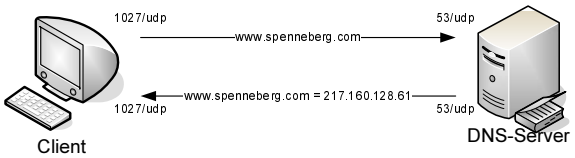


Abbildung 5.5: **Der DNS-Client fragt den DNS-Server mit dem UDP-Protokoll auf Port 53 nach der IP-Adresse für `www.spenneberg.com`.**

```
17:33:50.850459 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ↵
        17, length: 64) 192.168.0.112.35785 > 128.176.0.12.domain: [udp ↵
        sum ok] 52380+ A? www.spenneberg.com. (36)

17:33:50.891933 IP (tos 0x0, ttl 48, id 51717, offset 0, flags [none], ↵
        proto 17, length: 176) 128.176.0.12.domain > ↵
        192.168.0.112.35785: 52380 q: A? www.spenneberg.com. 2/2/2 www. ↵
        spenneberg.com. CNAME spenneberg.com., spenneberg.com.[|domain]
```

Hier schickt der Client 192.168.0.112 von Port 35785 ein UDP-Paket an den Server 128.176.0.12 auf dem Port 53 (domain). Der Server antwortet unter Verwendung derselben IP-Adressen und Ports. Diese Informationen können nun verwendet werden, um die Paketfilterregeln zu entwickeln.

Stellen Sie sich vor, zwischen dem Client und dem Server befindet sich ein Paketfilter, der genau diese Art von Anfragen erlauben soll (siehe Abbildung 5.6). Dabei sei aber nicht bekannt, bei welchem DNS-Server der Client die Frage stellt. Das bedeutet, dass der Paketfilter zwei Regeln benötigt:

1. Eine Regel muss das Paket von dem Client zum Server erlauben. Die IP-Adresse des Servers ist nicht bekannt. Lediglich das Protokoll (UDP) und der Port (53) stehen fest. Vom Client ist die IP-Adresse oder der Adressbereich bekannt.
2. Die zweite Regel muss das Antwortpaket des Servers an den Client erlauben. IPchains verfügt jedoch über kein Gedächtnis. Es ist nicht in der Lage, sich zu merken, dass ein Client ein Paket an einen DNS-Server herausgeschickt hat. Daher muss diese Regel jedes theoretisch mögliche Antwortpaket akzeptieren. Da der Client aber ein Paket an jeden möglichen DNS-Server geschickt haben kann, muss diese Regel Antwortpakete von jedem theoretisch möglichen DNS-Server im Internet erlauben. Sie kann diese Pakete nur anhand des verwendeten Protokolls (UDP) und Ports (53) erkennen.

Die Regel Eins würde in IPchains-Syntax folgendermaßen lauten:

```
CLIENTS=192.168.0.0/24
ipchains -A forward -p udp -s $CLIENTS -d any/0 53 -j ACCEPT
```

Diese Regel akzeptiert (`-j ACCEPT`) UDP-Pakete (`-p udp`) mit einer Absenderadresse aus dem Netzwerk 192.168.0.0/24 (`-s`) und einer beliebigen Zieladresse (`-d`) auf dem Port 53. So

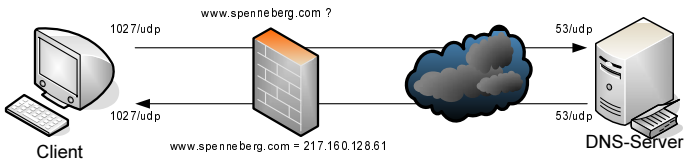


Abbildung 5.6: **Der Client wird nun durch einen Paketfilter geschützt.**

kann die Anfrage nach außen zum DNS-Server gelangen. Nun muss auch die Antwort akzeptiert werden. Regel Zwei sieht folgendermaßen aus:

```
ipchains -A forward -p udp -s any/0 53 -d $CLIENTS -j ACCEPT
```

Was leisten diese Regeln nicht? Diese Regeln können nicht verhindern, dass ein Angreifer UDP-Pakete mit dem Absenderport 53 von außen nach innen schickt. Dabei beschränken diese Regeln nicht den Zielport auf dem Client. Dies ist auch nicht möglich, da dieser Port nicht bekannt ist. Jeder Client kann einen fast beliebigen Port für seine Anfrage wählen. Bei jeder Anfrage wird er diesen auch neu wählen. Daher muss die Firewall Antworten auf jedem beliebigen Port erlauben.

Dies ist nicht ganz richtig, werden einige Leser jetzt bemerken. Und richtig, unter UNIX-ähnlichen Systemen darf ein Client nur einen nichtprivilegierten Port verwenden. Dies sind die Ports 1024 bis 65535. Meist verwendet der Client nur einen bestimmten Bereich, zum Beispiel würde ein Linux 2.6-Kernel einen Port-Bereich von 32768–61000 nutzen. Die meisten anderen Systeme halten sich auch an diese Konvention. Bei einem einfachen Paketfilter nutzt man diese Tatsache aus, um die Regeln noch sicherer zu machen:

Listing 5.5: **Der Client verwendet einen unprivilegierten Port für den Verbindungsaufbau.**

```
CLIENTS=192.168.0.0/24
ipchains -A forward -p udp -s $CLIENTS 1024:65535 -d any/0 53 -j ACCEPT
ipchains -A forward -p udp -s any/0 53 -d $CLIENTS 1024:65535 -j ACCEPT
```

Dennoch kann die zweite Regel einen Angreifer nicht daran hindern, UDP-Pakete durch die Firewall an einen beliebigen Client auf einem unprivilegierten Port zu schicken. Diese Regel muss jedes scheinbar sinnvolle Paket durchlassen. Da es durchaus einige UDP-Dienste gibt, die einen derartigen Port verwenden, ist es möglich, durch diese Firewall eine UDP-Verbindung von außen nach innen aufzubauen, vorausgesetzt, ich verwende als Client den Port 53/udp. Gute IPchains-Paketfilterkonfigurationen schließen diese Ports wieder aus.

INFO

Wie beeinflusse ich bei einem Angriff den Client-Port? Es gibt viele Möglichkeiten, dies zu tun. Wenn Sie aber nur schnell für einen Scan einer Firewall den Client-Port einstellen möchten, bieten die modernen Scan-Werkzeuge dies bereits an. So können Sie bei dem bekannten Werkzeug `nmap` (siehe Abschnitt 15.3) mit der Option `--source-port` den Client-Port einstellen. Der Befehl `nmap -sU --source-port 53 <client>` würde den Client scannen, vorausgesetzt, es wird nicht gleichzeitig NAT eingesetzt.

5.5.3 IPchains und TCP

Nachdem wir eine typische UDP-Verbindung betrachtet haben, soll nun eine TCP-Verbindung analysiert werden. Hier werden wir uns zwei verschiedene Applikationsprotokolle anschauen. Zunächst wird das HTTP-Protokoll betrachtet und anschließend das FTP-Protokoll. Wenn Sie sich hier wundern sollten, warum ich zwei Protokolle betrachte, werden Sie sich nach der Erklärung des FTP-Protokolls wahrscheinlich noch mehr wundern.

CE^S Bitte diese Änderung bestätigen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

Das TCP-Transport-Protokoll ist komplizierter als das UDP-Protokoll. Es garantiert im Gegensatz zum UDP-Protokoll die Übertragung der Informationen in der richtigen Reihenfolge.

INFO

TCP und UDP. *Wenn sie nach dem Unterschied zwischen TCP und UDP gefragt werden, antworten die meisten Netzwerkadministratoren, TCP sei verbindungsorientiert und UDP verbindungslos. Fragt man genauer nach, was dies denn bedeute, stehen viele Administratoren bereits auf dem Schlauch. Sie haben es zwar alle in der Vergangenheit gelernt, aber im Laufe der Jahre wieder vergessen.*

Ich möchte hier kurz erläutern, was für mich der wesentliche Unterschied zwischen diesen beiden Protokollen ist. Eine ausführlichere Betrachtung finden Sie in Abschnitt 42.6. TCP garantiert, dass die Applikation auf der Empfängerseite die Daten in derselben Reihenfolge erhält, in der sie losgeschickt wurden, selbst wenn die Pakete in unterschiedlicher Reihenfolge ankommen. Dies kann leicht passieren, da die Pakete im Internet über unterschiedliche Router transportiert werden können. So kann ein Paket ein anderes überholen. UDP leistet diese Garantien nicht. Hier muss sich die Applikation im Zweifelsfall darum kümmern.

Um dies zu tun, versieht TCP jedes übertragene Byte mit einer Nummer. Dies ist die Sequenznummer. Der Empfänger muss den Empfang jedes Pakets bestätigen (Bestätigungsnummer/Acknowledgement-Nummer). Anhand der Nummer kann der Empfänger die Daten in der richtigen Reihenfolge zusammenbauen und erkennen, ob noch Daten fehlen.

Aus Sicherheitsgründen darf eine TCP-Verbindung nicht mit der Nummer 0 beginnend die Daten nummerieren. Damit nun dennoch beide Seiten erkennen können, wo die Übertragung anfängt, welche Daten übertragen wurden und welche noch fehlen, müssen sich die beiden Kommunikationspartner zu Beginn auf die verwendeten Sequenznummern einigen. Dies erfolgt im TCP-Handshake. Im ersten TCP-SYN-Paket überträgt der Client seine initiale Sequenznummer an den Server zur Synchronisation. Der Server bestätigt diese Nummer (ACK) in seinem ersten Paket und überträgt seine initiale Sequenznummer ebenfalls zur Synchronisation (SYN). Diese wird im dritten Paket nun vom Client bestätigt, und der TCP-Handshake ist erfolgreich durchgeführt worden.

INFO

TCP wird immer dann von Applikationen eingesetzt, wenn größere Datenmengen transportiert werden müssen. UDP ist ein typisches Protokoll, wenn die Datenmengen sehr klein sind, sodass sie in ein Paket passen oder die Daten zeitkritisch übertragen werden müssen. Genügt ein Paket für den Transport, so ist die Reihenfolge der Ankunft unerheblich. Ob das Paket angekommen ist, kann der Client an der fehlenden Antwort des Servers erkennen. Sind die Daten zeitkritisch, so schadet die Verzögerung durch die erneute Sendung der Daten möglicherweise mehr, als wenn diese Daten übersprungen werden. Schließlich ist es in bestimmten Umgebungen sehr unwahrscheinlich, dass Daten verloren gehen oder die Reihenfolge durcheinander gerät. Im lokalen LAN wird daher häufig tFTP für den Datentransport mit Routern eingesetzt. Dieses sehr einfache Protokoll verwendet ebenfalls UDP.

Das TCP-Protokoll benötigt einen TCP-Handshake für den Aufbau einer Verbindung. Ansonsten unterscheidet es sich aus Sicht eines Paketfilters nicht von dem UDP-Protokoll (siehe Abschnitt 42.5).

Abbildung 5.7: **Der Webbrowser fragt den Webserver mit dem TCP-Protokoll auf Port 80 nach einer Webseite.**

Listing 5.6: **TCPDump-Mitschnitt der Web-Anfrage**

```
[root@bibo root]# tcpdump -ni eth1 host www.nohup.info and port 80
tcpdump: verbose output suppressed, use -v or -vv for full protocol ↵
        decode listening on eth1, link-type EN10MB (Ethernet), capture ↵
        size 96 bytes
12:57:44.543981 IP 192.168.0.112.39305 > 217.160.128.61.http: S ↵
        795473263:795473 263(0) win 5840 <mss 1460,sackOK,timestamp ↵
        24325838 0,nop,wscale 7>
12:57:44.568941 IP 217.160.128.61.http > 192.168.0.112.39305: S ↵
        3041076941:30410 76941(0) ack 795473264 win 5792 <mss 1452, ↵
        sackOK,timestamp 53796879 24325838,nop ,wscale 0>
12:57:44.568972 IP 192.168.0.112.39305 > 217.160.128.61.http: . ack 1 win ↵
         46 <no p,nop,timestamp 24325863 53796879>
12:57:44.570825 IP 192.168.0.112.39305 > 217.160.128.61.http: P ↵
        1:130(129) ack 1 win 46 <nop,nop,timestamp 24325865 53796879>
12:57:44.600939 IP 217.160.128.61.http > 192.168.0.112.39305: . ack 130 ↵
        win 6432 <nop,nop,timestamp 53796882 24325865>
... gekürzt ...
12:57:45.627324 IP 192.168.0.112.39306 > 217.160.128.61.http: . ack 15511 ↵
         win 29 5 <nop,nop,timestamp 24326922 53796983>
12:57:45.628648 IP 217.160.128.61.http > 192.168.0.112.39306: F ↵
        15511:15511(0) a ck 140 win 6432 <nop,nop,timestamp 53796983 ↵
        24326882>
12:57:45.629519 IP 192.168.0.112.39306 > 217.160.128.61.http: F ↵
        140:140(0) ack 1 5512 win 295 <nop,nop,timestamp 24326924 ↵
        53796983>
12:57:45.658078 IP 217.160.128.61.http > 192.168.0.112.39306: . ack 141 ↵
        win 6432 <nop,nop,timestamp 53796988 24326924>
```

Die Regeln für diese TCP-Verbindung sind zunächst den UDP-Regeln sehr ähnlich. Lediglich der Zielport auf dem Server und das Protokoll sind angepasst worden.

```
CLIENTS=192.168.0.0/24
ipchains -A forward -p tcp -s $CLIENTS 1024:65535 -d any/0 80 -j ACCEPT
ipchains -A forward -p tcp -s any/0 80 -d $CLIENTS 1024:65535 -j ACCEPT
```

Einfache Paketfilter wie IPchains können aber bei einer TCP-Verbindung auch noch das erste Paket erkennen. Dieses Paket zeichnet sich dadurch aus, dass von allen TCP-Flags (siehe Abschnitt 42.12) nur das SYN-Flag für die Sequenznummersynchronisation gesetzt ist. Das erste Antwortpaket des Servers enthält bereits zusätzlich das gesetzte ACK-Bit. Da diese Pakete nur von dem Client zum Server und nicht in umgekehrter Richtung transportiert werden dürfen, kann dies in den Regeln berücksichtigt werden. IPchains bietet hierfür die Option `-y, --syn`. Diese Option erkennt das erste SYN-Paket. Als Negation `! --syn` erkennt diese Option alle Pakete außer dem ersten SYN-Paket. Die Regeln lassen sich also folgendermaßen ändern:

```
CLIENTS=192.168.0.0/24
ipchains -A forward -p tcp -s $CLIENTS 1024:65535 -d any/0 80 -j ACCEPT
ipchains -A forward -p tcp -s any/0 80 -d $CLIENTS 1024:65535 ! --syn -j ↵
        ACCEPT
```

Damit wird der Empfang eines SYN-Pakets von außen nach innen nicht erlaubt. Dennoch ist es möglich, dass ein Angreifer aus einer Verbindung herausgelöste TCP-ACK- oder TCP-FIN-Pakete senden kann. Hiermit kann er zwar keine Verbindung aufbauen, aber dennoch möglicherweise wertvolle Informationen sammeln (siehe Abschnitt 15.3.3).

Bei einer FTP-Verbindung wird die Konfiguration der Firewall komplizierter und unsicherer. Das FTP-Protokoll verwendet mehrere TCP-Verbindungen, um die Informationen zu übertragen (siehe auch Abschnitt 32.10). Zunächst baut der Client ausgehend von einem unprivilegierten Port eine TCP-Verbindung zum Port 21 auf. Dies ist die Steuerungsverbindung (Control Connection). Diese Verbindung wird genutzt, um die Anmeldeinformationen und alle Befehle zu übertragen. Sobald aber der Server Daten zum Client übertragen muss, wird eine zusätzliche Datenverbindung geöffnet.

Für diese Datenverbindung gibt es zwei Möglichkeiten. Im klassischen Fall des aktiven FTPs wird diese Verbindung von dem Server von Port 20 zu einem unprivilegierten Port des Clients aufgebaut (siehe Abbildung 5.8). Im Fall des passiven FTPs wird die Verbindung von dem Client ausgehend von einem beliebigen unprivilegierten Port zu einem – vom Server vorher bestimmten – unprivilegierten Port des Servers aufgebaut.

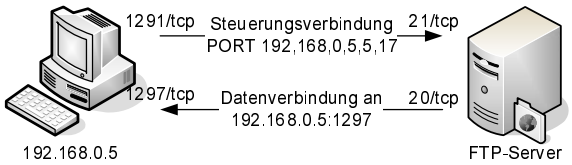


Abbildung 5.8: **Bei dem aktiven FTP baut der Server eine TCP-Verbindung zum Client auf.**

Aktives FTP durch einen IPchains-Paketfilter zu erlauben, heißt Türen und Tore öffnen. Der IPchains-Paketfilter muss den Aufbau einer TCP-Verbindung von außen erlauben. Die einzigen Einschränkungen betreffen den Quellport (20) und den Zielport (1024–65535). Da dies inakzeptabel ist, wird aktives FTP von Administratoren, die Paketfilter wie IPchains einsetzen, meist nicht erlaubt.

Passives FTP ist nicht ganz so kritisch, da die Verbindungen von innen aufgebaut werden. Jedoch kann der Firewall-Administrator nicht das Ziel der Verbindung beschränken. Der Client kann eine Verbindung zu einem beliebigen unprivilegierten Port im Internet aufbauen. Filesharing-Protokolle wie Torrent etc. nutzen auch derartige Verbindungen!

Wieso ist die Filterung so problematisch? Wenn der Paketfilter in der Lage wäre, einen Zusammenhang zwischen den Client- und den Serverpaketen und der Steuerungsverbindung und der Datenverbindung aufzubauen, würden viele Probleme entfallen!

5.5.4 IPchains und ICMP

Schließlich gibt es noch ein drittes sehr wichtiges Protokoll, das ein Paketfilter betrachten kann: das Internet Control Message Protocol (ICMP). Dieses Protokoll wird zum einen von dem Ping-Befehl verwendet und ist zum anderen für die Übertragung von Fehlerzuständen im Internet verantwortlich. Dieses Protokoll kennt keine Ports wie TCP oder UDP, sondern nur Typen und Codes.

Betrachten wir zunächst den Ping-Befehl. Dieser sendet, wie eine Client-Applikation, ein ICMP-Echo-Request-Paket an einen entfernten Rechner. Dieser Rechner antwortet, wie ein Server, mit einem ICMP-Echo-Reply-Paket.

```
[root@bibo root]# tcpdump -ni eth1 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol ↵
        decode listening on eth1, link-type EN10MB (Ethernet), capture ↵
        size 96 bytes
14:09:36.849952 IP 192.168.0.112 > 217.160.128.61: icmp 64: echo request ↵
        seq 0
14:09:36.875897 IP 217.160.128.61 > 192.168.0.112: icmp 64: echo reply ↵
        seq 0
```

Damit ein Client pingen kann, muss die Firewall dies erlauben. Hierzu muss sie die Echo-Request-Pakete nach außen und die Echo-Reply-Pakete nach innen passieren lassen.

ICMP wird aber auch verwendet, um Fehlermeldungen zu verschicken. Hierbei handelt es sich zum Beispiel um Fehlermeldungen, die anzeigen, dass das Ziel nicht erreichbar ist. Ist das Ziel nicht erreichbar, kann das Paket nicht zugestellt werden und wird verworfen. Mit dieser Fehlermeldung erhält der Absender die entsprechende Information. Diese Fehlermeldungen werden als Destination-Unreachable bezeichnet. Eine Filterung der Fehlermeldungen ist nur in Abhängigkeit des Typs und Codes möglich. Eine Prüfung, ob diese Fehlermeldung sich auf eine gültige Verbindung bezieht, kann von IPchains nicht durchgeführt werden!

5.5.5 IPchains und Masquerading

Einige erfahrene IPchains-Firewall-Administratoren werden bei den letzten Beispielen bereits innerlich auf die Masquerading-Funktion von IPchains hingewiesen haben. Hierbei handelt es sich um eine Art Network Address Translation, die in der Lage ist, den Zustand von Verbindungen zu überwachen und nur die Pakete durchzulassen, die Teil einer bestimmten Verbindung sind.

Das ist natürlich richtig. Wenn IPchains mit NAT eingesetzt wurde, konnte so die Sicherheit stark erhöht werden. Sobald aber auf NAT verzichtet wurde, tauchten die oben erwähnten Probleme auf. Der Firewall-Administrator konnte diese Funktion auch nicht für die eigentliche Filterung der Pakete einsetzen, sondern musste auf die Intelligenz des Masquerading-Codes vertrauen.

5.5.6 IPtables und Conntrack

Mit `iptables` können Sie die oben aufgeführten IPchains-Regelsätze eins zu eins umsetzen und genau dieselben Sicherheitsprobleme erzeugen, die ich oben erläutert habe. Sie müssen lediglich die angepasste Syntax bei Iptables gegenüber IPchains beachten. Sie können aber auch durch Nutzung des Connection Tracking (es wird auch State Machine genannt) wesentlich intelligenter und sicherer filtern. Dieser Abschnitt zeigt Ihnen, wie das geht und welche Möglichkeiten sich Ihnen eröffnen. In Kapitel 19 finden Sie weitere Hinweise zum Tuning der Verbindungsüberwachung.

Iptables verfügt über eine Verbindungsüberwachung, die Sie für die Filterung von Verbindungen einsetzen können. Während ein normaler Paketfilter Pakete filtert, kann Iptables bei dem Einsatz der Verbindungsüberwachung Verbindungen filtern! Sie akzeptieren ein Paket einer Netzwerkverbindung, und automatisch werden alle weiteren Pakete, die Teil dieser Verbindung sind, auch akzeptiert! Dabei erkennt das System automatisch und fehlerfrei alle weiteren Pakete, auch die Antwortpakete des Servers.

Wie funktioniert das? Das Conntrack-System des Linux-Kernels pflegt intern eine Tabelle aller bekannten Verbindungen. Dieses Conntrack-System kann sowohl als Modul geladen werden als auch fest im Linux-Kernel einkompiliert sein. Allerdings ermöglicht das Laden des Conntrack-Systems zusätzliches Tuning über Optionen (siehe Kapitel 19). Sie laden das Modul mit dem Befehl `modprobe nf_conntrack`.[4] Auf älteren System lautet der Name des Moduls `ip_conntrack`.

Sobald nun ein neues Paket den Rechner erreicht, klassifiziert das Conntrack-System dieses Paket als `NEW`, `ESTABLISHED`, `RELATED` oder `INVALID`.[5] Diese Klassifizierung erfolgt übrigens in der `PREROUTING`-Kette der `NAT`-Tabelle (siehe Abschnitt 5.7) für Pakete, die von außen kom-

4 Sobald Sie das Conntrack-System laden, werden alle Pakete von dem Conntrack-System automatisch defragmentiert. Während Sie die Defragmentierung bis zum Linux-Kernel 2.2 ein- und abschalten konnten, ist dies nun nicht mehr möglich (siehe auch Abschnitt 35.3).

5 Wenn der Kernel die Unterstützung für die Raw-Tabelle besitzt, ist auch noch der Zustand `UNTRACKED` verfügbar.

men, und in der OUTPUT-Kette der NAT-Tabelle für lokal generierte Pakete. Hierzu betrachtet das Conntrack-System das Paket und vergleicht es mit den bereits in der Verbindungstabelle befindlichen Verbindungen. Den aktuellen Stand der Liste können Sie sich mit dem Befehl `cat /proc/net/nf_conntrack` anzeigen lassen. Falls sich bereits Verbindungen in dieser Tabelle befinden, werden sie der folgenden ähneln.

Listing 5.7: **Eine TCP-Verbindung in /proc/net/ipconntrack**

```
tcp      6 431995 ESTABLISHED src=192.168.0.112 dst=205.156.51.200 sport ↵
         =36188 dport=80 src=205.156.51.200 dst=192.168.0.112 sport=80 ↵
         dport=36188 [ASSURED] use=1
```

Eine aufbereitete Darstellung (siehe Abbildung 5.9) der Tabelle kann mit dem Befehl `iptstate` angezeigt werden (siehe auch Abschnitt 11.3).

Das Conntrack-System vergleicht nun das Transportprotokoll (TCP, UDP, ICMP etc.), die IP-Adressen und – wenn wie bei TCP und UDP vorhanden – die Ports der Einträge mit dem Paket. Das Paket erhält dann einen der folgenden vier Zustände:

» NEW. Das Paket erhält den Zustand NEW, wenn ein Paket mit dieser Kombination aus Protokoll, IP-Adressen und Ports in der Tabelle noch nicht existiert. Diese neue Verbindung wird aber nicht automatisch in der Tabelle eingetragen.

» ESTABLISHED. Das Paket erhält den Zustand ESTABLISHED, wenn es mit seinem Protokoll, den IP-Adressen und den Ports genau auf eine Verbindung in der Tabelle passt. Dabei ist es einerlei, in welcher Richtung das Paket transportiert wird. Handelt es sich um das erste derartige Paket, wird gleichzeitig der Zustand der Verbindung in der Tabelle aktualisiert, sodass diese Verbindung nun den Zustand [ASSURED] hat.

» RELATED. Das Paket erhält den Zustand RELATED, wenn es in einer besonderen Beziehung zu einer bereits in der Tabelle vorhandenen Verbindung steht. Hierbei handelt es sich typischerweise um ICMP-Fehlermeldungen, die sich auf eine aufgebaute Verbindung beziehen. Diese können nicht den Zustand ESTABLISHED erhalten, da sie bereits ein anderes

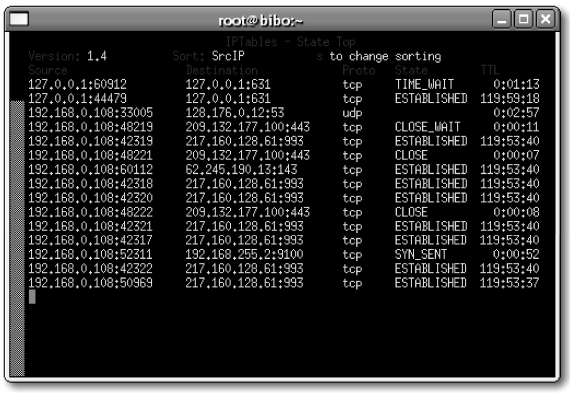


Abbildung 5.9: **Der Befehl iptstate zeigt die Verbindung ähnlich dem Befehl top an.**

Protokoll (ICMP) verwenden als die in der Tabelle eingetragene Verbindung. Das Conntrack-System kann jedoch erkennen, auf welche Verbindung sich eine ICMP-Fehlermeldung bezieht. Existiert diese Verbindung in der Tabelle, bekommt das Paket der ICMP-Fehlermeldung den Zustand RELATED. Außerdem werden auch FTP-Datenverbindungen unter Umständen als RELATED angesehen (siehe unten).

» `INVALID`. Dass ein Paket den Zustand INVALID erhält, kann zwei Ursachen haben. Entweder verfügt das System nicht über ausreichend Ressourcen, um den Zustand zu ermitteln und zu speichern, oder es handelt sich um eine ICMP-Fehlermeldung, die sich auf eine nicht existente Verbindung bezieht.

INFO

Maximale Anzahl der Verbindungen in der Tabelle.
Die Größe der Tabelle zur Verbindungsüberwachung ist natürlich begrenzt. Allerdings können Sie die Größe dieser Tabelle, wie auch viele andere Parameter des Connection Tracking, anpassen und tunen. Dies wird aber in einem eigenen Kapitel (siehe Kapitel 19) beschrieben.

Mithilfe dieses Systems ist es nun möglich, komplett auf die löchrigen Regeln zu verzichten, die mögliche Antwortpakete für aufgebaute Verbindungen zulassen. Um dies zu verdeutlichen, sollen die oben aufgeführten Beispiele nun mit Iptables und Conntrack nachvollzogen werden.

5.5.7 Conntrack und UDP

Wir beginnen wieder mit dem Beispiel einer DNS-Verbindung (siehe Abbildung 5.10). Hier sendet der Client ein Paket an den DNS-Server, und der DNS-Server antwortet mit einem Paket.

Wenn Sie einem Client durch Ihren Paketfilter den Zugriff auf einen beliebigen DNS-Server erlauben möchten, benötigen Sie folgende Regel:

Listing 5.8: **Der Client baut mit seiner DNS-Anfrage eine neue UDP-Verbindung auf.**

```
CLIENTS=192.168.0.0/24
iptables -A FORWARD -s $CLIENTS -p udp --dport 53 -m state --state NEW -j ↵
         ACCEPT
```

Diese Regel erlaubt (`-j ACCEPT`) es einem Client (`-s $CLIENTS`), ein UDP-Paket (`-p udp`) an den Zielport 53 (`--dport 53`) zu schicken, wenn die Zustandsmaschine (`-m state`) dieses Paket als neu kategorisiert (`--state NEW`). Zusätzlich merkt sich die Zustandsmaschine

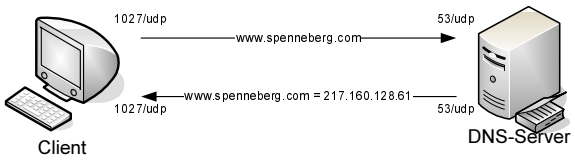


Abbildung 5.10: **Der DNS-Client fragt den DNS-Server mit dem UDP-Protokoll auf Port 53 nach der IP-Adresse für** `www.spenneberg.com`.

diese Verbindung nun in ihrer Tabelle. Um die Antwort des DNS-Servers auch zu erlauben, benötigen Sie die folgende Regel:

Listing 5.9: **Akzeptiere alle weiteren Pakete in der Verbindung.**

```
iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
```

Diese Regel prüft, ob es sich bei dem Paket um ein Paket einer Verbindung handelt, die bereits erlaubt und aufgebaut wurde (`--state ESTABLISHED`). Ist dies der Fall, wird auch dieses Paket akzeptiert (`-j ACCEPT`). Es ist hier nicht nötig, erneut den Port zu testen. Verwendet das Paket einen anderen Port als 53/udp, dann gehört es nicht zu der aufgebauten Verbindung und wird daher verworfen.

Wie stellt sich das in der Verbindungstabelle dar? Nachdem das erste Paket von dem Client durch die Firewall an den Server geschickt wurde, wird die Zustandsmaschine folgenden Eintrag in der Zustandstabelle protokollieren:

```
udp      17 28 src=10.0.2.100 dst=5.255.255.254 sport=1024 dport=53 [ ↵
         UNREPLIED] src=5.255.255.254 dst=10.0.2.100 sport=53 dport=1024  ↵
         use=1 mark=0
```

Die Angabe [UNREPLIED] zeigt an, dass die Zustandsmaschine noch keine Antwort für dieses Paket verarbeitet hat. Die weiteren Angaben des Eintrags sind das Protokoll (udp) und die Protokollnummer (17). Anschließend wird die Verweildauer des Eintrags in der Tabelle in Sekunden (28) angegeben. Nun folgen die Absender- und die Zieladresse sowie der Absender- und der Zielport. Eine genauere Erklärung der Angaben finden Sie in Kapitel 19.

Sobald nun von dem Server eine Antwort geschickt wird, ändert sich der Zustand in der Tabelle:

Listing 5.10: **Die Einträge der Zustandstabelle können über die Datei /proc/net/nf_conntrack betrachtet werden.**

```
udp      17 20 src=10.0.2.100 dst=5.255.255.254 sport=1024 dport=53 src ↵
         =5.255.255.254 dst=10.0.2.100 sport=53 dport=1024 use=1 mark=0
```

Nun ist der Status [UNREPLIED] nicht mehr vorhanden. Die Zustandsmaschine hat für diese Verbindung Pakete in beiden Richtungen gesehen und zugelassen.

Ein aufmerksamer Leser mag sich nun fragen, wie lange die UDP-Verbindung in der Tabelle verbleibt. Da das UDP-Protokoll keinen Verbindungsabbau kennt, kann die Zustandsmaschine dies nur über Zeitgeber lösen. Ein Standardkernel erlaubt eine UDP-Verbindung im Zustand UNREPLIED für 30 Sekunden in der Tabelle. Sobald die Zustandsmaschine Antwortpakete verarbeitet hat, bleibt die Verbindung für 180 Sekunden in der Zustandstabelle. Jedes weitere Paket dieser Verbindung setzt diese Zähler wieder auf ihre Ausgangswerte zurück.

5.5.8 Conntrack und TCP

Bei dem TCP-Protokoll wollen wir auch zunächst wieder eine HTTP-Verbindung betrachten. In Abbildung 5.11 ist noch einmal die HTTP-Verbindung schematisch gezeigt.

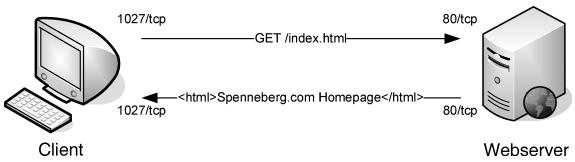


Abbildung 5.11: **Der Webbrowser fragt den Webserver mit dem TCP-Protokoll auf Port 80 nach einer Webseite.**

Wenn Sie nun einem Client durch Ihren Paketfilter den Zugriff auf einen beliebigen Webserver im internen Netz erlauben möchten, können Sie die Regeln, die wir für den DNS-Server aufgestellt haben, fast unverändert übernehmen. Zunächst benötigen Sie auch eine Regel, die den Aufbau der Verbindung von innen nach außen erlaubt:

```
CLIENTS=192.168.0.0/24
iptables -A FORWARD -s $CLIENTS -p tcp --dport 80 -m state --state NEW -j ↵
         ACCEPT
```

Diese Regel erlaubt es dem ersten Paket einer jeden HTTP-Verbindung, von innen nach außen zu gelangen. Dieses erste Paket wird von der Zustandsmaschine als NEW kategorisiert.[6]

Damit alle weiteren Pakete der Verbindung zugelassen werden, ist eine zweite Regel erforderlich:

```
iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
```

Aufmerksame Leser werden feststellen, dass sich diese Regel nicht von der entsprechenden Regel für die UDP-Verbindungen unterscheidet. Sobald nun ein Paket von dieser Regel bearbeitet wird, das zu einer Verbindung gehört, die sich bereits in der Zustandstabelle befindet, wird das Paket akzeptiert. Es genügt als eine einzige Regel, um alle weiteren Pakete aufgebauter Verbindungen zu akzeptieren! Während Sie bei `ipchains` jeweils Regeln für die Rückrichtungen brauchten, genügt hier eine universelle Regel! Die Anzahl der Regeln ist daher bei Iptables wesentlich kleiner.

Auch die TCP-Verbindungen werden in der Zustandstabelle gepflegt und können über die Datei `/proc/net/nf_conntrack` betrachtet werden. Allerdings besitzen diese Einträge auch weitere Zustände, die daher hier kurz vorgestellt werden sollen. Insgesamt kennt die Zustandsmaschine acht verschiedene Zustände einer TCP-Verbindung, die hier aber nicht alle gezeigt werden können. Sie finden weitere Informationen in dem Kapitel 19.

Sobald der Client sein erstes Paket schickt, befindet sich in der Zustandstabelle folgender Eintrag:

6 Die Zustandsmaschine macht hier aber noch zusätzliche Einschränkungen: Damit ein TCP-Paket den Zustand NEW haben kann, muss es sich um ein Paket handeln, das entweder nur das TCP-SYN-Flag oder das TCP-ACK-Flag oder kein Flag gesetzt hat. Die Zustand der TCP-Flags URG und PSH ist hierbei unerheblich.

```
tcp      6 115 SYN_SENT src=10.0.2.100 dst=5.255.255.254 sport=1024 dport ↵
         =80 [UNREPLIED] src=5.255.255.254 dst=10.0.2.100 sport=80 dport ↵
         =1024 use=1 mark=0
```

Normalerweise sehen Sie diesen Eintrag gar nicht, da der Server sofort antwortet! Hier hat die Verbindung aber noch den Zustand [UNREPLIED]. Zusätzlich sehen Sie den Zustand `SYN_SENT`. Dies ist eine zusätzliche Information über die Verbindung.

Sobald der TCP-Handshake komplett vollzogen wurde, wechselt die Verbindung ihren Zustand:

```
tcp      6 431996 ESTABLISHED src=10.0.2.100 dst=5.255.255.254 sport=1024  ↵
         dport=80 src=5.255.255.254 dst=10.0.2.100 sport=80 dport=1024 [ ↵
         ASSURED] use=1 mark=0
```

Da nun Pakete in beiden Richtungen gesehen wurden, erhält die Verbindung den Zustand [ASSURED] und zusätzlich die Information ESTABLISHED.

Sobald ein Client nun ein TCP-FIN-Paket für den Abbau der Verbindung schickt, wechselt wieder der Zustand (siehe auch Abschnitt 19.4):

```
tcp      6 119 FIN_WAIT src=10.0.2.100 dst=5.255.255.254 sport=1024 dport ↵
         =80 src=5.255.255.254 dst=10.0.2.100 sport=80 dport=1024 [ ↵
         ASSURED] use=1 mark=0
```

Die Verbindung bleibt [ASSURED], hat aber nun zusätzlich den Zustand `FIN_WAIT`. Sobald nun auch die Gegenseite ihr FIN gesendet hat, wechselt die Verbindung abermals den Zustand:

```
tcp      6 119 TIME_WAIT src=10.0.2.100 dst=5.255.255.254 sport=1024 dport ↵
         =80 src=5.255.255.254 dst=10.0.2.100 sport=80 dport=1024 [ ↵
         ASSURED] use=1 mark=0
```

Diese verschiedenen Zustände sind wichtig, da die Zustandsmaschine diesen verschiedenen Zuständen unterschiedliche Verweildauern in der Zustandstabelle einräumt. So hat der Zustand `SYN_SENT` zum Beispiel eine Verweildauer von 2 Minuten in der Zustandstabelle. Das bedeutet, dass innerhalb von 2 Minuten das nächste Paket der Verbindung die Firewall erreichen muss, damit die Verbindung nicht aus der Tabelle entfernt wird. Der Zustand ESTABLISHED hat eine Verweildauer von 5 Tagen!

Je nach verwendetem Kernel können Sie diese Zeiten anpassen. Die weiteren Zeiten und ihre Anpassung werden in Kapitel 19 erklärt.

Erinnern Sie sich noch an das FTP-Protokoll und die Probleme, dieses Protokoll mit IPchains zu filtern? Nun, die Zustandsmaschine bietet auch hier eine geniale Lösung an. Bei dem FTP-Protokoll handeln der Client und der Server dynamische Ports für den Transport der Daten aus. Diese Ports können daher nicht von vornherein in den Regeln berücksichtigt werden. Jedoch werden diese Informationen ja in dem Moment, in dem sie benötigt werden, live übertragen. Die Zustandsmaschine besitzt speziellen Code, der diese Informationen auslesen und verwerten kann. Dies ist das `nf_conntrack_ftp`-Kernelmodul. Dieser Code kann als Modul

(was sich empfiehlt) oder fest in den Kernel eingebaut werden. Sobald dieses Modul geladen (oder fest eingebaut) wurde, ist es aktiv. Es überwacht dann alle Pakete, die an den Port 21 (FTP) gehen, und analysiert diese. Sobald eine Datenverbindung über die Steuerungsverbindung angekündigt wird, extrahiert es die Daten und trägt die resultierende zu erwartende Datenverbindung in der Zustandstabelle ein. Alle Pakete, die anschließend auf diese Verbindung passen, erhalten den Zustand `RELATED`. Wenn Sie also FTP-Verbindungen (aktiv wie passiv) sicher durch Ihre Firewall erlauben wollen, genügen die folgenden Befehle:

```
modprobe nf_conntrack_ftp
iptables -A FORWARD -s $CLIENTS -p tcp --dport 21 -m state --state NEW -j ↵
         ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Die erste Zeile lädt das Modul zur Erkennung der Datenverbindungen.[7] Die zweite Zeile erlaubt den Aufbau der FTP-Steuerungsverbindung. Die dritte Zeile schließlich erlaubt alle Pakete, die zur FTP-Steuerungsverbindung gehören, und alle von der Zustandsmaschine erkannten FTP-Datenverbindungen. So können Sie sicher FTP-Verkehr filtern!

5.5.9 Conntrack und ICMP

Kommen wir nun zum dritten sehr wichtigen Protokoll bei der Definition von Firewall-Regeln, zum ICMP-Protokoll. Wie ich bereits gesagt habe, wird dieses Protokoll sowohl für die Meldung von Übertragungsfehlern als auch für den Ping-Befehl genutzt. Beginnen wir mit dem `ping`-Befehl. Erfreulicherweise betrachtet die Zustandsmaschine die Pakete, die von dem Ping-Befehl erzeugt werden, wie die Pakete, die zwischen einem Client und einem Server ausgetauscht werden. Eine Ping-Anfrage (ICMP-Echo-Request) baut eine Verbindung auf und erhält den Zustand `NEW`. Eine Ping-Antwort (ICMP-Echo-Reply) wird von der Zustandsmaschine als `ESTABLISHED` eingeordnet und entfernt auch sofort die Verbindung wieder aus der Tabelle. Das bedeutet, dass für jedes ausgesandte Echo-Request-Paket nur ein, und zwar das richtige Echo-Reply-Paket zugelassen wird. Die Zuordnung der einzelnen Pakete erfolgt über die ICMP-Sequenznummer.

Ausgehend von unserem Firewall-Szenario benötigen Sie die folgende Regel, um nun das Echo-Request-Paket zuzulassen:

```
CLIENTS=192.168.0.0/24
iptables -A FORWARD -s $CLIENTS -p icmp --icmp-type echo-request -m state ↵
         --state NEW -j ACCEPT
```

7 Wenn Sie einen ungewöhnlichen FTP-Server verwenden, der nicht auf dem Port 21, sondern auf einem anderen Port Verbindungen entgegennimmt, wird das Modul die Verbindung nicht erkennen. Hierfür können Sie beim Laden des Moduls eine Liste von bis zu 8 Ports angeben (mit dem Parameter `ports`), die das Modul betrachten soll. Dies ist nicht möglich, wenn der Code fest in den Kernel integriert wurde!

Diese Regel lässt ein neues Paket (`-m state --state NEW`) eines Clients (`-s $CLIENTS`) zu (`-j ACCEPT`), wenn es das Protokoll ICMP (`-p icmp`) und den ICMP-Typ Echo-Request (`--icmp-type echo-request`) verwendet.

Nun benötigen Sie noch eine Regel, die die Antwortpakete erlaubt. Hier genügt wieder die bereits oben vorgestellte allgemeine Regel, die alle Pakete mit dem Zustand `ESTABLISHED` erlaubt:

```
iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
```

Wie sieht das nun in der Zustandstabelle aus? Nachdem die Firewall das ICMP-Echo-Request-Paket gesehen und verarbeitet hat, können Sie den folgenden Eintrag in der Zustandstabelle beobachten:

```
icmp     1 29 src=10.0.2.100 dst=5.255.255.254 type=8 code=0 id=31489 [ ↵
         UNREPLIED] src=5.255.255.254 dst=10.0.2.100 type=0 code=0 id ↵
         =31489 use=1 mark=0
```

In den meisten Fällen werden Sie diesen Eintrag nie sehen, da er direkt gelöscht wird, sobald die Zustandsmaschine das ICMP-Echo-Reply-Paket sieht. So können Sie Ping-Pakete zustandsorientiert filtern. Sie müssen nicht grundsätzlich alle Reply-Pakete durchlassen wie bei IPchains.

Die Verbindung verbleibt übrigens im Zustand UNREPLIED für maximal 30 Sekunden in der Verbindungstabelle.

Wie behandelt die Zustandsmaschine nun ICMP-Fehlermeldungen? Erfreulicherweise erkennt die Zustandsmaschine die Fehlermeldungen und kann diese den Verbindungen in der Zustandstabelle zuordnen. Dabei erkennt sie, ob sich die Fehlermeldung auf eine der Verbindungen in der Tabelle bezieht oder nicht. Ist das der Fall, so erhält das Paket den Zustand `RELATED`. Ist das nicht der Fall, so ist das Paket `INVALID`. So können Sie alle Fehlermeldungen sicher mit einer Regel filtern:

```
iptables -A FORWARD -m state --state RELATED -j ACCEPT
```

Diese Regel akzeptiert auch FTP-Datenverbindungen vom Zustand RELATED. Wenn Sie das nicht möchten, können Sie das auch für ICMP-Pakete einschränken:

```
iptables -A FORWARD -p icmp -m state --state RELATED -j ACCEPT
```

Sobald die Zustandsmaschine übrigens eine Fehlermeldung erkennt und zuordnen kann, führt dies auch dazu, dass die betroffene Verbindung aus der Zustandstabelle entfernt wird!

5.5.10 Conntrack und alle weiteren generischen Protokolle

Alle weiteren IP-Protokolle, wie IGMP, GRE, ESP und so weiter, können auch von Conntrack überwacht werden.

Da jedoch diese Protokolle weder über Ports noch über ICMP-IDs verfügen, ist Conntrack nicht in der Lage, gleichzeitig mehrere derartige Verbindungen zu überwachen. Für das PPTP-Protokoll gibt es hier einen besonderen Patch, der dies doch wieder ermöglicht.

Ansonsten erfolgt die Filterung der generischen Protokolle ähnlich dem UDP-Protokoll. Das erste Paket erhält den Status NEW, während alle weiteren Pakete den Status ESTABLISHED erhalten. Um zum Beispiel das ESP-Protokoll zustandsorientiert zu filtern, verwenden Sie die folgenden Regeln:

```
CLIENTS=192.168.0.0/24
iptables -A FORWARD -s $CLIENTS -p 50 -m state --state NEW -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
```

Das ESP-Protokoll ist das IP-Protokoll 50 (siehe `/etc/protocols`). Die erste Iptables-Regel akzeptiert jedes neue Paket eines Clients. Die zweite Regel akzeptiert alle Pakete, die zu dieser aufgebauten Verbindung gehören.

Die zustandsorientierte Filterung dieser Protokolle ist nicht einfach, da Conntrack keine Informationen über den tatsächlichen Zustand der Verbindung extrahieren kann. Alle Pakete werden gleichwertig behandelt. Zur Überwachung des Zustands der Verbindung werden lediglich Zeitgeber eingesetzt, die entscheiden, ob eine Verbindung beendet wurde oder nicht (siehe Kapitel 19). Dies führt häufig zu Problemen. Daher sollten Sie überlegen, ob Sie diese Protokolle besser zustandslos filtern. Dies ist mit den folgenden zwei Zeilen möglich:

```
CLIENTS=192.168.0.0/24
iptables -A FORWARD -s $CLIENTS -p 50 -j ACCEPT iptables -A FORWARD -d ↵
        $CLIENTS -p 50 -j ACCEPT
```

5.6 Filterregeln

Sie haben bereits in den Abschnitten 5.2 und 5.3 erste Filterregeln erzeugt. Dieses Thema soll in diesem Kapitel vertieft werden. Während Sie in Abschnitt 5.2 nur einzelne Regeln zum Test der Funktionalität erzeugt haben, ist in Abschnitt 5.3 bereits ein funktionsfähiges Firewall-Skript entstanden. Dieses wies jedoch lediglich Regeln auf, die Verbindungen durch die Firewall von innen nach außen erlaubten. Hier wurden dann aber alle denkbaren Verbindungen zugelassen. Weitere Verbindungen direkt auf dem Gateway wurden von dem Skript nicht geduldet. Für den ersten Start ist dies meist ausreichend, anschließend aber häufig vollkommen indiskutabel.

Was fehlt? Nun, in vielen Fällen sind zusätzliche Verbindungen erwünscht, die von der Firewall abgehen oder auf der Firewall ankommen, da aus Budget-Gründen auf der Firewall zusätzlich zum Beispiel ein Squid-Proxy oder ein Postfix-Mailserver installiert werden sollen. Außerdem ist es für viele Firmen indiskutabel, alle Verbindungen von innen nach außen zu erlauben. Meist existiert in den Firmen bereits eine Benutzerrichtlinie, die genau festlegt, welche Dienste die Benutzer bei dem Zugriff auf das Internet nutzen dürfen und zu welchem Zweck.

Es ist dann sinnvoll, bei der Definition der Regeln nur diese Dienste zu erlauben, um so die Benutzer vor sich selbst zu schützen.

Im Folgenden sollen diese Regeln vorgestellt und soll das Skript aus Abschnitt 5.3 entsprechend angepasst werden.

Zunächst wollen wir die Regeln betrachten, die notwendig sind, um Dienste direkt auf der Firewall anzusprechen. Da die Firewall aber nicht ganz ohne Schutz geöffnet werden soll, ist es sinnvoll, diesen Zugriff nur auf wenige Dienste zu beschränken. Ein typischer Dienst, der häufig gewünscht wird, ist der Secure-Shell-Dienst (sshd), mit dem eine Fernadministration der Firewall möglich ist. Wenn Sie von innen auf diesen Dienst zugreifen möchten, benötigen Sie folgende Regel:

Listing 5.11: **Erlaube den Aufbau einer SSH-Verbindung von innen.**

```
$IPTABLES -A INPUT -s $CLIENTS -i $INTDEV -p tcp --dport 22 -m state --
        state NEW -j ACCEPT
```

Diese Regel wird an die `INPUT`-Kette angehängt. Diese Kette analysiert sämtliche Pakete, die von außen an die Firewall gesendet werden. Zur Erläuterung sind in Abbildung 5.12 noch einmal die Ketten der Filter-Tabelle dargestellt.

Wenn Sie diese Abbildung betrachten, werden Sie erkennen, dass sämtliche Pakete, die die Firewall von außen erreichen und an die Firewall gerichtet sind und nicht weitergeleitet werden, von der `INPUT`-Kette verarbeitet werden. Unsere neue Regel akzeptiert in der `INPUT`-Kette alle Pakete, die eine neue TCP-Verbindung (`-p tcp`) auf dem Port 22 (`--dport 22`) öffnen.

Nun müssen Sie auch die Antworten der Firewall an den SSH-Client zulassen. Hierzu ist eine entsprechende Regel in der `OUTPUT`-Kette erforderlich. Diese Kette analysiert alle Pakete, die auf der Firewall erzeugt werden und diese verlassen, also auch die Antwortpakete des Secure-Shell-Servers.

```
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Diese Aufgabe kann sehr leicht der Zustandskontrolle übertragen werden. Diese Regel akzeptiert in der `OUTPUT`-Kette alle Pakete, die zu erlaubten aufgebauten Verbindungen gehören

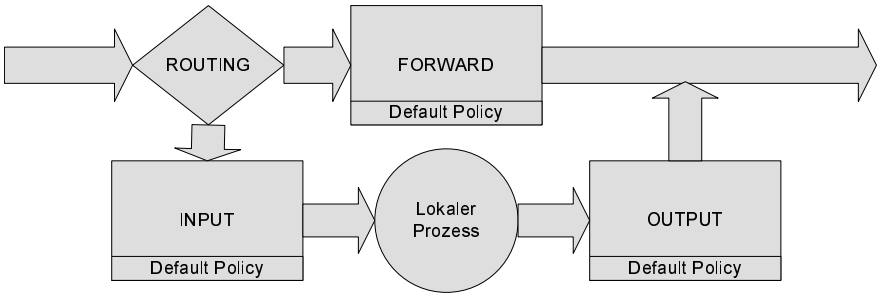


Abbildung 5.12: **Die Filtertabelle besteht aus drei Ketten.**

oder wie eine Fehlermeldung sich auf diese beziehen. Eine erneute Prüfung des Ports ist hier nicht erforderlich, denn nur Verbindungen, die wir vorher erlaubt haben, werden in die Verbindungstabelle aufgenommen.

Damit ist das Regelwerk aber noch nicht ganz komplett. Auch der SSH-Client wird weitere Pakete schicken, die nun nicht mehr den Zustand NEW, sondern den Zustand ESTABLISHED oder im Falle einer Fehlermeldung RELATED haben können. Um auch diese Pakete auf der Firewall zu erlauben, ist es erforderlich, dass auch in der INPUT-Kette diese Pakete erlaubt werden:

```
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Nach diesem Muster können Sie nun weitere Dienste hinzufügen. Möchten Sie zum Beispiel auch von außen auf den SSH-Server zugreifen, genügt nun die folgende zusätzliche Regel:

Listing 5.12: **Erlaube den Zugriff mit SSH von außen.**

```
$IPTABLES -A INPUT -i $EXTDEV -p tcp --dport 22 -m state --state NEW -j ↵
        ACCEPT
```

Möchten Sie einen Webserver auf der Firewall betreiben, der von außen erreichbar sein soll, so genügt nun die folgende Regel:

Listing 5.13: **Erlaube den Zugriff mit SSH von außen.**

```
$IPTABLES -A INPUT -i $EXTDEV -p tcp --dport 80 -m state --state NEW -j ↵
        ACCEPT
```

Ich halte es nicht für eine gute Idee, auf der Firewall gleichzeitig auch noch weitere Dienste, wie SSH und HTTP, anzubieten. Dennoch weiß ich, dass es in vielen Umgebungen zwingend erforderlich ist, da das Budget für dedizierte Systeme häufig fehlt. Dennoch sollten Sie auf einer Firewall nur die nötigen Dienste installieren und betreiben. Falls dies aus Kostengründen nicht möglich ist, sollten Sie mindestens über eine Virtualisierung nachdenken.

Auf der anderen Seite können Sie natürlich auf jedem Linux-Server so eine lokale Firewall installieren und nutzen! Dies wird aber noch in Kapitel 7 genauer besprochen.

5.7 Die NAT-Tabelle und -Regeln

Jetzt sehen wir uns die NAT-Tabelle an. In den letzten Abschnitten haben Sie bereits einige NAT-Regeln aufgesetzt. Dabei handelte es sich um Regeln, die die Absenderadresse von Paketen maskieren konnten. In diesem Abschnitt will ich nun alle Möglichkeiten der NAT-Tabelle vorstellen und kurz erläutern. Ausführlichere technische Hinweise finden Sie in Kapitel 20.

Die NAT-Tabelle hat insgesamt drei Ketten: OUTPUT, PREROUTING und POSTROUTING. Diese drei Ketten sind komplett eigenständige Ketten und sind nicht identisch mit den gleichnamigen Ketten der anderen Tabellen. Abbildung 5.13 zeigt, wie sich die Tabellen in die Filtertabelle einbinden.

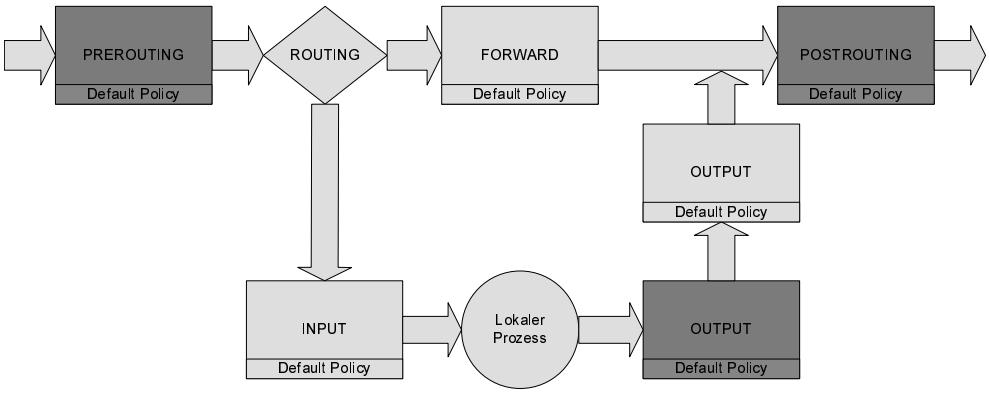


Abbildung 5.13: **Die NAT-Tabelle besteht aus drei Ketten.**

Bei der Network Address Translation (NAT) ändert das System die Adressierung eines Pakets. Es gibt grundsätzlich zwei verschiedene Arten der Network Address Translation:

» Source-NAT: Hier wird die Absender-Adresse geändert.

» Destination-NAT: Hier wird die Zieladresse geändert.

Ein Source-NAT ist nur in der `POSTROUTING`-Kette der NAT-Tabelle erlaubt. Ein Destination-NAT ist nur in der `PREROUTING`- und der `OUTPUT`-Kette erlaubt. Wenn Sie sich Abbildung 5.13 kurz anschauen, wird Ihnen auch sofort der Sinn klar.

Beim Source-NAT ändert die Firewall die Absenderadresse. Da das in der POSTROUTING-Kette erfolgt, ist sichergestellt, dass die Filterregeln in der FORWARD- und der OUTPUT-Kette das Paket vor der Adressänderung analysieren. Ihre Filterregeln betrachten also das originale Paket mit der „echten“ Absenderadresse.

Beim Destination-NAT ändert die Firewall die Zieladresse. Dies passiert in der PREROUTING- oder der OUTPUT-Kette der NAT-Tabelle und damit vor der Analyse durch die entsprechenden Filterketten. Bei diesem NAT wird die scheinbare Zieladresse durch eine neue tatsächliche Zieladresse ausgetauscht. Die Filtertabelle betrachtet nun also wieder das Paket mit der „echten“ Zieladresse!

Im Folgenden betrachten wir zunächst das Source-NAT und dann das Destination-NAT ein wenig genauer. Dabei werden Sie für beide Fälle auch ein paar Beispielregeln kennenlernen und in Ihrem Skript einbauen.

5.7.1 Source-NAT

Das Source-NAT ändert die Absender-Adresse eines Pakets. Dies ist häufig bei der Anbindung von Netzwerken an das Internet erforderlich. Lokale Netzwerke werden üblicherweise unter Zuhilfenahme von sogenannten privaten IP-Adressen aufgebaut. Hierbei handelt es sich um spezielle IP-Adressen, die im Internet nicht verwendet werden.

INFO

Private IP-Adressen nach RFC1918.
Bei dem Aufbau eines Netzes ist es zwingend erforderlich, dass die verwendeten IP-Adressen nicht doppelt verwendet werden, sondern eindeutig sind. Dies gilt auch, wenn mehrere Netze miteinander verbunden werden. Sobald Sie ein Netz mit dem Internet verbinden, müssen Sie auch hier die Eindeutigkeit der verwendeten IP-Adressen sicherstellen. Das bedeutet für Sie, dass Sie garantieren müssen, dass die von Ihnen verwendeten IP-Adressen nicht im Internet genutzt werden! Das ist unmöglich, solange Sie nicht wissen, dass bestimmte IP-Adressen im Internet „verboten" sind. Hierzu wurde der RFC1918 geschaffen. Sie können die RFCs im Internet übrigens auf der Webseite `http://www.rfc-editor.org` *nachlesen. Dieser RFC (mit dem Titel Address Allocation for Private Internets) beschreibt die privaten IP-Adressen. Die folgenden IP-Adressen werden als private IP-Adressen reserviert:*

- *10.0.0.0–10.255.255.255*
- *172.16.0.0–172.31.255.255*
- *192.168.0.0–192.168.255.255*

Sobald Sie Ihre eigenen Netze mit diesen IP-Adressen aufbauen, erzeugen Sie bei einer Anbindung an das Internet keinen IP-Adresskonflikt. Für kleine Netze ist der Bereich 192.168.0.0/24 üblich.

Wenn Sie jedoch private IP-Adressen für den Aufbau Ihres Netzes verwenden und nun das Netz mit dem Internet verbinden, haben Sie wieder ein Problem. Es kann zwar nun nicht mehr zum Adresskonflikt kommen, jedoch werden Sie auch keine Antwort erhalten. Betrachten Sie dazu Abbildung 5.14.

Der Client sendet das Paket mit seiner privaten Absenderadresse über seinen Router in das Internet, wo es den Webserver erreicht. Der Webserver beantwortet das Paket und schickt sein Antwortpaket an die private Adresse des Clients. Da diese Adressen jedoch im Internet verboten sind, wissen die Internet-Router nicht, wohin sie dieses Antwortpaket weiterleiten sollen. Das Paket wird mit der Fehlermeldung „Ziel nicht erreichbar" (Destination unreachable) verworfen.

Für diesen Zweck wurde nun das Source-NAT entwickelt. Dabei kann der Router des Clients dessen private IP-Adresse gegen eine offizielle IP-Adresse austauschen, deren Ort im Internet bekannt ist (siehe Abbildung 5.15). Damit die Antworten auch bei dem Router wieder ankommen, verwendet man hier die offizielle Internet-Adresse des Routers selbst. Grundsätzlich

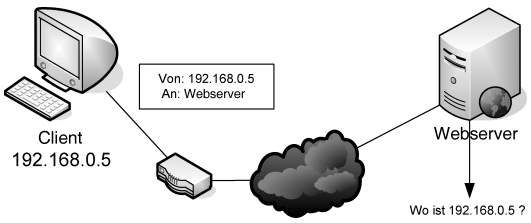


Abbildung 5.14: **Ein Rechner im Internet kann auf einen Verbindungsaufbau mit privater IP-Adresse nicht reagieren.**

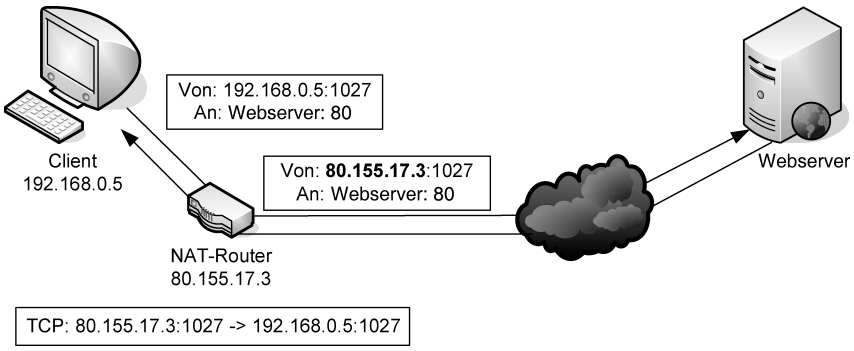


Abbildung 5.15: **Bei dem Source-NAT tauscht der Router die private IP-Adresse des Clients gegen seine eigene offizielle IP-Adresse aus und merkt sich diese Tatsache in einer Tabelle.**

wäre es auch möglich, hier irgendeine andere Adresse einzutragen, jedoch würden die Antwortpakete des Webservers nicht bei dem Router landen und damit nie den Client erreichen.

Der Router merkt sich die Verbindungen, die von ihm genattet wurden, damit er alle weiteren Pakete der Verbindungen identisch natten kann und auch die Antworten aus dem Internet wieder an die richtigen Clients nach innen weitersenden kann. Iptables benötigt hierfür das `nf_conntrack`-Modul, das auch für die Zustandsüberwachung verantwortlich ist.

Sobald Sie nur eine offizielle IP-Adresse für das Source-NAT von mehreren Clients verwenden, handelt es sich eigentlich nicht mehr um ein NAT (Network Address Translation), sondern um ein NAPT (Network Address and Port Translation). Dies wollen wir hier aber nicht genauer besprechen. Sie finden hierfür eine ausführliche Erklärung in Abschnitt 20.2.

Um nun ein Source-NAT zu realisieren, gibt es bei dem Befehl `Iptables` das Ziel `SNAT`. Hiermit können Sie ein Source-NAT durchführen:

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 3.0.0.1
```

Diese NAT-Regel sorgt nun dafür, dass sämtliche Verbindungen auf die Absenderadresse 3.0.0.1 genattet werden. Das bedeutet, dass in allen Paketen die Absenderadresse entsprechend ausgetauscht wird. Die Antwortpakete werden übrigens dann auch automatisch wieder an die Clients passend zurückgeschickt. Das heißt, das NAT wird für die Antwortpakete automatisch rückgängig gemacht. Sie benötigen hier keine zusätzliche Regel!

Sobald Sie Änderungen an einer der Ketten der NAT-Tabelle vornehmen wollen oder diese nur anzeigen wollen, ist es wichtig, dass Sie die NAT-Tabelle spezifisch definieren:

```
iptables -t nat -L # Zeigt den Inhalt der NAT-Tabelle
iptables -L # Zeigt nur den Inhalt der Filter-Tabelle (default)
```

Mit der Option `--to-source` können Sie auch einen IP-Adressbereich oder sogar mehrere Adressen durch Wiederholung der Option angeben:

```
--to-source 192.168.0.1-192.168.0.5
--to-source 192.168.0.1 --to-source 192.168.0.3
```

Diese IP-Adressen werden dann immer abwechselnd verwendet. In vielen Fällen ist das Ziel SNAT jedoch nicht einsetzbar. Vielleicht haben Sie zu Hause einen eigenen Internetzugang. Wahrscheinlich handelt es sich hierbei um einen ISDN- oder ADSL-Zugang. Bei den meisten derartigen Zugängen erhalten Sie für die Dauer Ihrer Onlinezeit eine zufällige IP-Adresse zugewiesen, die Sie für diese Zeit nutzen dürfen. Betreiben Sie nun einen Router und dahinter ein Netz mit privaten IP-Adressen, ist der Aufwand groß. Sie müssten immer Ihr Skript ändern und die IP-Adresse, auf die Ihre Pakete genattet werden sollen, an die aktuell zugewiesene Adresse anpassen.

Hierfür wurde ein zusätzliches Ziel geschaffen: MASQUERADE. Wenn Sie dieses Ziel anstelle von SNAT einsetzen, müssen Sie nicht mehr die IP-Adresse angeben. Iptables wird nun selbst die aktuelle externe IP-Adresse ermitteln und verwenden. Wenn sich die IP-Adresse ändert, so verwendet Iptables automatisch anschließend die richtige IP-Adresse!

Damit wird der Aufbau einer typischen NAT-Regel viel einfacher:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Diese Regel kennen Sie schon aus Ihrem ersten Skript aus dem letzten Abschnitt.

5.7.2 Destination-NAT

Das Destination-NAT ist seltener anzutreffen, aber mindestens genauso interessant wie das Source-NAT. Während ein Source-NAT heute allgemein üblich und allen Administratoren bekannt ist, sind das Destination-NAT und die damit verbundenen Möglichkeiten häufig unbekannt. Das Destination-NAT wird häufig auch als Port-Forwarding bezeichnet.

Bei einem Destination-NAT (DNAT) wird die Zieladresse eines Pakets modifiziert. Damit können Sie ein Paket, das eigentlich an die Firewall gerichtet war, auf einen anderen Rechner weiterleiten. Abbildung 5.16 verdeutlicht die Möglichkeiten.

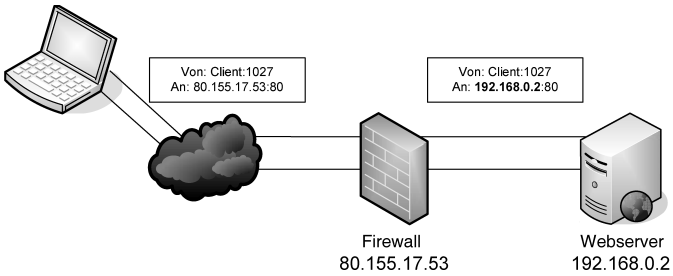


Abbildung 5.16: **Bei dem DNAT können Anfragen von außen auf interne Systeme mit privaten Adressen weitergeleitet werden.**

So können Sie zum Beispiel sämtliche Anfragen auf dem TCP-Port 80 Ihrer Firewall nach innen auf einen Webserver weiterleiten. Der Client im Internet merkt nicht, dass er in Wirklichkeit nicht mit der Firewall, sondern mit dem privaten Webserver spricht. So können Sie Dienste, die Sie nicht direkt auf der Firewall anbieten wollen, dennoch von außen erreichbar anbieten!

Die Regel für dieses Beispiel sähe folgendermaßen aus:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to- ↵
        destination 192.168.0.2
```

Natürlich ist es möglich, einen anderen Port auf einen anderen Rechner weiterzuleiten. Möchten Sie zum Beispiel SMTP-Verbindungen auf Ihren internen E-Mailserver weiterleiten, können Sie die folgende Regel hinzufügen:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 25 -j DNAT --to- ↵
        destination 192.168.0.3
```

Sie können ein Destination-NAT in der `PREROUTING`- und in der `OUTPUT`-Kette durchführen. In der PREROUTING-Kette betrifft das DNAT sämtliche Pakete, die den Rechner von außen erreichen und entweder an ihn lokal gerichtet sind oder weitergeleitet werden müssen. In der OUTPUT-Kette betrifft das DNAT nur die lokal erzeugten Pakete.

Zusätzlich zum Ziel `DNAT` gibt es noch ein weiteres Ziel: `REDIRECT`. Dieses Ziel erlaubt es, Pakete, die normalerweise an ein anderes beliebiges Ziel gerichtet sind, auf einen lokalen Port umzulenken (siehe Abbildung 5.17). Hiermit ist es möglich, einen transparenten Proxy aufzubauen. Mehr Informationen über einen transparenten Proxy erhalten Sie in Kapitel 28.

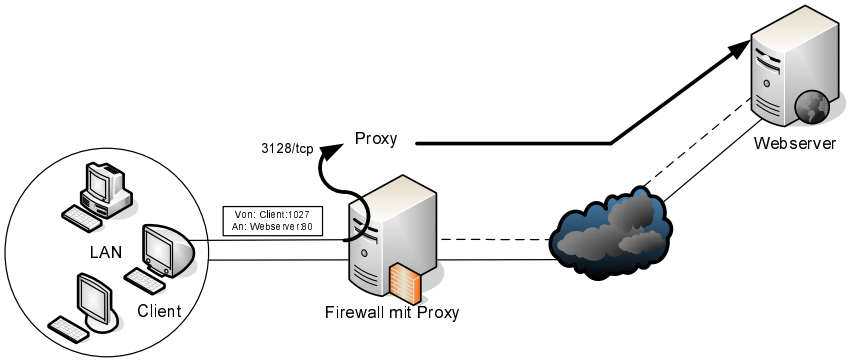


Abbildung 5.17: **Das REDIRECT-Ziel leitet die Pakete auf einen anderen lokalen Port um.**

5.8 NAT-Beispiel

Da sehr viele Erstanwender mit der Implementierung ihrer NAT-Regeln auf Probleme stoßen, folgt nun ein kleines Beispiel. Stellen Sie sich vor, Sie hätten eine Internetanbindung mit einer statischen IP-Adresse, die Ihnen von Ihrem Provider fest zugeteilt worden ist. Nun möchten

Sie sowohl auf das Internet und dessen Dienste zugreifen können als auch selbst Dienste anbieten. Da Sie aber die Dienste nicht auf der Firewall installieren wollen, haben Sie sich folgendes Setup (siehe Abbildung 5.18) ausgedacht: Ihre Firewall ist mit dem Internet verbunden. Die Dienste installieren Sie auf dedizierten Rechnern in Ihrem internen Netz. In dem DNS für Ihre Domäne tragen Sie für `www.meine-domaene.de` die offizielle IP-Adresse Ihrer Firewall ein. Das Gleiche machen Sie für den Mail-Exchanger (MX)[8] Ihrer Domäne.

Im Folgenden sollen nun die NAT-Regeln definiert werden. Natürlich benötigen Sie zusätzlich auch noch Firewall-Regeln. Obwohl Sie diese nun selbst erzeugen können sollten, werden wir sie auch noch betrachten.

Zunächst soll sichergestellt werden, dass die Clients im internen Netz das Internet erreichen können. Hierzu definieren Sie eine Regel, die sämtliche Pakete nattet:

```
EXTDEV=eth0 INTDEV=eth1

# Offizielle IP-Adresse der Firewall: EXTIP
EXTIP=3.0.0.1

$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -j SNAT --to-source $EXTIP
```

Nun möchten Sie noch erreichen, dass Verbindungen, die von außen Ihre Firewall auf dem TCP-Port 80 und 25 erreichen, an Ihren Webserver bzw. Ihren E-Mailserver weitergeleitet werden.

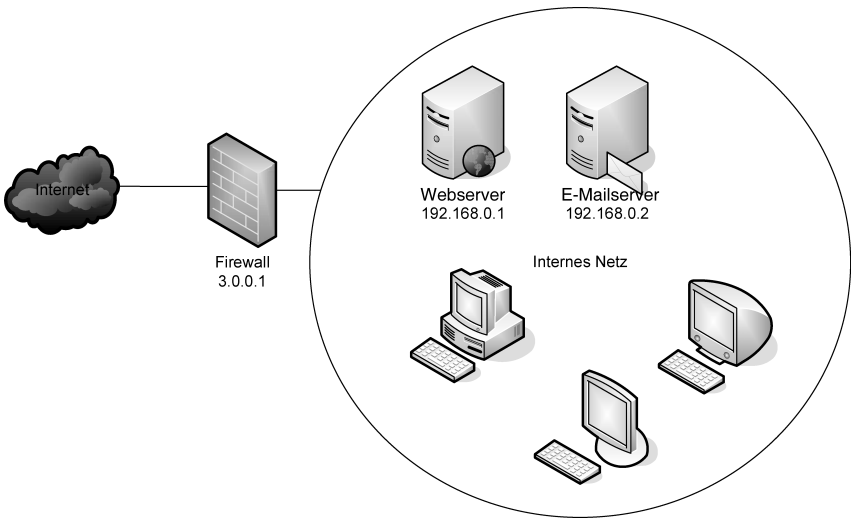


Abbildung 5.18: **In dem internen Netz befinden sich ein Webserver und ein E-Mailserver, die aus dem Internet erreicht werden sollen.**

8 Der Mail-Exchanger definiert, welcher Rechner für eine Domäne die E-Mails annimmt. Sie können im DNS mehrere Mail-Exchanger (MX) mit unterschiedlicher Priorität zur Lastverteilung und Ausfallsicherheit angeben.

```
# Interne IP-Adressen des Webservers und Mailservers
WEBSERVER=192.168.0.1
EMAILSERVER=192.168.0.2

$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp --dport 80 -j DNAT --to- ↵
        destination $WEBSERVER
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp --dport 25 -j DNAT --to- ↵
        destination $EMAILSERVER
```

Diese Regeln reichen für das NAT bereits vollkommen aus. Nun benötigen Sie noch einige Firewall-Regeln, die zunächst die Verbindungen von innen nach außen für Ihre Clients bei dem Zugriff auf das Internet zulassen:

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Zusätzlich benötigen Sie noch Regeln, die die Web- und die E-Mail-Verbindungen nach innen auf Ihren Web- und Ihren E-Mailserver erlauben:

```
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -p tcp --dport 80 -d  ↵
        $WEBSERVER -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -p tcp --dport 25 -d  ↵
        $EMAILSERVER -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Die erste Regel erlaubt neue Pakete (`-m state --state NEW`) von außen (`-i $EXTDEV`) nach innen (`-o $INTDEV`), wenn sie an den Webserver (`-d $WEBSERVER`) auf dem TCP-Port 80 gerichtet sind (`-p tcp --dport 80`). Die zweite Regel wiederholt dies für den E-Mailserver. Die dritte Regel erlaubt sämtliche weiteren Pakete dieser Verbindungen. Diese dritte Regel hatten wir aber bereits weiter oben definiert. Daher brauchen wir diese hier nicht erneut anzugeben!

Ein komplettes Skript würde also folgendermaßen aussehen:

```
#!/bin/bash
# Autor  : Ralf Spenneberg
# Version: 0.1
# Datum  : 17. Dez 2004
# Dieses Skript lädt die Regeln für die Firewall.
# Außerdem erlaubt es den Zugriff von außen auf einen internen Web- und E ↵
        -Mailserver

INTDEV="eth1"
# Achtung: Beim Einsatz des virtuellen Clients lautet diese Variable  ↵
        INTDEV="tap0"
EXTDEV="eth0"
```

```
# Offizielle IP-Adresse der Firewall: EXTIP
EXTIP=3.0.0.1
# Interne IP-Adressen des Webservers und Mailservers
WEBSERVER=192.168.0.1
EMAILSERVER=192.168.0.2

# Definiere einige Befehle
ECHO="/bin/echo"
SYSCTL="/bin/sysctl"
IPTABLES="/sbin/iptables"

# Leere die Ketten
$IPTABLES -F

# Die NAT-Tabelle muss extra geleert werden
$IPTABLES -t nat -F

# Verwerfe erst mal alles
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Akzeptiere Verbindungsaufbauten von innen
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -m state --state NEW -j ACCEPT

# Akzeptiere alle Pakete, die Teil einer aufgebauten Verbindung sind
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# Akzeptiere Verbindungen für den Web- und E-Mailserver
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -p tcp --dport 80 -d ↵
        $WEBSERVER -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -p tcp --dport 25 -d ↵
        $EMAILSERVER -m state --state NEW -j ACCEPT

# Setze NAT-Regeln

# Natte den Zugriff nach außen
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -j SNAT --to-source $EXTIP

# Wenn Sie die IP-Adressen nicht kennen, können Sie auch diese Zeile ↵
        nutzen:
# $IPTABLES -t nat -A POSTROUTING -o $EXTDEV -j MASQUERADE
```

```
# Natte den Zugriff von außen auf den Web- und E-Mailserver
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp --dport 80 -j DNAT --to- ↵
        destination $WEBSERVER
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp --dport 25 -j DNAT --to- ↵
        destination $EMAILSERVER

# Aktiviere das Forwarding
$ECHO "1" > /proc/sys/net/ipv4/ip_forward

# Alternativ:
#$SYSCTL -w net.ipv4.ip_forward=1
```

5.9 Die Mangle-Tabelle

Die Mangle-Tabelle führt ein Schattendasein bei Iptables. Sie bietet nicht besonders viele Funktionen. Dabei verfügt sie über die meisten Ketten: `INPUT`, `OUTPUT`, `FORWARD`, `PREROUTING` und `POSTROUTING` (siehe Abbildung 5.19)[9]. In der Mangle-Tabelle können Sie Pakete modifizieren. Leider bietet Iptables nur einige wenige Möglichkeiten für diese Modifikationen.

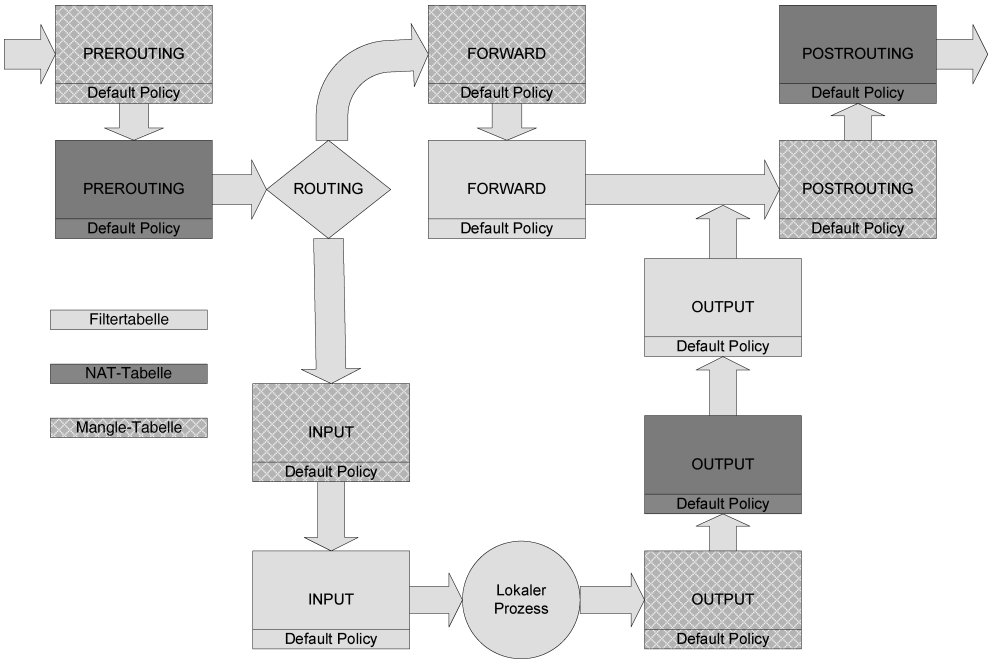


Abbildung 5.19: **In der Mangle-Tabelle können Sie ein Paket modifizieren.**

9 Wenn Sie noch einen alten Kernel <2.4.18 einsetzen sollten, verfügt Ihr Kernel in der Mangle-Tabelle nur über eine PREROUTING- und eine OUTPUT-Kette.

Bevor wir uns über die verschiedenen Möglichkeiten einen Überblick verschaffen, betrachten Sie bitte Abbildung 5.19. Ihnen sollte auffallen, dass jedes Paket zuerst die entsprechende Mangle-Kette durchläuft, bevor die NAT- und Filter-Ketten durchlaufen werden. Dadurch können Sie in der Mangle-Tabelle ein Paket modifizieren und diese Modifikation in einer der Filter-Ketten zum Beispiel prüfen.

Welche Möglichkeiten gibt es nun in der Mangle-Tabelle? Die meisten Distributionen bieten folgende Ziele für die Mangle-Tabelle an:

» `DSCP`: Dieses Ziel erlaubt es, die Diffserv-Codepoints (DSCP) in dem TOS-Feld des IP-Pakets zu ändern. Diffserv-Codepoints werden für die Realisierung von Quality-of-Service verwendet. Weitere Informationen zu DSCP finden Sie in Abschnitt 21.5.

» `ECN`: Hiermit können Sie die ECN-Bits aus dem TCP-Header entfernen. ECN ist eine Art Stauvermeidung für das Internet. Leider nutzen nur sehr wenige Betriebssysteme bisher diese Funktion, und viele Firewalls verwerfen derartig markierte Pakete. Weitere Informationen finden Sie in Abschnitt 21.6.

» `MARK`: Hiermit können Sie Pakete mit einer Zahl markieren. Diese Markierung verändert das Paket nicht, sondern klebt wie ein Aufkleber an dem Paket, und zwar so lange, wie es sich auf der Firewall befindet. Sobald das Paket die Firewall verlässt, wird die Markierung entfernt. Weitere Informationen über dieses Ziel finden Sie in Abschnitt 21.7.

» `TOS`: Dieses Ziel kann das Type-of-Service-Feld im IP-Header setzen. Weitere Informationen über dieses Ziel finden Sie in Abschnitt 21.8.

» `TTL`: Hiermit können Sie den TTL-Wert eines Pakets verändern. So können Sie den TTL-Wert sowohl herauf- als auch herabsetzen. Es existieren kaum sinnvolle Anwendungen hierfür. Weitere Informationen finden Sie in Abschnitt 21.10.

» `CLASSIFY`: Der Linux-Kernel erlaubt die Einordnung von Paketen in unterschiedliche Quality-of-Service-Klassen. Dieses Ziel erlaubt die Einsortierung der Pakete mit Firewall-Regeln. Weitere Informationen finden Sie in Abschnitt 21.3.

Ob Ihr Kernel alle diese Ziele unterstützt, können Sie leicht selbst herausbekommen. Schauen Sie einfach in das Verzeichnis `/lib/modules/<Kernel-Version>/kernel/net/ipv4/netfilter`. Befindet sich dort die Datei `ipt_TTL.ko`, so unterstützt der Kernel das Ziel. Achten Sie auf die Groß- und Kleinschreibung.

5.10 Die Raw-Tabelle

Die `Raw`-Tabelle besitzt zwei Ketten: `PREROUTING` und `OUTPUT`. Sie steht auf alten Kerneln nicht zur Verfügung. Jedes Paket durchläuft zuerst diese Ketten. Wenn das Paket von außen kommt, durchläuft es die Raw-PREROUTING-Kette. Wenn das Paket lokal erzeugt wurde, durchläuft es die Raw-OUTPUT-Kette (siehe Abbildung 5.20). Da zu diesem Zeitpunkt `nf_conntrack` noch nicht die Pakete analysiert hat[10], können Sie hier mit dem Ziel `NOTRACK` Pakete von

10 Das passiert in der NAT-Tabelle.

Bitte bestätigen Sie die Änderung.

Editor's or typesetter's annotations (will be removed before the final TEX run)

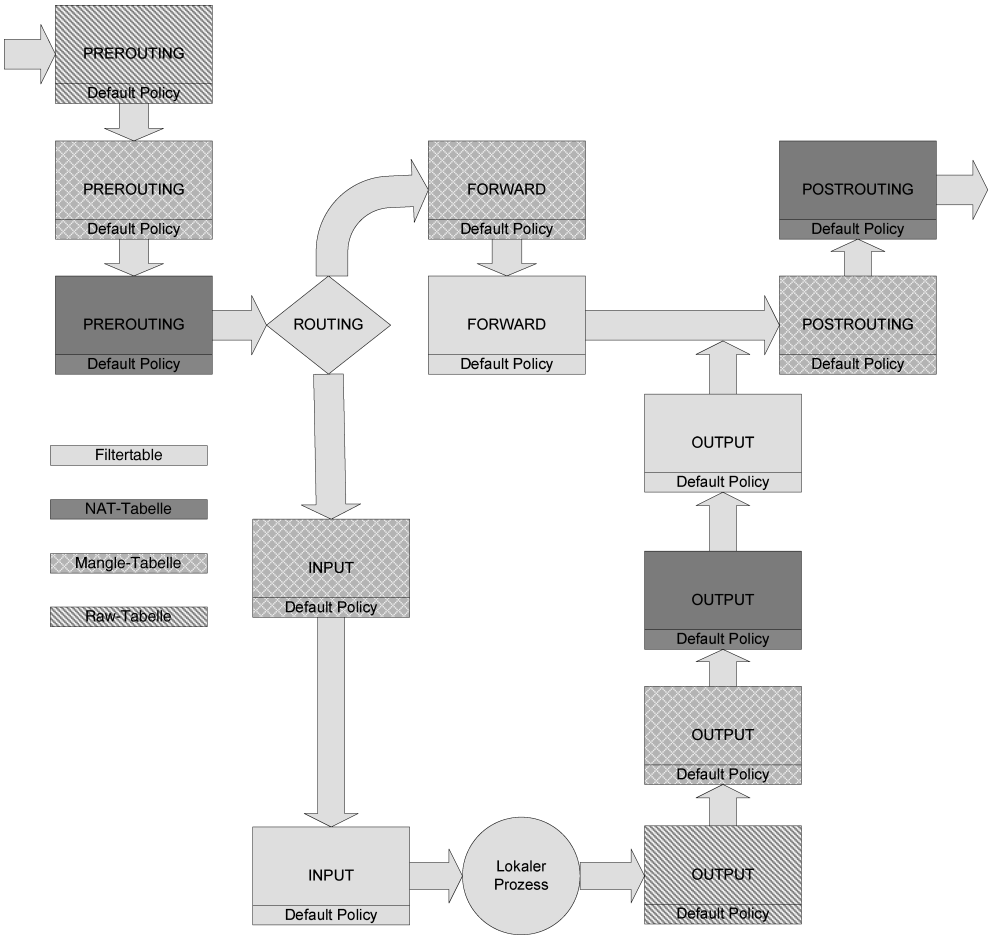


Abbildung 5.20: **Die Raw-Tabelle wird vor jeder anderen Tabelle durchlaufen. Sie steht nicht auf allen Kerneln zur Verfügung.**

der Überwachung ausschließen. Das bedeutet dann aber auch gleichzeitig, dass diese Pakete nicht mehr genattet werden können.

Als zweites Ziel dieser Tabelle existiert `TRACE`. Dieses Ziel erlaubt es, bestimmte Pakete zu verfolgen, während sie Ihr Regelwerk durchlaufen. Ein Paket, das mit dem TRACE-Ziel markiert wurde, wird anschließend von jeder Regel protokolliert, die auf das Paket zutrifft:

```
TRACE: tablename/chainname/rulenum packet
```

Weitere Informationen über die Raw-Tabelle finden Sie in Kapitel 22.

5.11 Einstellungen des Kernels

Zusätzlich zu den Firewall-Regeln gibt es auch noch eine ganze Reihe von Kerneleinstellungen, die eine zusätzliche Sicherheit für Ihr System schaffen können. Hierbei handelt es sich um Einstellungen, die Sie in dem Verzeichnis `/proc/sys/net/ipv4` verändern können, wodurch Sie so zur Laufzeit das Verhalten des Kernels modifizieren.

Eine erste Einstellung haben Sie bereits kennengelernt: `ip_forward`. Hiermit können Sie entscheiden, ob Ihr System Pakete zwischen den Netzwerkkarten weiterleiten soll oder nicht. Ist der Wert in dieser Datei eine 0, so werden die Pakete nicht weitergeleitet. Ist der Wert 1, so findet eine Weiterleitung statt.

Sie haben bereits zwei Varianten kennengelernt, wie Sie diesen Wert einstellen können:

1. `echo 1 > /proc/sys/net/ipv4/ip_forward`

2. `sysctl -w net.ipv4.ip_forward=1`

Wenn Sie bereits die erste Syntax kennen, können Sie sehr leicht die Syntax des `sysctl`-Kommandos ableiten. Entfernen Sie einfach den Pfad `/proc/sys/`, und ersetzen Sie anschließend alle Schrägstriche durch einen Punkt.[11] Mit dieser Syntax können Sie die zu setzenden Variablen auch in der Datei `/etc/sysctl.conf` eintragen. Diese Datei steht auf fast allen aktuellen Distributionen zur Verfügung und wird üblicherweise nach dem Booten und nach jedem Neustart des Netzwerks gelesen.

Weitere wichtige Kernelparameter sind:

» `ip_echo_ignore_broadcasts`: Klassischerweise reagiert jeder UNIX/Linux-Rechner auf einen Broadcast-Ping. Hierbei handelt es sich um eine Ping-Anfrage, die an die Broadcast-Adresse eines Netzes gesendet wurde (z.B. mit `ping -b 192.168.0.255`). Dies kann jedoch für eine Denial-of-Service-Attacke genutzt werden. Der Angreifer braucht nur ein Ping-Paket mit gefälschter Absenderadresse loszuschicken, und sämtliche UNIX/Linux-Systeme in dem Zielnetz antworten. So kommt es zu einer Vervielfältigung des Pakets, die zum DoS auf die Netzwerkleitung führen kann. Dieser Angriff ist als SMURF-Angriff bekannt. Die Netze, die eine Vervielfältigung erlauben, sind sogenannte Amplifier-(Verstärker-)Netze. Eine Übersicht über derartige Netze gibt die Smurf-Amplifier-Registry (`http://www.powertech.no/smurf/`). Damit die Gefahr durch eine fehlerhafte Konfiguration des Netzes gar nicht erst entsteht, wird allgemein empfohlen, diese Funktion des Linux-Systems abzuschalten:

  ```
  sysctl -w net.ipv4.ip_echo_ignore_broadcasts=1
  ```

  Viele Distributionen haben diese Einstellung bereits zum Default gemacht.

11 Der letzte Schritt ist nicht zwingend erforderlich. Es funktioniert auch: `sysctl -w net/ipv4/ip_forward=1`

» `ip_default_ttl`: Mit diesem Wert können Sie den TTL-Wert einstellen, den sämtliche Pakete erhalten sollen, die von Ihrem Rechner erzeugt werden. Typisch für Linux ist 64. Andere Betriebssysteme verwenden andere (z.B. 255) Werte. Daher kann eine grobe Erkennung des Betriebssystems über diesen Wert erfolgen. Eine Änderung ist nicht sinnvoll.

» `tcp_ecn`: Hiermit stellen Sie ein, ob Sie die Explicit Congestion Notification (ECN)[12] wünschen. Dabei handelt es sich um eine neue Methode, mit der zwei kommunizierende Systeme im Internet eine „Verstopfung" frühzeitig erkennen können und durch präventive Verlangsamung der Pakete diese zu umgehen versuchen. Leider gibt es viele Firewalls, die Pakete verwerfen, die die ECN-Bits tragen. Daher ist es im Moment nicht sinnvoll, ECN CE U grundsätzlich einzuschalten. Wenn Sie ECN einschalten, können Sie es mithilfe von Firewall-Regeln in der Mangle-Tabelle für bestimmte Ziele auch wieder abschalten (siehe Abschnitt 21.6).

» `tcp_syncookies`: Der TCP/IP-Stack der meisten Betriebssysteme (auch von Linux) ist durch einen SYN-Flood-DoS verwundbar. Hierbei handelt es sich um sehr viele TCP-SYN-Pakete, die der Angreifer mit gefälschter Adresse an den Rechner schickt. Der angegriffene Rechner kann zwischen den gefälschten TCP-SYN-Paketen und regulären Paketen nicht unterscheiden und muss auf alle diese Pakete reagieren und sich die Informationen des Verbindungsaufbaus merken. Dies kann bei genügend gefälschten Paketen zu einem Überlauf der hierfür verwendeten Tabelle führen. Die TCP-Syncookies schaffen diese Tabelle ab und speichern die notwendigen Daten in der Verbindung selbst.[13] Ein Einschalten dieser Funktion hilft natürlich nur auf dem angegriffenen Server selbst. Da ein Paketfilter die Pakete jedoch nur durchleitet und nicht beantwortet, hat diese Funktion auf einem Paketfilter keine Auswirkung auf die gerouteten Pakete.[14]

» `conf/<interface>/accept_source_route`: Diese Funktion sollte auf jedem System abgeschaltet sein. Bei eingeschaltetem Zustand besteht die Gefahr, dass ein Angreifer Pakete nach seinem Willen routen kann. Als Source-Routing bezeichnet man nämlich die Funktion, dass der Absender den Weg des Pakets über IP-Optionen beeinflussen kann.

» `conf/<interface>/rp_filter`: Dies ist ein in den Kernel eingebauter Anti-Spoofing-Schutz. Sie können diesen Schutz anschalten (1) und abschalten (0). Der Linux-Kernel betrachtet bei eingeschaltetem Anti-Spoofing-Schutz jedes Paket und prüft, ob ein potenzielles Antwortpaket den umgekehrten Weg nehmen würde. Diese Reverse-Path-Filterung ist im RFC 1812 beschrieben.

» `conf/<interface>/log_martians`: Diese Protokollfunktion protokolliert alle unmöglichen Pakete über den zentralen Syslog-Daemon. Dazu gehören zum Beispiel die Pakete, die von `rp_filter` und `accept_source_route` verworfen wurden.

12 Weitere Informationen erhalten Sie hier: `http://www.icir.org/floyd/ecn.html`

13 Für die Interessierten: Die Daten werden in der Sequenznummer verschlüsselt hinterlegt. Weitere Informationen über Syncookies finden Sie unter: `http://www.heise.de/security/artikel/43066/2` und `http://cr.yp.to/syncookies.html`.

14 OpenBSD kann als SYN-Proxy vor einem SYN-Flood schützen. Allerdings werden hier keine Syncookies verwendet. OpenBSD verwirft zufällig Verbindungen, sobald die Tabelle sich füllt.

CE U Bitte diese Einfügung bestätigen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

» `conf/<interface>/forwarding`: Hiermit können Sie für jedes Interface einzeln die Forwarding-Funktion konfigurieren. Wenn Sie das Forwarding für den gesamten Rechner einschalten (`ip_forward`), wird dieser Parameter bei jeder Netzwerkkarte ebenfalls eingeschaltet. Schalten Sie ihn für eine bestimmte Netzwerkkarte wieder ab, so leitet die entsprechende Netzwerkkarte Pakete nicht mehr weiter.

Alle Parameter hier aufzuzählen, würde den Rahmen dieses Abschnitts sprengen. Sie finden weitere wichtige Parameter und ihre Erläuterung in Kapitel 23.

Für Sie zunächst sinnvolle zusätzliche Parameter sind:

```
sysctl -w net.ipv4.conf.all.rp_filter=1
sysctl -w net.ipv4.conf.all.accept_source_route=0
```

5.12 Protokollierung

Die Protokollierung ist ein sehr wichtiger Vorgang bei einer Firewall. Jeder Firewall-Administrator sollte wissen, was auf seiner Firewall vorgeht. Dafür ist eine Protokollierung der abgelehnten Verbindungen zwingend erforderlich. Vielleicht ist auch eine Protokollierung der erlaubten Verbindungen in bestimmten Umgebungen wünschenswert.

In diesem Abschnitt werden Sie die einfache Protokollierung mit Iptables kennenlernen. Werkzeuge zur Auswertung werden in einem späteren Kapitel (Kapitel 24) vorgestellt. Auch eine weitere fortgeschrittene Protokollalternative (`ULOG`) wird in einem späteren Abschnitt besprochen (siehe Abschnitt 24.1).

Für die Protokollierung mit Iptables gibt es ein einfaches Ziel, das in der Firewall-Regel angegeben wird: `LOG`. Eine Protokollregel sieht dann zum Beispiel so aus:

```
# Die nächste Regel protokolliert alle Verbindungsaufnahmen von außen
iptables -A FORWARD -i $EXTDEV -m state --state NEW -j LOG
```

Während alle anderen Ziele, die Sie bisher kennengelernt haben (DROP, REJECT, ACCEPT), direkt über das Schicksal des Pakets entschieden haben und eine weitere Abarbeitung der Regeln in der Kette nicht erfolgte, ist das bei LOG nicht so. Wenn ein Paket protokolliert wurde, wird die weitere Abarbeitung der Regeln in der Kette nicht abgebrochen, da über sein Schicksal noch nicht entschieden worden ist.

Möchten Sie also ein Telnet-Paket protokollieren und anschließend verwerfen, so benötigen Sie zwei Regeln:

```
$IPTABLES -A FORWARD -p tcp --dport 23 -j LOG
$IPTABLES -A FORWARD -p tcp --dport 23 -j DROP
```

Achten Sie darauf, dass Sie nicht die Reihenfolge durcheinanderbringen. Dann würde nämlich das Paket zuerst verworfen werden, die Abarbeitung der weiteren Regeln würde nicht erfolgen, und die LOG-Regel würde nie ausgeführt werden!

Genauso können Sie ein Paket protokollieren und akzeptieren. Möchten Sie alle neuen Telnet-Verbindungen erlauben, aber auch protokollieren, können Sie die folgenden Regeln verwenden:

```
$IPTABLES -A FORWARD -p tcp --dport 23 -m state --state NEW -j LOG
$IPTABLES -A FORWARD -p tcp --dport 23 -m state --state NEW -j ACCEPT
```

Wie sehen nun die protokollierten Pakete aus? Das Protokollformat ist zunächst ein wenig gewöhnungsbedürftig, enthält aber alle wichtigen Informationen:

```
Jul 14 18:41:45 bibo kernel: IN= OUT=eth0 SRC=192.168.255.100 DST ↵
        =209.132.177.100 LEN=453 TOS=0x00 PREC=0x00 TTL=64 ID=33418 DF ↵
        PROTO=TCP SPT=35379 DPT=443 WINDOW=2068 RES=0x00 ACK PSH URGP=0
```

Dies ist ein typisches TCP-Paket, das so vom Syslog-Daemon protokolliert wurde. In welcher Datei die Protokollmeldungen auftauchen, hängt von Ihrer Syslog-Konfiguration ab.

INFO

Syslog-Daemon-Konfiguration.
Bei der Konfiguration des Syslog-Daemons sollten Sie zunächst prüfen, welche Variante Ihr aktuelles Linux-System nutzt. Allgemein werden im Moment vor allem drei alternative Syslog-Daemons eingesetzt:

- *Der von dem BSD-Syslogd abgeleitete Syslog-Daemon wird in der Datei `/etc/syslog.conf` konfiguriert, und der laufende Prozess heißt `syslogd`.*
- *Der Syslog-ng ist ein moderner Syslog-Daemon, der viele Erweiterungen gegenüber dem BSD-Syslog erfahren hat. Seine Konfigurationsdatei ist die `/etc/syslog-ng.conf`, und der Prozess heißt auch entsprechend. Während im Folgenden kurz der BSD-Syslogd angesprochen wird, wird dieser Daemon später besprochen (siehe Abschnitt 9.3).*
- *Der Rsyslogd ist eine Weiterentwicklung des Syslogd. Er ist abwärtskompatibel zum originalen Syslogd und wird ebenfalls in einem eigenen Abschnitt besprochen (siehe Abschnitt 9.2).*

Hier wollen wir uns kurz auf den BSD-Syslog beschränken. Schauen Sie in die Datei `/etc/syslog.conf`, und suchen Sie nach einem nicht auskommentierten Eintrag `kernel.<irgendwas>` oder `*.<irgendwas>`. In der rechten Spalte finden Sie dann den Namen der Datei, in der alle Meldungen des Kernels bzw. aller Protokollquellen mit mindestens der Priorität <irgendwas> protokolliert werden. In dieser Datei finden Sie dann auch die Firewall-Protokolle.

Betrachten wir den Protokolleintrag der Reihe nach. Zunächst schreibt der Protokolldienst das aktuelle Datum und die Uhrzeit: `Jul 14 18:41:45`.

Der UNIX-Protokolldienst schreibt keine Jahreszahlen! Wenn Sie die Protokolle länger aufbewahren wollen, müssen Sie selbst sich das Jahr merken (z.B. in dem Namen der Protokolldatei).

Anschließend schreibt der Protokolldienst den Namen des Rechners, der den Eintrag erzeugt hat: `bibo`. Dies ist wichtig, da der Protokolldienst grundsätzlich netzwerkfähig ist und Sie die Protokolle auf einem Protokollserver zusammenfassen können. Damit Sie später wissen, welche Einträge von welchem Server kommen, wird auch der Name protokolliert.

Es folgt die Quelle der Meldung: `kernel:`. An dieser Stelle finden Sie auch `mail`, `cron`, `news` etc.

Nun folgt die eigentliche Meldung. Sie beginnt mit der Information, über welche Netzwerkkarte das protokollierte Paket den Rechner erreicht hat und verlassen wird: `IN= OUT=eth0`. Hier handelt es sich um ein lokal erzeugtes Paket, das den Rechner über eth0 verlässt.

Anschließend werden die Absender- und Ziel-IP-Adresse protokolliert:
`SRC=192.168.255.100 DST=209.132.177.100`.
Handelt es sich um ein ankommendes Paket, sehen Sie zuvor auch noch den Ethernet-Header des Pakets: `MAC=00:0f:1f:1c:bd:15:00:12:17:dd:dd:07:08:00`. Dieser Ethernet-Header enthält in den ersten sechs Bytes die Quell-MAC-Adresse (00:0f:1f:1c:bd:15), dann die Ziel-MAC-Adresse (00:12:17:dd:dd:07) und schließlich noch die Information, dass sich in dem Ethernet-Paket ein IP-Paket befindet (08:00).

Auf die Adressen folgen die weiteren Informationen aus dem IP-Header des Pakets: die Länge in Bytes (`LEN=40`), der Inhalt des TOS- und PREC-Feldes (`TOS=0x00 PREC=0x00`), der TTL-Wert (`TTL=64`) und die Fragmentierungsidentifikationsnummer (`ID=33418`). Wenn das DF-Bit gesetzt ist, erscheint im Protokoll ein `DF`. Als Letztes folgt das Transportprotokoll (`PROTO=TCP`).

Bei einem TCP-Paket folgen nun der Quell- und Zielport (`SPT=35379 DPT=443`), die Größe des TCP-Windows (`WINDOW=2068`), der Zustand der reservierten Bits im TCP-Header (`RES=0x00`) und die weiteren Flags des TCP-Headers (`ACK PSH`). Falls ein URG-Flag gesetzt ist, muss auch der Urgent-Pointer gesetzt sein: `URGP=0`.

Anhand des Quell- und Zielports können Sie ableiten, um welche Art von Verbindung es sich wahrscheinlich handelt. Hier finden Sie einen Zielport 443. Es wird sich daher um eine HTTPS-Verbindung (eine SSL-verschlüsselte HTTP-Verbindung) handeln.

Das LOG-Ziel verfügt über einige Optionen, die sein Verhalten noch verändern können:

» `--log-level <level>`: Hiermit können Sie die Priorität der Meldungen anpassen und damit diese Meldungen durch eine Anpassung der `/etc/syslog.conf` in unterschiedlichen Dateien protokollieren.

» `--log-prefix <prefix>`: Sie können eine Zeichenkette von bis zu 29 Buchstaben angeben, die jeder Protokollmeldung vorangestellt wird. Damit können Sie anschließend sehr leicht (zum Beispiel mit `grep`) in Ihren Protokollen nach bestimmten Meldungen suchen.

```
$IPTABLES -A FORWARD -p tcp --dport 23 -m state --state NEW -j LOG -- ↵
        log-prefix "Neue-Telnet-Verbindung"
```

» `--log-tcp-sequence`: Dies protokolliert bei jedem Paket auch die verwendeten Sequenznummern. Sind die Sequenznummern und der Mechanismus ihrer Erzeugung bekannt, ist es möglich, einen Man-in-the-Middle-Angriff auf die Verbindung durchzuführen. Bei einer Protokollierung sollten Sie daher sicherstellen, dass zumindest keine normalen Benutzer die Protokolle lesen dürfen.

» `--log-tcp-options`, `--log-ip-options`: Hiermit werden mögliche zusätzliche Optionen des TCP-Headers (z.B. MSS) und des IP-Headers (z.B. Source-Routing) protokolliert.

» `--log-uid`: Diese Option ist nur in der OUTPUT-Kette sinnvoll. Sie protokolliert bei lokal erzeugten Paketen zusätzlich die UID des Users, dessen Prozess das Paket erzeugt hat.

STOP

Achten Sie darauf, dass Sie vorsichtig mit der Protokollierung umgehen. Eine unachtsam gesetzte Protokollregel kann schnell mehrere 100 MByte an Protokollen pro Tag erzeugen! Der Analyse der Protokolle ist das ganze Kapitel 24 gewidmet.

5.13 Test der Firewall

Dies ist häufig eines der schwierigsten Kapitel für den Einsteiger. Wie testet man eine Firewall? Wie testet man, ob die Firewall sicher ist, aber dennoch die notwendigen Zugriffe erlaubt?

Zunächst sollten Sie prüfen, ob das von Ihnen geschriebene Skript syntaktisch korrekt ist. Wenn bei dem Aufruf Ihres Skripts bereits Fehlermeldungen erscheinen, sollten Sie diese Fehler suchen und entfernen. Anschließend können Sie die Funktion testen.

Besonders schön ist es natürlich, wenn Sie eine möglichst reale Testumgebung für Ihre Versuche haben. Das kann auch eine virtuelle Testumgebung sein, die auf KVM, XEN oder VMware basiert.

Dann sollten Sie zunächst prüfen, ob alle Dienste so funktionieren, wie Sie es wünschen. Dies können Sie relativ einfach und effektiv mit den realen Programmen durchführen. Wenn diese Ihnen nicht zur Verfügung stehen, können Sie mit dem Werkzeug `nmap` zumindest prüfen, ob die benötigten Verbindungen aufgebaut werden können. Hierzu starten Sie Nmap mit der Option `-sT`. Dies führt einen TCP-Connect-Scan durch. Das bedeutet, Nmap baut tatsächlich die TCP-Verbindung so auf, wie es der Client auch machen würde.

```
# nmap -sT <ziel>
```

Funktionieren alle Dienste entsprechend Ihren Wünschen, können Sie nun prüfen, ob alle unerwünschten Zugriffe korrekt abgelehnt und bei entsprechender Konfiguration auch protokolliert werden.

Hierzu müssen Sie vor allem von außen auch den Zugriff testen. Falls Sie keine Testumgebung zur Verfügung haben und einen Test auf dem System in seiner Live-Umgebung durchführen müssen, müssen Sie den Scan aus dem Internet durchführen. Hierzu können Sie zum Beispiel einen zweiten Rechner nutzen, der sich über ISDN oder Modem eingewählt hat. Vielleicht verfügen Sie auch über einen Bekannten, der einen Internetzugang besitzt und den Scan durchführen kann. Trifft das alles nicht zu, so gibt es im Internet Dienstleister, die Ihnen anbieten, Sie zu scannen.CE[V]

Beachten Sie bitte, dass ein Scan wesentlich länger dauert, wenn Ihre Firewall die unerwünschten Pakete verwirft (`DROP`) und nicht ablehnt (`REJECT`). Am Ende eines Scans erhalten Sie von Nmap eine Ausgabe, die der folgenden ähnelt:

CE[V] Hier URLs ergänzen.

```
[root@bibo ~]# nmap -sT www.nohup.info

Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2005-07-14 20:06 ↵
        CEST
Interesting ports on mail.spenneberg.net (217.160.128.61):
(The 1618 ports scanned but not shown below are in state: closed)
PORT    STATE    SERVICE
22/tcp  open     ssh
25/tcp  open     smtp
53/tcp  open     domain
80/tcp  open     http
140/tcp filtered emfis-data
161/tcp filtered snmp
168/tcp filtered rsvd
225/tcp filtered unknown
237/tcp filtered unknown
258/tcp filtered Fw1-mc-gui
271/tcp filtered unknown
310/tcp filtered bhmds
331/tcp filtered unknown
366/tcp filtered odmr
379/tcp filtered is99c
423/tcp filtered opc-job-start
443/tcp open     https
536/tcp filtered opalis-rdv
... gekürzt ...


Nmap finished: 1 IP address (1 host up) scanned in 24.558 seconds
```

Hier können Sie erkennen, dass sehr viele Ports von einer Firewall gefiltert werden, aber doch einige wenige Ports offen sind: 22, 25, 53, 80, 443 und 993. Auf dem System laufen anscheinend ein SSH-Daemon (22) und ein DNS-Server (53). Außerdem findet sich ein Webserver, der sowohl HTTP (80) als auch HTTPS-Verbindungen (443) entgegennimmt, und ein E-Mailserver, der SMTP- (25) und SSL-verschlüsselte IMAP-Verbindungen (993) entgegennimmt. Alle hier nicht aufgeführten Ports werden nicht gefiltert, sind aber dennoch geschlossen, da auf dem Zielsystem kein Dienst installiert wurde.

Weitere Hinweise zur Anwendung von Nmap und weitere Werkzeuge zum Test Ihrer Firewall finden Sie in Kapitel 15.

6. Härtung eines Linux-Systems

Wenn Sie eine Linux-Distribution als Firewall-System nutzen wollen, sollte diese gehärtet sein. Die üblichen Distributionen (SUSE, OpenSUSE, Fedora und Debian) sind noch nicht an die besonderen Anforderungen einer Firewall angepasst. Wenn Sie eine für diesen speziellen Zweck vorbereitete Distribution nutzen, haben die Hersteller meist bereits gute Vorarbeit geleistet. Dennoch kann es nicht schaden, wenn Sie trotzdem noch einmal einen Blick auf die Härtung werfen.

Was ist Härtung? Eine gute Frage. Es gibt keine allgemeine Antwort. Kein RFC oder Standard definiert die Härtung eines Betriebssystems.[1] Im Folgenden gebe ich Ihnen meine Vorstellung von der Härtung eines Betriebssystems wieder.

Da die Härtung eines Betriebssystems keine einfache Aufgabe ist, schnell wichtige Punkte vergessen werden und häufig ähnliche Systeme anschließend unterschiedlich konfiguriert wurden, stelle ich Ihnen anschließend in Abschnitt 6.9 Bastille-Linux vor. Bastille-Linux ist ein Werkzeug für die wichtigsten Linux-Distributionen (und auch Mac OS X und HP-UX), das Sie bei dieser Arbeit menügeführt unterstützt. Selbst wenn Sie es nicht für diesen Zweck einsetzen möchten, können Sie es sehr gut als Assessment-Werkzeug verwenden, um Ihre manuelle Konfiguration zu prüfen.

STOP

Leider wird Bastille-Linux aktuell nicht gepflegt. Es existiert aber leider kein alternatives Produkt. Daher wird es, in der Hoffnung dass bald die Pflege wieder übernommen wird, hier weiterhin vorgestellt.

6.1 Warum sollte eine Firewall gehärtet werden?

Um zu klären, warum eine Firewall gehärtet werden sollte, sollten wir uns zunächst ansehen, was Härtung bedeutet. Wenn Sie eine aktuelle Linux-Distribution durch bloßes Anwählen der Default-Einstellungen installieren, so installieren Sie zahllose, für eine Firewall überflüssige Softwarepakete. Diese Softwarepakete sind häufig nicht nur überflüssig, sondern können auch Sicherheitslücken enthalten. Wenn die Distribution nach einem Neustart einige dieser überflüssigen Netzwerkdienste aktiviert, sodass sie von außen erreichbar sind, könnten diese Dienste auch über das Netz angreifbar sein.

1 Es gibt mit dem RFC2196 jedoch ein Site-Security-Handbook. Dies betrachtet u.a. auch Firewall-Systeme.

Natürlich installieren Sie Ihre Firewall-Regeln so, dass ein Zugriff auf die Dienste gar nicht erst möglich ist. Aber vielleicht unterläuft Ihnen bei der Konfiguration ein Fehler? Um derartige Probleme von vornherein zu umgehen, sollten Sie diese Dienste deaktivieren und besser gar nicht installieren. Auch sollten alle überflüssigen Benutzerkonten von dem System entfernt werden. Der Bootvorgang sollte auf Möglichkeiten zur Kompromittierung des Systems überprüft werden, und die Rechte der Verzeichnisse[TS a] und der ausführbaren Befehle (besonders die SetUID- und SetGID-Rechte) sollten überprüft und angepasst werden.

6.2 Installation des Linux-Systems

Ich möchte hier nicht eine Empfehlung für eine Linux-Distribution aussprechen, denn ich denke, dass es *die beste* Distribution nicht gibt. Sicherlich gibt es Unterschiede im Design und in der Anwendung, und es mag subjektive Gründe geben, warum Sie und ich die eine oder andere Distribution vorziehen. Allerdings können Sie jede Distribution für eine Firewall einsetzen. Es ist nicht erforderlich, eine spezielle Distribution wie Fli4L[2] oder IPCop[3] zu nutzen. Wenn Sie diese Distributionen nicht kennen, sind sie sicherlich einen Blick wert. Ob es allerdings Sinn macht, bei dem Aufbau einer Firewall eine neue Distribution zu lernen, möchte ich bezweifeln. Verwenden Sie die Distribution, die Sie am besten kennen und konfigurieren können. Lediglich dann, wenn Sie auf unüberwindbare Probleme stoßen, die der Distribution zuzuschreiben sind, würde ich Ihnen raten, eine andere Distribution zu testen.

Dennoch gibt es einige allgemeine Hinweise für die Installation des Systems.

Zunächst sollten Sie sich Gedanken über die Ausfallsicherheit des Systems machen. Die Firewall stellt Ihre Internetverbindung dar. Wenn diese ausfällt, kann das verheerende Konsequenzen für Ihr Geschäftsmodell haben, wenn Ihre gesamte Kommunikation (E-Mail, VoIP etc.) auf dem Internet basiert. Falls es sich lediglich um eine Firewall für private Zwecke handelt, ist das sicherlich nicht so tragisch, aber auch hier kann ein Ausfall des Internets unerwünscht sein.

Wählen Sie also eine möglichst robuste Hardware. Wenn das System ununterbrochen laufen soll, achten Sie besonders auf die Luftzufuhr und Kühlung. Wählen Sie lieber einen stromsparenden Prozessor, der auch weniger Abwärme erzeugt als die neueste Heizplatte. Ein reiner Paketfilter benötigt nicht einen Pentium 4 mit 3 GHz. Wenn Sie auch noch andere Funktionen (z.B. VPN oder Proxy) auf demselben System betreiben möchten, sieht das natürlich anders aus. Aber meist reicht auch hier noch ein einfacher stromsparender Prozessor.

Investieren Sie Ihr Geld lieber in zwei Festplatten und wenn möglich in ein Mainboard mit redundanter Stromversorgung über zwei Netzteile. Falls es Ihr Budget hergibt, ist auch ein Hardware-Raid-Controller sinnvoll. Achten Sie aber darauf, dass der Controller auch von Linux unterstützt wird. Ein Festplattenspiegel mit Raid 1 reicht vollkommen aus. Ein Raid 5 ist nicht erforderlich.

2 http://www.fli4l.de
3 http://www.ipcop.org

CE Es werden in der Folge SELINUX, AppArmor etc. behandelt. Sollte man dies evtl. hier erwähnen?

TS a Hier ein "s" entfernt.

Selbst wenn Sie nicht das Geld für einen Raid-Controller ausgeben möchten, empfehle ich den Einsatz von zwei Festplatten mit einem Software-Raid 1. In einem Fehlerfall ist die Wiederherstellung zwar etwas aufwendiger, jedoch verlieren Sie nicht Ihre Daten. Die meisten großen Distributionen können bereits bei der Installation ein Software-Raid einrichten und installieren. Ansonsten finden Sie in dem Linux Software-Raid-Howto[4] entsprechende Hinweise für die Einrichtung.

Der wesentliche Vorteil bei einem Raid-System ist die Tatsache, dass das System auch bei Ausfall einer Festplatte weiterarbeitet. Aus demselben Grund empfehle ich auch die Verwendung redundanter Netzteile. Die Netzteile und die Festplatten sind am häufigsten für den Ausfall eines Systems verantwortlich. Sind beide redundant vorhanden, arbeitet das System trotz Ausfall einer Komponente weiter.

Wenn Sie das ganze Firewall-System redundant implementieren möchten, sollten Sie Kapitel 27TS[b] über den HA-Firewall-Cluster lesen. Dort wird die Konfiguration eines Firewall-Clusters auf der Basis von Linux beschrieben.

Sobald Sie die Hardware ausgewählt und angeschafft haben, müssen Sie sich mit der Installation beschäftigen. Hier sind zunächst die Partitionierung und die Paketauswahl wichtig. Bei der Partitionierung sollten Sie bedenken, dass die spätere Firewall nur einen geringen Platz für die binären Pakete benötigt. Jedoch werden die Systeme Protokolldateien erzeugen, die schnell sehr groß werden können. Dies sollten Sie in der Partionierung berücksichtigen. Folgendes Schema hat sich bewährt:

- `/boot` 128 Mbyte. So haben Sie immer ausreichend Platz für Kernel-Updates.
- `/usr` 1 Gbyte. Sie installieren nur wenige Pakete.
- `/` 512 Mbyte. Die Konfigurationsdateien (`/etc/`) und Geräte (`/dev`) benötigen nicht viel Speicher. Bei Bedarf können Sie auch mehr zuteilen.
- `/tmp` 256 Mbyte. Der temporäre Speicher sollte auf einer eigenen Partition liegen, sodass er nicht die Root-Partition füllen kann.
- `/var` Rest. Hier liegen die Protokolldateien. Dieses Verzeichnis wird bei der Verwendung der Firewall wachsen. Eventuell ist es sinnvoll, das Verzeichnis `/var/log` auch noch auf eine eigene Partition auszulagern.

Bei der Paketauswahl sollten Sie dann eine Minimalinstallation wählen. Sollte der oben angegebene Platz nicht ausreichen, müssen Sie die eine oder andere Partitionsgröße anpassen. Allerdings kenne ich keine Distribution, bei der das aktuell der Fall wäre.

Nach der Installation sollten Sie sich für den unwahrscheinlichen Fall, dass beide Festplatten ausfallen, mit einer Disaster-Recovery-Lösung wappnen. Dies ist umso wichtiger, wenn Sie nur eine Festplatte verwenden. Mehrere Disaster-Recovery-Lösungen sind mir bekannt und werden von mir verwendet:

- Mondo-Rescue: `http://www.mondorescue.org/`
- Mkcdrec: `http://mkcdrec.sourceforge.net/`

4 `http://www.tldp.org/HOWTO/Software-RAID-HOWTO.html`

TS[b] Oder ist ein Unterabschnitt gemeint? Bitte bestätigen.

» Rear (Relax and Recover): `http://rear.sourceforge.net/`

» Alternativ auch Partimage: `http://www.partimage.org`

Die drei ersten Programme analysieren das System und erzeugen ein oder mehrere bootfähige CD-Images, die für die Wiederherstellung des Systems genutzt werden können. Dabei wird das System nicht blockweise kopiert, sondern die Programme analysieren die Partitionierung und Formatierung, sichern alle Dateien und können diese dann auf einer neuen, mindestens gleich großen Festplatte wiederherstellen. Beide Systeme können mit Software-Raid- und Logical-Volume-Manager-Partitionen (LVM) umgehen.

Es empfiehlt sich, eine der beiden Lösungen zu installieren und zu testen. Später sollten Sie dann nach jeder wesentlichen Änderung des Systems oder vielleicht monatlich ein neues Disaster-Recovery-Image erzeugen. Achten Sie darauf, dass die Protokolle nicht gesichert werden. Dies bläht die Images nur unnötig auf.

6.3 Updates

Wenn Sie Ihr Linux-System mithilfe von CDs installieren, dann sind die darauf enthaltenen Dateien bei der Installation sicherlich zumindest teilweise veraltet. Bei vielen Paketen wurden Fehler gefunden, und bei einigen Paketen sind die Fehler möglicherweise sogar sicherheitsrelevant. Sie sollten daher, direkt anschließend an die Installation, die neuesten Updates installieren. Sehr einfach ist das, wenn Ihre Distribution dafür einen Mechanismus anbietet. Dies ist unter anderem bei Debian, OpenSUSE, Ubuntu, RedHat und Fedora der Fall. Die Anwendung dieser Mechanismen wird weiter unten erläutert.

Sie sollten sich auch überlegen, ob Sie diesen Update-Mechanismus automatisch einbinden wollen. Für mich persönlich habe ich auf vielen Systemen diese Entscheidung getroffen. So werden auf meinen Systemen die Updates stündlich geprüft und eingespielt. Manch einer mag nun die Nase rümpfen. Was passiert, wenn das Update fehlschlägt? Nun, ich habe für mich eine Risikoabwägung durchgeführt. Was passiert, wenn eine Sicherheitslücke bekannt wird und ich nicht rechtzeigt das Update durchführe? Der Schaden ist ungemein größer, als wenn automatisch ein Update durchgeführt wird, dieses fehlschlägt und der Dienst anschließend für einige Zeit nicht zur Verfügung steht. Er kann zumindest nicht mehr angegriffen werden.

Außerdem sollten Sie sich überlegen, wie Sie vorgehen, wenn Sie selbst manuell das Paket aktualisieren. Wahrscheinlich werden Sie das Update von Ihrer Distribution installieren. Im Grunde führen Sie dieselben Schritte durch, nur manuell und später. Das Update kann genauso fehlschlagen. Sie müssen das System genauso reparieren.

Ich persönlich habe erst ein einziges Mal eine schlechte Erfahrung mit der automatischen Aktualisierung der Pakete gemacht. Dabei handelte es sich um das `bind`-Paket der RHEL-3 Distribution. Nachdem das Update eingespielt wurde, wurde der Nameserver gestoppt, aber nicht wieder gestartet. Da es sich um einen primären Nameserver handelte, für den es noch weitere sekundäre Nameserver gab, fiel dies nicht sofort auf. Erst nach zwei Wochen, als die sekundären Nameserver aufgrund des fehlenden primären Nameservers ihre Arbeit einstellten, funktionierte die Namensauflösung nicht mehr. Wenn Sie wie ich nachts schlafen, ab und

zu in den Urlaub fahren und nicht 24 Stunden am Tag ein Support-Team beschäftigen möchten, kann ich Ihnen nur empfehlen, eine automatische Aktualisierung der kritischen Systeme einzurichten.

6.3.1 Debian/Ubuntu-Updates

Auf einem Debian-System ist ein Update einfach durchzuführen. Hierzu rufen Sie nacheinander die folgenden Befehle auf:

```
/usr/bin/apt-get update -q -y
/usr/bin/apt-get upgrade -q -y
```

Der erste Befehl aktualisiert die lokal vorgehaltenen Paketlisten der verfügbaren Programmpakete. Der zweite Befehl aktualisiert die Pakete anhand dieser Liste. Die Option `-q` unterdrückt die Ausgabe von unnötigen Informationen (quiet), und die Option `-y` beantwortet automatisch alle möglichen Fragen mit Ja (yes).

6.3.2 OpenSUSE-Updates

Bei SUSE kann bei älteren Versionen das Yast-Online-Update (YOU, siehe Abbildung 6.1) über das Werkzeug Yast konfiguriert werden. Hierzu müssen Sie aber auch sicherstellen, dass

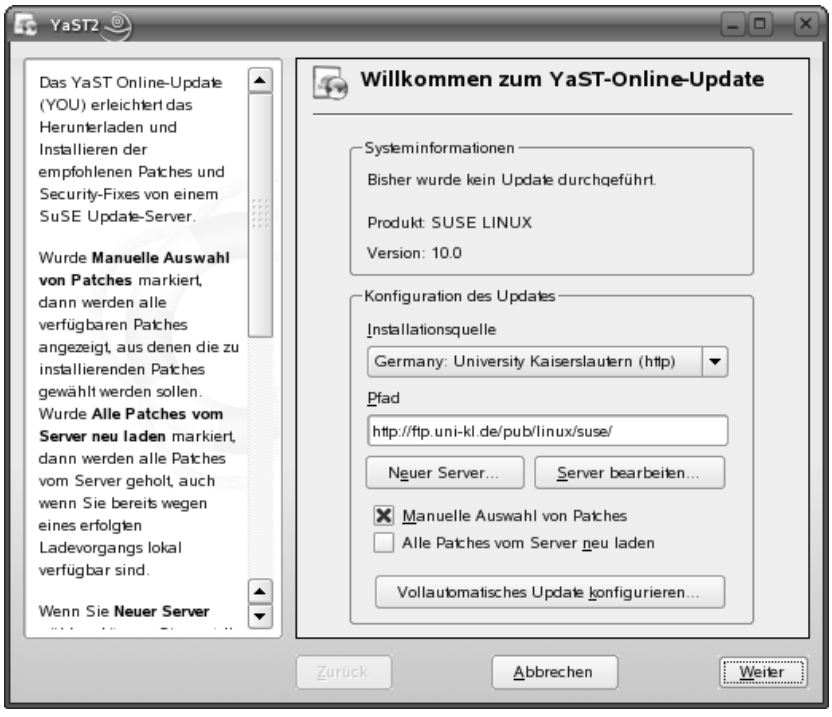


Abbildung 6.1: **Yast ermöglicht das Online-Update.**

das Online-Update-Paket installiert wurde. Bei dem Einsatz eines SUSE-Linux-Enterprise-Servers (SLES) benötigen Sie für das Online-Update auch noch ein Kennwort.

Auf den neueren Systemen nutzen Sie einfach `zypper upgrade`.

6.3.3 Fedora-Updates

Bei der freien Fedora-Distribution ist ein Update der Pakete sehr einfach mit dem Befehl `yum` möglich. Ein `yum update` aktualisiert das System mit den aktuellen Paketen. Die Option `-y` beantwortet auch hier alle Fragen mit Ja.

6.4 Deaktivieren überflüssiger Dienste

Sobald Sie die Installation des Systems abgeschlossen und eventuelle Updates eingespielt haben, sollten Sie überflüssige Dienste deaktivieren. Wie finden Sie nun diese Dienste? Im Grunde gibt es drei verschiedene Varianten, wie auf einem typischen Linux-System ein Dienst automatisch gestartet wird.

Startskripte. Viele Linux-Distributionen nutzen SysVInit-Skripte, um die Dienste zu starten. Einige, vor allem kleinere Distributionen nutzen ein einziges Startskript.

Internet-Super-Server. Der Internet-Super-Server `inetd` oder `xinetd` startet Netzwerkdienste bei Bedarf.

Cron-Daemon. Der Crond-Daemon startet automatisch Programme zu bestimmten Zeiten.

Die Konfiguration dieser Varianten wird weiter unten erläutert.

Woran erkennen Sie nun einen überflüssigen Dienst? Ich hoffe, dass Sie das bei vielen Diensten selbst erkennen können, da Sie wissen, was der Dienst treibt. Falls das nicht der Fall ist, sollten Sie versuchen, es herauszufinden. Hilfreich sind hier die Manpage, die Dokumentation des Pakets[5] und natürlich Google. Falls Ihnen diese Informationen nicht weiterhelfen, können Sie versuchen, zum Test den Dienst zu deaktivieren. Falls bei einem Reboot keine Fehlermeldung auftritt und alle wichtigen Funktionen zur Verfügung stehen, war der Dienst nicht erforderlich.

6.4.1 Startskripte

Die meisten Distributionen nutzen SysVInit-Startskripte. Diese Skripte befinden sich in dem Verzeichnis `/etc/init.d`. Ob ein Dienst gestartet wird oder nicht, wird über Verknüpfungen in den Verzeichnissen `/etc/rc.d/rc[0-6].d` konfiguriert. Anstatt diese Verknüpfungen jedoch von Hand zu modifizieren, sollten Sie den Befehl `chkconfig` verwenden, der auf den meisten Distributionen[6] vorhanden ist. Falls dieser Befehl bei Ihnen nicht existiert, lesen Sie bitte in

5 Mit dem Befehl `rpm -qif /etc/init.d/<dienst>` zeigen Sie die Informationen über das RPM-Paket des Dienstes an.

6 Unter Debian Lenny können Sie diesen Befehl nachinstallieren.

der Dokumentation der Distribution nach. Mit dem Befehl `chkconfig --list` können Sie den aktuellen Zustand der Startskripte anzeigen.

```
# chkconfig --list postfix
postfix 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

Mit `chkconfig <dienst> off` schalten Sie den entsprechenden Dienst ab. Alternativ zu den SysVInit-Skripten gibt es vor allem auf kleinen Distributionen ein einzelnes Startskript, das die Dienste startet. Hier müssen Sie in die Datei schauen und die entsprechenden Zeilen durch Kommentieren deaktivieren. Teilweise (z.B. bei OpenWRT) existiert auch ein Verzeichnis `/etc/init.d`, und jedes Skript in diesem Verzeichnis wird automatisch aufgerufen. Um einen Dienst zu deaktivieren, genügt es dann meist, die Ausführrechte von der Datei zu entfernen.

Änderungen in den Startskripten machen sich erst bei einem Neustart oder dem Wechsel des Runlevels bemerkbar.

6.5 Internet-Super-Server

Fast jede Linux-Distribution verfügt über den Internet-Super-Server `inetd` oder `xinetdq`. Wird dieser Dienst über ein Startskript gestartet, so startet er bei Bedarf, entsprechend seiner Konfiguration, weitere Dienste. Das bedeutet, dass Sie den Telnet-Server in der Ausgabe von `ps -ef` nicht sehen, obwohl er verfügbar ist. Er wird erst bei Bedarf von dem `(x)inetd` gestartet.

Der Inetd[TS c] verfügt über eine Konfigurationsdatei `/etc/inetd.conf`, in der pro Zeile ein Dienst konfiguriert wird. Sie schalten die entsprechenden Dienste durch Auskommentieren der Zeile ab.

Bei dem Xinetd[TS c] werden die Dienste meist über einzelne Dateien in dem Verzeichnis `/etc/xinetd.d` gesteuert. Diese Dateien enthalten eine Zeile `enable=` oder `disable=`. Durch Setzen des ensprechenden Wertes (`enable=no` bzw. `disable=yes`) deaktivieren Sie den Dienst.

Natürlich müssen Sie nach einer Modifikation der Konfiguration den Internet-Super-Server neu starten.

6.5.1 Der Cron-Daemon

Schließlich können Programme auch automatisch von dem Crond-Daemon[TS d] gestartet werden. Der Crond-Daemon wird über Cron-Tabellen konfiguriert. Auf den meisten Linux-Systemen existieren zwei Arten von Cron-Tabellen.

Jeder Benutzer kann eine persönliche Cron-Tabelle anlegen, wenn ihm dies über die Dateien `/etc/cron.allow` oder `/etc/cron.deny` erlaubt wird. Existiert die erste Datei, so dürfen nur die dort aufgeführten Benutzer eine Cron-Tabelle anlegen. Existiert die zweite Datei, so dürfen alle Benutzer – außer den dort aufgeführten – eine Cron-Tabelle anlegen. Diese Cron-Tabelle

[TS c] Sollte die Formatierung hier auch `Inetd`/`Xinetd` sein?

[TS d] Bitte unterscheiden: cron, crond, cron-Daemon.

wird mit dem Befehl `crontab -e` angelegt und editiert. Mit dem Befehl `crontab -l` zeigen Sie Ihre eigene Cron-Tabelle an, und mit `crontab -r` löschen Sie diese.

Die Syntax dieser Datei ist recht einfach. Es handelt sich um eine Tabelle mit sechs Spalten. Die ersten fünf Spalten enthalten die Definition eines Zeitpunkts: Minuten, Stunden, Tag, Monat und Wochentag. Dabei wird der Wochentag ebenfalls mit einer Ziffer angegeben (0,7 = Sonntag). Der Cron-Daemon vergleicht jede Minute die aktuelle Systemzeit mit der eingetragenen Zeit und führt bei Übereinstimmung den Befehl in der sechsten Spalte aus. Betrachten Sie folgendes Beispiel:

```
#Min  Std Tag Monat WoTag Befehl
  0    5   *    *     *   /bin/test.sh
*/30 7-18  *    *    1-5  /bin/getmail.sh
```

Das Skript `/bin/test.sh` wird um 5:00 Uhr an jedem Tag (*), in jedem Monat (*) und unabhängig vom Wochentag (*) ausgeführt. Das Skript `/bin/getmail.sh` wird von 7:00 Uhr bis 18:30 alle 30 Minuten (wenn die Division ohne Rest aufgeht) von Montag (1) bis Freitag (5) unabhängig vom Datum ausgeführt.

Zusätzlich zu den Cron-Tabellen der Benutzer existiert auf einem modernen Linux-System auch eine System-Cron-Tabelle `/etc/crontab`. Diese Cron-Tabelle gleicht stark den Tabellen der Benutzer, weist jedoch eine zusätzliche Spalte zwischen der fünften und sechsten Spalte auf. Hier kann der Benutzer angegeben werden, in dessen Kontext der Cron-Daemon den Befehl in der nun siebten Spalte aufrufen soll.

```
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Die obigen Zeilen stammen von einer Fedora Linux-Distribution. Der hier von dem Cron-Daemon aufgerufene Befehl ist `run-parts`. Diesem Befehl werden in Abhängigkeit von der Zeit unterschiedliche Verzeichnisse als Argument übergeben. Der Befehl durchsucht das angegebene Verzeichnis und ruft alle Befehle in diesem Verzeichnis auf. So wird immer eine Minute nach der vollen Stunde jede ausführbare Datei in dem Verzeichnis `/etc/cron.hourly` aufgerufen. Ähnlich wird immer am Sonntag (0) um 4:22 Uhr das Verzeichnis `/etc/cron.weekly` abgearbeitet.

Um zu erkennen, welche Programme aufgerufen werden, sollten Sie daher die Verzeichnisse analysieren und überflüssige Programme entfernen oder ihnen das Ausführrecht entziehen. Dieses Verhalten ist eigentlich bei allen Linux-Distributionen ähnlich.

Änderungen an den Cron-Tabellen der Benutzer sind sofort aktiv. Ein Neustart des Cron-Daemons ist nur bei einer Modifikation der Datei `/etc/crontab` erforderlich. Auch Änderungen in den Verzeichnissen `/etc/cron.*` erfordern keinen Neustart.

6.6 Entfernen überflüssiger Software

Nachdem Sie überflüssige Dienste deaktiviert haben, können Sie nun die entsprechenden Pakete und vielleicht auch noch weitere Pakete entfernen. Das Entfernen von überflüssigen Paketen ist unter Linux seit der Einführung der Paketverwaltungssysteme recht einfach geworden, da diese die Abhängigkeiten überwachen und im Zweifelsfall die Deinstallation verhindern. Beginnen Sie am einfachsten damit, dass Sie alle Pakete auflisten. Dies geht auf einer RPM-basierten Distribution recht einfach mit `rpm -qa`. Auf einer Debian-Distribution zeigt der Befehl `dpkg -l` die installierten Pakete an. Untersuchen Sie die Liste, und versuchen Sie, Pakete zu deinstallieren, die Sie nicht benötigen. Sobald das Paket noch von weiteren Paketen benötigt wird, wird die Paketverwaltung Ihnen einen Abhängigkeitskonflikt anzeigen. Entscheiden Sie dann, ob Sie diese Pakete auch deinstallieren wollen.

Wenn Sie diese Entscheidungen mangels Erfahrung noch nicht treffen können oder wollen, sollten Sie zumindest sicherstellen, dass Sie möglicherweise vorhandene C-Compiler von dem System entfernen. In der Vergangenheit sind bereits mehrfach Würmer aufgetreten, die nach einem Einbruch auf einem System zunächst sich selbst übersetzt haben. Fehlt der C-Compiler, ist die Verbreitung eines solchen Wurms gestoppt.

6.7 Sicherheit auf Dateisystemebene

Die meisten Linux-Distributionen sind nicht speziell auf Firewall-Zwecke ausgerichtet. Ihre Zielsetzung ist eher ein Multifunktionssystem. Es soll möglichst einfach sein, sowohl Netzwerkdienste anzubieten als auch als normaler Benutzer über die grafische Oberfläche eine DVD abzuspielen und zu brennen. Hierfür müssen diese Systeme viele Tricks anwenden, damit das auch so unproblematisch funktioniert, wie es der Anwender möchte. So hat normalerweise ein einfacher Benutzer nicht das Recht, auf das CD- oder DVD-Laufwerk zuzugreifen, und er besitzt dort schon gar keine Schreibrechte. Auch ein einfaches Ping darf der normale Benutzer nicht starten. Diese und viele weitere Aktionen dürfen nur von dem privilegierten Benutzer root durchgeführt werden. Damit dennoch ein Benutzer den Befehl benutzen darf, wird entweder der Benutzer für die Ausführung des Befehls mit root-Privilegien ausgestattet oder die Rechte der Ressource, auf die der Benutzer zugreifen möchte, werden entsprechend angepasst.

Die grundsätzliche Problematik ist schon sehr alt. Wie soll zum Beispiel ein Benutzer sein Kennwort ändern, wenn er keine Schreibrechte an der Datei `/etc/passwd` oder `/etc/shadow` hat? Daher wurde sehr früh in der Geschichte von UNIX (1971) bereits ein Mechanismus geschaffen, der es erlaubt, für die Ausführung eines Befehls dem Prozess erweiterte Privilegien zu übertragen. Das SetUID- und das SetGID-Recht waren geboren. Verfügt ein Befehl über diese Rechte, so werden für die Ausführung des Befehls die Rechte des Eigentümers (SetUID) oder der Gruppe (SetGID) auf den Prozess übertragen. Sie erkennen diese Rechte an einem kleinen `s`.

```
-r-s--x--x 1 root root 18840  7. Mär 2005  /usr/bin/passwd
-r-xr-sr-x 1 root tty   9752 27. Apr 17:42 /usr/bin/wall
```

Wenn ein Benutzer den Befehl `passwd` aufruft, erbt er für die Ausführung die Rechte des Eigentümers root. Beim Aufruf des Befehls `wall` erbt er die Rechte der Gruppe tty. Es ist dann Aufgabe der Befehle zu prüfen, dass der Anwender auch tatsächlich nur den vorgesehenen Vorgang durchführt und die Rechte nicht missbraucht. Leider wiesen die Befehle in der Vergangenheit immer wieder Fehler auf, sodass für die Zukunft weitere Fehler nicht ausgeschlossen werden können.

Daher sollten Sie sich fragen, ob auf Ihrer Firewall überhaupt normale unprivilegierte Benutzer arbeiten sollen. Wenn Sie bei dem Aufruf der Befehle bereits root sind oder die Rechte immer über `sudo` (siehe unten) erhalten, ist es nicht nötig, dass diese Rechte gesetzt sind.

Diese Rechte haben den zusätzlichen Nachteil, dass Sie jedem Benutzer (auch einem Systembenutzer wie nobody) für die Ausführung die Privilegien übertragen. Sie sollten daher diese Rechte entfernen. Um das System nach derartigen Befehlen abzusuchen, können Sie die folgenden Befehle verwenden:

```
# find / -perm +4000 -ls
# find / -perm +2000 -ls
```

Sie können die Rechte mit `chmod u-s <datei>` und `chmod g-s <datei>` entfernen. Prüfen Sie auch hier anschließend mit einem Neustart, ob das System noch so arbeitet, wie Sie es sich vorstellen.

INFO

Sudo.

Sinnvollerweise legen Sie auf der Firewall lediglich für jeden Firewall-Administrator ein personalisiertes Benutzerkonto an, das Sie mit individuellen Kennwörtern versehen. Die Administration des Systems kann dann mit dem Befehl `sudo` erfolgen. Mit diesem Befehl können Sie einem bestimmten Benutzer bei der Ausführung eines bestimmten Befehls erweiterte Privilegien zuweisen. Bei der ersten Ausführung verlangt der Befehl zusätzlich noch die Authentifizierung des Benutzers, die Sie aber auch abschalten können.

Sie konfigurieren den Befehl `sudo` in der Datei `/etc/sudoers`. Diese Datei sollten Sie jedoch nicht direkt, sondern mit dem Befehl `visudo` editieren. Dieser Befehl prüft die Syntax und warnt Sie vor dem Abspeichern bei Fehlern in der Datei.

Um nun eine Gruppe von Administratoren mit root-Privilegien auszustatten, können Sie die folgenden Zeilen verwenden:

```
User_Alias ADMIN = spenneb, thorsten
ADMIN ALL=(root) ALL
```

Die erste Zeile definiert einen Alias `ADMIN` für die Benutzer `spenneb` und `thorsten`. Die zweite Zeile erlaubt diesen Benutzern, auf allen Rechnern als `root` jeden Befehl auszuführen. Die Einschränkung der Rechner ermöglicht es Ihnen, eine zentrale Datei zu erzeugen und für mehrere Systeme zu verwenden und zu verteilen. Dann können Sie hier den Rechnernamen angeben, für den diese Zeile gelten soll.

Die Benutzer `spenneb` und `thorsten` benötigen nun nicht mehr das root-Kennwort. Sie können selbst jeden Befehl ausführen. Bei der ersten Verwendung und nach Ablauf von 5 Minuten müssen Sie sich mit Ihrem eigenen Kennwort authentifizieren. Jeder Zugriff wird außerdem protokolliert:

```
Sep 11 15:16:59 bibo sudo: spenneb : TTY=pts/2 ; PWD=/buch/fw_buch/buch ; ↵
        USER=root ; COMMAND=/usr/bin/tail /var/log/messages
```

6.8 Sicherheit beim Bootvorgang

Damit Ihre Bemühungen, das System zu sichern, nicht vergebens sind, sollten Sie auch den Bootvorgang des Systems in Ihre Betrachtungen mit einbeziehen.

Zunächst sollten Sie sorgfältig den Aufstellungsort für das System wählen. Achten Sie darauf, dass kein Unbefugter physikalischen Zugang zu dem System erhält. Wählen Sie also nicht die Besenkammer am Ende des Flurs, sondern einen Raum, den Sie abschließen können.

Zusätzlich sollten Sie darauf achten, dass die Bootreihenfolge im BIOS es nicht ermöglicht, das System von Diskette, CD oder USB-Stick zu booten. Sichern Sie das BIOS gegen unbefugte Änderungen mit einem Kennwort.

Achten Sie auch darauf, dass der Boot-Manager mit einem Kennwort gegen unbefugte Änderungen geschützt ist. Erfreulicherweise bieten moderne Distributionen Ihnen dies direkt bei der Installation an. Falls dies nicht der Fall ist, fügen Sie ein Kennwort zur Konfiguration hinzu.

Bei dem Boot-Manager Grub können Sie mit dem Befehl `grub-md5-crypt` ein Kennwort verschlüsseln und in der Grub-Konfigurationsdatei mit dem Parameter `password --md5 1....` angeben. Anschließend ist für jede Modifikation des Bootverhaltens das Kennwort anzugeben. Auf Red Hat- und Fedora Linux-Distributionen existiert zusätzlich noch die Möglichkeit, während des Startens der Dienste diese interaktiv zu bestätigen. Dazu müssen Sie, während der Init-Prozess startet, `I` eingeben. Sie sollten auch diese Möglichkeit abschalten, damit niemand bei dem Boot einen Dienst überspringen kann. Setzen Sie hierzu in der Datei `/etc/sysconfig/init` die Variable `Prompt=no`.

Bei dem Boot-Manager Grub2 erfolgt die Konfiguration etwas anders. Hier tragen Sie am Ende der Datei `/etc/grub.d/00_header` folgende Zeilen ein:

```
set superusers="user1"
password user1 password1
```

Nun kann nur noch der Superuser *user1* die Einträge editieren. Grub2 kann auch nur einzelnen Benutzern den Boot eines Menüeintrags erlauben. Dies soll hier aber nicht betrachtet werden.

Bei dem heute weniger gebräuchlichen Boot-Manager Lilo können Sie einfach zwei Zeilen an den Anfang der Datei `/etc/lilo.conf` anfügen:

```
password=geheimes_Kennwort
restricted
```

Leider müssen Sie das Kennwort in Klartext angeben. Sie sollten daher darauf achten, dass keiner außer root diese Datei lesen darf. Der Parameter `restricted` sorgt dafür, dass lediglich Modifikationen des Bootvorgangs ein Kennwort verlangen. Nach der Modifikation der Datei müssen Sie Lilo neu installieren. Rufen Sie hierzu `lilo -v` auf.

Nun sollte keine unbefugte Modifikation des Bootvorgangs mehr möglich sein.

6.9 Bastille-Linux

Die von mir bisher beschriebenen Schritte zur Härtung können nur einen allgemeinen Weg aufzeigen. Je nachdem, welche Distribution Sie verwenden, sind vielleicht noch weitergehende Schritte erforderlich, oder Sie können den einen oder anderen Schritt überspringen. Es ist in jedem Fall ein großer Aufwand, alle Schritte richtig und in der richtigen Reihenfolge reproduzierbar und dokumentiert durchzuführen. Bastille-Linux hilft Ihnen dabei.

Bastille-Linux ist ein Härtungswerkzeug für Linux- und UNIX-Systeme. Es unterstützt RedHat Linux, RedHat Enterprise Linux, Fedora, SuSE, Mandrake, Gentoo, Debian, HP-UX und MacOS X. Sie können mit diesem Werkzeug einfach, reproduzier- und automatisierbar Dienste und Systemeinstellungen konfigurieren. Es schaltet unnötige Dienste ab und kann sogar Dienste in einem Chroot laufen lassen.

STOP

Leider wurde das Werkzeug seit 2008 nicht mehr weiterentwickelt, sodass es die aktuellen Distributionen nicht erkennt. Ich habe aber die Hoffnung, dass dieses Werkzeug wieder zum Leben erweckt wird. Daher möchte ich den Hinweis in diesem Buch lassen. Die in der ersten Auflage vorhandene ausführliche Beschreibung habe ich aber entfernt.

6.10 SELinux und AppArmor

Leider unterstützt ein klassisches Linux-System nur ein sehr einfaches Rechte-Modell. So können der Benutzer root und der Eigentümer einer Datei die Rechte dieser Datei modifizieren. Dabei können die Rechte für den Eigentümer, eine Gruppe und alle weiteren Benutzer definiert werden. Mit modernen Dateisystemen können auch die POSIX-ACLs verwendet werden (siehe Abschnitt 6.11)TS. Dann ist es möglich, auch mehreren Benutzern unterschiedliche Rechte zuzuweisen.

INFO

*Auch beim Einsatz der POSIX-ACLs haben Sie lediglich die Rechte `r`, `w` und `x`. Sie können also lediglich die Rechte für das Lesen, Schreiben und Ausführen verwalten. Weitergehende Rechte existieren nicht.*TS[a]

TS Bitte diesen Verweis bestätigen.

TS[a] Bitte neuen "Tip" bestätigen.

Wünschenswert wäre die Möglichkeit, die Privilegien des Benutzers root einzelnen normalen Benutzern für bestimmte Aufgaben zukommen zu lassen. Obwohl dies seit Jahren mit dem Capability-System möglich ist, existierte bis zur Einführung von AppArmor und SELinux keine sinnvolle Administrationsoberfläche, die zum festen Bestandteil moderner Distributionen geworden ist. Alle anderen verschiedenen Ansätze (LIDS, grsecurity, RSBAC etc.) sind nur als Patch verfügbar. Diese Systeme werden häufig auch als Mandatory-Access-Control-Systeme (MAC) bezeichnet. Dies setzt sie von dem klassischen UNIX als Discretionary-Access-Control-System (DAC) ab. UNIX ist ein DAC-System, da hier der Eigentümer einer Datei ihre Rechte definiert (die Rechte werden entsprechend der Diskretion des Eigentümers definiert). Der Eigentümer kann hierbei auch leicht Fehler machen. Bei einem MAC-System werden die Rechte zentral für alle Dateien verwaltet. Selbst wenn der Eigentümer root fälschlicherweise einem anderen Benutzer das Leserecht an der Datei `/etc/shadow` zuweisen würde, könnte das MAC-System, das von einem über root stehenden Superuser verwaltet wird, diesen Zugriff noch verhindern.

Alle MAC-Systeme verfügen über einen derartigen von root unterschiedlichen Superuser. Häufig wird hierfür ein weiteres Kennwort verlangt. Teilweise ist zur Laufzeit keine Modifikation des MAC-Systems möglich.

Alle MAC-Systeme für Linux stellen nur zusätzliche Systeme dar. Das bedeutet, dass bei dem Zugriff auf eine Datei sowohl das DAC- als auch das MAC-System den Zugriff erlauben müssen.

Derartige Systeme können stark die Sicherheit des Betriebssystems erhöhen. Leider ist eine komplette Besprechung dieser Systeme im Rahmen dieses Buches nicht möglich und sinnvoll. Ich habe AppArmor und SELinux aber ein weiteres Buch im Addison-Wesley Verlag gewidmet[CE b].

Verwenden Sie einfach eine Distribution, die bereits ein derartiges MAC-System vorkonfiguriert mitbringt. Aktuell sind das zum Beispiel Fedora und Debian (SELinux) bzw. OpenSUSE und Ubuntu (AppArmor).

6.11 POSIX-ACLs

Die klassischen UNIX- und Linux-Dateisysteme erlauben es nur, die Rechte für den Eigentümer, eine Gruppe und den Rest (others) zu verwalten. Eine weitere Abstufung ist nicht möglich. Es kann nicht eine Gruppe mit Leserechten und eine weitere Gruppe mit Schreibrechten ausgestattet werden. Viele andere Betriebssysteme bieten diese Möglichkeit. Für UNIX wurden die POSIX-ACLs geschaffen, um dieses Problem zu beheben. Alle aktuellen Linux-Dateisysteme unterstützen inzwischen auch diese erweiterten ACLs. Um die ACLs zu nutzen, müssen Sie das Dateisystem mit der Option `acl` mounten. Hierzu tragen Sie die Option in der Datei `/etc/fstab` in der entsprechenden Spalte ein oder geben sie bei dem Mount-Vorgang

CE b Evtl. Fußnote mit Titel und ISBN angeben?

Editor's or typesetter's annotations (will be removed before the final TeX run)

auf der Kommandozeile ein.[7] Um das Dateisystem `/home` mit ACL-Optionen neu zu mounten, verwenden Sie den folgenden Befehl:

```
mount -o remount,acl /home
```

Nun können Sie für Dateien erweiterte ACLs definieren. Hierfür gibt es die Befehle `setfacl` und `getfacl`. Leider unterstützen die Befehle `ls` und `chmod` die ACLs nicht.

```
# ls -l datei
-rw-r--r-- 1 root root 0 19. Sep 18:17 datei
# setfacl -m u:test:rw datei
# ls -l datei
-rw-rw-r--+ 1 root root 0 19. Sep 18:17 datei
# getfacl datei
# file: datei
# owner: root
# group: root
user::rw-
user:test:rw-
group::r--
mask::rw-
other::r--

# setfacl -m u:test:rwx datei
# setfacl -m m::rx datei
# ls -l datei
-rw-r-xr--+ 1 root root 0 19. Sep 18:17 datei
# getfacl datei
# file: datei
# owner: root
# group: root
# user::rw-
# user:test:rwx       effective:r-x
# group::r--
# mask::r-x
# other::r--
```

Sobald Sie eine ACL einer Datei hinzugefügt haben, zeigt der Befehl `ls` bei den Rechten ein + an. Dieses zeigt Ihnen, dass weitere ACLs verborgen sind. Außerdem wird an der Stelle, an der üblicherweise die Gruppenrechte angezeigt werden, nun die Maske angezeigt. Diese Maske definiert die maximalen Rechte, die mit den ACLs vergeben werden können. Alle über die Maske per ACL hinausgehenden Rechte werden automatisch nicht aktiv.

7 OpenSUSE aktiviert diese Option bereits selbst per Default. Bei Fedora ist diese Option in das Dateisystem fest einkompiliert. Dort können diese Funktionen sofort ohne Änderungen genutzt werden.

Wenn Sie ACLs einsetzen, sollten Sie darauf achten, dass das Dateisystem immer mit der Option `acl` gemountet wird. Ansonsten sind die ACLs nicht aktiv. Außerdem sollten Sie Ihre Backup-Programme überprüfen. Viele Backup-Programme können nicht mit ACLs umgehen. So sichert `tar` die ACLs nicht! Anstelle von `tar` können Sie aber `star`[8] verwenden.

6.12 Intrusion-Detection- und -Prevention-Systeme

Ist Ihre Firewall sicher? Natürlich haben Sie bereits einige Erfahrung in der Konfiguration von Firewalls oder werden dieses Buch aufmerksam lesen. Aber sind Sie sicher, dass Sie keinen Fehler machen?

Vielleicht kommt ein potenzieller Angriff auch gar nicht von außen, sodass die Firewall ihn nicht abwehren kann. Viele Angriffe werden von innen ausgeführt. Diese Angriffe müssen die externe Firewall nicht mehr passieren und können daher weder abgewehrt oder protokolliert werden!

Hier helfen Intrusion-Detection- und -Prevention-Systeme. Während ein Intrusion-Detection-System (IDS) keine zusätzliche Sicherheit schafft, sondern nur nach einem Angriff diesen meldet, wehrt ein Intrusion-Prevention-System (IPS) diesen Angriff direkt bei der Erkennung ab. Im Folgenden werde ich Ihnen eine kurze Einführung in die verschiedenen Technologien geben. Weitere Informationen finden Sie in meinem Buch „Intrusion Detection und Prevention mit Snort & Co.“, das ebenfalls im Addison-Wesley Verlag erschienen ist.

6.12.1 Intrusion-Detection-Systeme

Die Intrusion Detection versucht, Einbrüche und Missbrauch zu erkennen und zu melden. Hierzu versuchen die verschiedenen Systeme, sowohl das Netzwerk als auch die Rechner auf Anzeichen eines Angriffs, Einbruchs oder Missbrauchs zu analysieren und im Zweifel einen Alarm auszulösen. Achtung: Die Intrusion Detection verhindert nicht den Einbruch. Sie ist mit einem Feuermelder in einem Haus vergleichbar. Wenn keine Reaktion auf den Feueralarm erfolgt, wird das Haus trotz des Alarms niederbrennen. Für ein erfolgreiches Intrusion-Detection-Konzept ist es erforderlich, dass die Meldungen analysiert und anschließend Reaktionen eingeleitet werden. Die Intrusion-Detection-Systeme werden allgemein in zwei Gruppen eingeteilt:

» netzwerkbasierte IDS (NIDS)

» hostbasierte IDS (HIDS)

Die NIDS beziehen ihre Daten aus dem Netzwerk. Sie analysieren jedes Paket und jede Netzwerkverbindung auf ihre Gültigkeit, auf Angriffsmuster und auf das Vorhandensein von Signaturen bekannter Angriffe. NIDS kämpfen heute vor allem mit zwei Problemen:

8 Dieses Programm ist aufgrund von Lizenzproblemen nicht in der Debian-Distribution enthalten.

1. Die LANs werden immer schneller und transportieren immer mehr Daten. Für die Realisierung dieser LANs wird daher meist eine Paket-Switching-Technologie eingesetzt, die es nicht mehr jedem Rechner im Netzwerk erlaubt, den gesamten Verkehr zu beobachten. Daher müssen NIDS über spezielle Network-Taps oder Spanning-Ports angeschlossen werden. Gleichzeitig sollen die NIDS aber mehrere Systeme, meist ganze Netze, überwachen. Das bedeutet, dass die NIDS heute mit 1-Gbit/s- oder 10-Gbit/s-Verkehr umgehen müssen. Die Anforderungen an die Hardware sind enorm. Meist kann übliche Hardware diese Anforderungen nicht erfüllen, und spezielle prozessorunterstützte Netzwerkkarten sind erforderlich.
2. Die wesentlichen Funktionen des NIDS werden immer noch signaturbasiert erreicht. Auch wenn die Signaturen der modernen IDS wie Snort wesentlich besser geworden sind, besteht dennoch die Problematik, dass meist nur für bekannte Angriffe Signaturen existieren. Eine komplette Überprüfung sämtlicher Applikationsprotokolle ist häufig zu aufwendig.

Bei den HIDS gibt es wesentlich mehr Variabilität. Es existieren viele verschiedene Technologien, die ihre Daten von einem Host beziehen und auf unterschiedlichste Weise versuchen, einen Einbruch zu erkennen. Die bekanntesten sind Werkzeuge wie Tripwire, Aide oder Samhain. Hierbei handelt es sich um File Integrity Verifier (auch als System Integrity Assessment bezeichnet). Diese analysieren ein System und melden Änderungen an den überwachten Dateien. Sie müssen dann entscheiden, ob die Modifikation der Datei erlaubt war oder ob dies auf eine nicht autorisierte Handlung hinweist. Nachdem Tripwire lange Zeit die am häufigsten eingesetzte Applikation war, können Samhain und OSSEC seit einigen Jahren eine steigende Zahl an Benutzern vorweisen. Sehr intelligente Funktionen machen es möglich, mehrere Systeme gleichzeitig zentral zu überwachen. Dies ist mit der Open-Source-Variante von Tripwire nicht möglich.

Andere HIDS überwachen die Anzahl der Prozesse, die angemeldeten Benutzer oder den Speicherverbrauch und melden hier ungewöhnliche Zustände. Systeme wie Logwatch oder Logsurfer analysieren die Protokolldateien und erkennen unbekannte Meldungen, die sie anschließend melden. Der Administrator muss nun wieder entscheiden, ob die Meldung harmlos ist oder auf einen Einbruch hinweist.

Bei allem Hype um Intrusion-Detection-Systeme sollten Sie nie vergessen, dass ein Intrusion-Detection-System nie gute Systemadministration, gutes Patchmanagement und eine gute Firewall ersetzen kann. Außerdem vereinfacht ein IDS nicht die weitere Administration, sondern erhöht den Aufwand, da die Meldungen des IDS analysiert und ausgewertet werden müssen. Denken Sie an den Feuermelder, den niemand hört!

6.12.2 Intrusion-Prevention-Systeme

Im Juni 2003 erklärte die Beratungsfirma Gartner, dass IDS bis zum Jahr 2005 überflüssig werden würden. Dies führte zu viel Verwirrung bei den führenden Anbietern von IDS und ihren potenziellen Kunden. Gartners Schlüsselaussage war, dass Firmen in Zukunft mehr Geld in

ihre Firewalls investieren würden, um Angriffe abzuwehren, statt in IDS, um die erfolgreichen Einbrüche zu melden.

Diese Aussage weist eine gewisse Logik auf. Jedoch existierten kaum Firewalls, die die Angriffe abwehren können, die ein Intrusion-Detection-System erkennen kann. Die Hersteller von IDS sahen hier ein neues Geschäftsfeld. Die Intrusion Prevention war geboren.

Ein Network-Intrusion-Prevention-System (NIPS) ist die aktive und intelligente Kombination aus Firewall und IDS. Dabei wird der Verkehr von der Firewall gefiltert, und bestimmte Protokolle werden zusätzlich von dem IDS analysiert. Sobald das IDS einen Angriff erkennt, verbietet die Firewall die Weiterleitung des Netzwerkpakets. Jeder namhafte Hersteller hat seit einigen Jahren NIPS-Produkte in seinem Portfolio.

Genauso haben die Anbieter von HIDS die Entwicklung von Host-Intrusion-Prevention-Systemen begonnen. Diese erkennen zum Beispiel den Versuch einer Dateimodifikation und verhindern diesen direkt. Zusätzlich kann der betroffene Benutzer gleichzeitig abgemeldet werden, können erneute Anmeldungen unterbunden werden oder ähnliche Maßnahmen von dem HIPS eingeleitet werden. Ein HIPS kann Prozesse beenden oder dafür sorgen, dass ein Prozess immer läuft. Sobald ein bestimmter Prozess versucht, andere unautorisierte Prozesse zu starten (zum Beispiel eine Shell), kann der Prozessaufruf unterbunden werden oder sogar der Elternprozess beendet werden.

Die größte Schwäche dieser IPS derzeit sind falsch-positive Angriffserkennungen. Während dies bei einem IDS ärgerlich ist und mit Fine-Tuning recht gut in den Griff zu bekommen ist, ist dies bei einem IPS unverzeihlich. Die Netzwerkverbindung oder der Zugriff auf eine Datei werden effektiv unterbunden. Falls gerade wichtige, nicht wiederherstellbare Daten transportiert oder gespeichert werden sollen, kann ein IPS hohe Verluste erzeugen.

Wenn die Intrusion-Prevention-Systeme in Zukunft zielsicherer werden und die Gefahr falschpositiver Erkennung zurückgeht, werden sie eine sinnvolle Ergänzung einer Firewall darstellen.

Teil III

Typische Firewall-Konfigurationen

7. Eine lokale Firewall

Im letzten Kapitel haben wir uns mit einer typischen Firewall beschäftigt. Dabei handelte es sich um einen Router, der sämtliche Pakete gefiltert hat, die durch ihn weitergeleitet wurden. Viele Anwender vergessen jedoch, dass die kompletten Firewall-Funktionalitäten auch auf jedem anderen Linux-System vorhanden sind. Sie können die Firewall-Funktionen also auch lokal nutzen, um Ihr System vor den Zugriffen anderer Benutzer im lokalen Netz zu schützen. Das ist insbesondere dann interessant, wenn es sich bei dem Rechner um einen Standalone-Rechner handelt. Das kann zum Beispiel ein Root-Server bei einem Provider sein oder ein DNS-Server, der relativ ungeschützt in einer demilitarisierten Zone (DMZ) steht. Dieses Kapitel zeigt Ihnen, wie Sie die Firewall einrichten, sodass Ihre lokalen Dienste geschützt werden.

7.1 Wieso eine lokale Firewall?

Wieso sollten Sie eine lokale Firewall einrichten, wenn Sie doch schon Ihr Netzwerk mit einer Firewall gesichert haben? Wie ich bereits in der Einführung erwähnt habe, können Sie bei einigen Rechnern (z.B. Root-Servern) gar keinen Einfluss auf die zentrale Firewall des Netzes nehmen. Auch Systeme in einer DMZ sind meist nicht ganz so gut geschützt. Zumindest untereinander werden sie häufig nicht noch einmal von einer Firewall getrennt. Auch wenn Sie über eine zentrale Firewall verfügen, möchten Sie vielleicht spezielle Rechner (z.B. Datenbankserver) zusätzlich durch eine Firewall vor den anderen Rechnern schützen. Im Zeitalter der Viren und Trojaner kann dies die Rettung für derartige Systeme darstellen.

Im Windows-Desktop-Bereich haben sich lokale Firewalls seit einiger Zeit etabliert. Obwohl man unterschiedlicher Meinung über diese Firewall-Systeme sein kann (siehe den Exkurs zu Windows-Desktop-Firewalls), haben diese Systeme einige nicht zu unterschätzende Vorteile.

EXKURS

Windows-Desktop-Firewalls. *Die ersten Windows-Desktop-Firewalls erschienen für Windows 9x und ME. Während diese Systeme eine scheinbare Sicherheit vorgaukelten, konnten sie diese in Wirklichkeit nicht bieten. Da diese Windows-Betriebssysteme nicht das Benutzer/Rechte-Konzept kennen, gibt es unter ihnen keine Möglichkeit, den Firewall-Prozess besonders zu schützen. Sobald der Benutzer ein Programm aufruft, kann dieses Programm mit den Rechten des Benutzers die Firewall beenden, entfernen und umkonfigurieren. Einen Schutz können diese Programme also erst bieten, sobald der normale Benutzer keinerlei Zugriff mehr auf die Administration der Firewall hat. Dies ist erst ab Windows NT möglich. Hier kann eine Firewall mit Administratorrechten installiert und konfiguriert werden. Solange anschließend der Benutzer nicht mit*

Administratorrechten arbeitet, kann so verhindert werden, dass der Benutzer oder ein von ihm aufgerufenes Programm die Firewall abschaltet.

Dennoch stehe ich weiterhin derartigen Systemen auf dem Windows-Desktop kritisch gegenüber. Sie kennen das Phänomen vielleicht auch: Sie haben einen Bekannten, der einen Windows-Desktop benutzt und seit einer Woche eine derartige Firewall installiert hat. Anschließend fragt er Sie bei jedem Treffen (denn Sie haben ja Ahnung von Rechnern), warum er denn so häufig angegriffen werde, ob die Angriffe gefährlich seien und dass er ja früher vollkommen ungeschützt gewesen sei. Möglicherweise ist der Bekannte aber auch so verängstigt, dass er den Rechner gar nicht mehr verwendet.

Viele Desktop-Firewall-Produkte erzeugen jedes Mal ein Popup-Fenster, sobald ein beliebiger Zugriff auf den Rechner erfolgt. Bei einem Portscan öffnen manche Firewalls schneller die Popup-Fenster, als der Anwender sie schließen kann. Während ein derartiger Portscan vollkommen ungefährlich ist, suggeriert die Firewall einen gefährlichen Angriff. Denken Sie daran: Wo kein Dienst angeboten wird, kann ein Angreifer auch in keinen Dienst einbrechen. Wir vernachlässigen hier einmal die Gefahr eines Einbruchs in den Kernel des Betriebssystems selbst. Eine sehr schöne Funktion einer derartigen Desktop-Firewall ist jedoch die Möglichkeit nachzuvollziehen, ob und welche Software gerade einen Zugriff auf das Internet durchführt. So können Sie mit einer Desktop-Firewall bei entsprechendem Wissen recht einfach Spyware und Trojaner erkennen.

Meiner Meinung nach ist der wichtigste Vorteil einer lokalen Firewall die Möglichkeit, zu entscheiden, welcher Prozess eine Kommunikation mit dem Netzwerk führen darf. Da die Firewall auf demselben System installiert ist, auf dem auch der Prozess läuft, kann Iptables diesen Zusammenhang nutzen, um Pakete zu erlauben. So können Sie zum Beispiel auf einem DNS-Server nur dem Prozess `named` die Erlaubnis geben, Pakete zu versenden.

7.2 Die Ketten

Während auf einer Firewall, die als Router fungiert, vor allem die FORWARD-Kette der Filtertabelle betrachtet werden muss, sind für eine lokale Firewall die Ketten INPUT und OUTPUT der Filtertabelle interessant (siehe Abbildung 7.1).

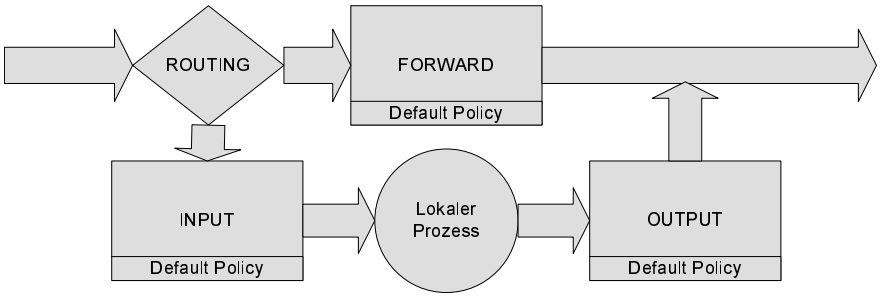


Abbildung 7.1: **Für die Filterung der lokalen Pakete sind die INPUT- und OUTPUT-Ketten zuständig.**

Bitte prüfen: Hier Satzpunkt ergänzt.

Editor's or typesetter's annotations (will be removed before the final TEX run)

Jedes Paket von außen wird in der INPUT-Kette gefiltert. Jedes Paket, das auf dem System erzeugt wird und den Rechner verlässt, wird in der OUTPUT-Kette gefiltert.

Wenn Sie möchten, dass ein System nur Verbindungen nach außen öffnen darf, dass aber keine Verbindungen von außen aufgebaut werden dürfen, können Sie folgende Regeln verwenden:

Listing 7.1: **Diese lokale Firewall erlaubt alle ausgehenden Verbindungen.**

```
# (1) Verwirf alle nicht explizit akzeptierten Pakete
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP

# (2) Lösche alle Regeln in den INPUT- und OUTPUT-Ketten
$IPTABLES -F INPUT
$IPTABLES -F OUTPUT

# (3) Erlaube neue Verbindungsaufbauten nach außen
$IPTABLES -A OUTPUT -m state --state NEW -j ACCEPT

# (4) Erlaube Antwortpakete und Fehlermeldungen für aufgebaute ↵
        Verbindungen
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# (5) Erlaube aufgebaute Verbindungen nach außen
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Die Regeln haben die folgende Aufgabe:

1. Zunächst werden die Grundregeln (Policies) der Ketten INPUT und OUTPUT auf DROP gesetzt. Damit wird jedes Paket verworfen, das nicht explizit durch eine Regel akzeptiert wird.
2. Dies löscht alle vorhandenen Regeln in den Ketten INPUT und OUTPUT. Damit ist sichergestellt, dass die anschließend eingefügten Regeln die einzigen Regeln in diesen Ketten sind.
3. Diese Regel erlaubt Pakete in der OUTPUT-Kette, wenn sie eine neue Verbindung aufbauen. Das bedeutet, dass eine derartige Verbindung noch nicht in der Zustandstabelle enthalten sein darf. Damit wird also das erste Paket einer jeden Verbindung erlaubt.
4. Mit dieser Regel werden die Antwortpakete (`ESTABLISHED`) anderer Rechner akzeptiert, wenn in der Verbindungstabelle (Connection Tracking-Table, `nf_conntrack`) eine passende Verbindung gefunden wird. Außerdem akzeptiert diese Regel Fehlermeldungen, wenn sie eine der Verbindungen in der Verbindungstabelle betreffen (`RELATED`).
5. Diese letzte Regel wird gern vergessen. Jedoch funktioniert ohne sie nichts. Diese Regel akzeptiert alle weiteren lokal erzeugten Pakete, die zu einer aufgebauten Verbindung gehören. Die Regel Drei erlaubt nur das eine Paket zum Verbindungsaufbau. Alle wei-

teren Pakete werden von der Regel Drei nicht mehr akzeptiert, da nun die Verbindung bekannt ist und die Pakete nicht mehr den Status NEW aufweisen. Alle weiteren lokal erzeugten Pakete einer Verbindung besitzen den Status ESTABLISHED. Mit den RELATED-Paketen werden auch Fehlermeldungen akzeptiert, die sich auf diese Verbindungen beziehen. Wenn diese Regel fehlt, erfolgt ein Verbindungsaufbau, der angesprochene Rechner antwortet, und die Antwort erreicht auch den lokalen Prozess, aber jedes weitere Paket des lokalen Prozesses wird verworfen.

Natürlich können Sie auch den Zugriff auf bestimmte Dienste einschränken. Dabei sollten Sie aber nicht vergessen, dass Ihr System für seine Funktion bestimmte Dienste grundsätzlich benötigen kann. Hierbei handelt es sich zum Beispiel um DHCP, DNS oder NTP. Wenn Ihre Firewall diese Dienste dann nicht erlaubt, ist ein einwandfreies Arbeiten des Systems nicht möglich. Die Funktion dieser Dienste und Ihre Filterung mit Iptables wird in einem eigenen Kapitel (siehe Kapitel 32) behandelt. Hier soll nur exemplarisch der Aufbau eines derartigen Regelwerks besprochen werden.

Stellen Sie sich vor, Sie möchten auf Ihrem Client lediglich zwei Dienste nutzen: DNS für die Namensauflösung und HTTP zum Surfen im Internet. Ihre lokale Firewall soll daher nur den Zugriff auf diese beiden Dienste erlauben. Sie könnten hierfür folgende Regeln nutzen:

Listing 7.2: **Diese lokale Firewall erlaubt nur den Zugriff auf DNS und HTTP.**

```
# (1) Verwirf alle nicht explizit akzeptierten Pakete
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP

# (2) Lösche alle Regeln in den INPUT- und OUTPUT-Ketten
$IPTABLES -F INPUT
$IPTABLES -F OUTPUT

# (3a) Erlaube neue DNS-Anfragen
$IPTABLES -A OUTPUT -p udp --dport 53 -m state --state NEW -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 53 -m state --state NEW -j ACCEPT

# (3b) Erlaube neue HTTP-Verbindungen
$IPTABLES -A OUTPUT -p tcp --dport 80 -m state --state NEW -j ACCEPT

# (4) Erlaube Antwortpakete und Fehlermeldungen für aufgebaute ↵
        Verbindungen
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# (5) Erlaube aufgebaute Verbindungen nach außen
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Zunächst ist der Aufbau der Regeln wieder identisch mit dem ersten Ansatz. Die Regeln unterscheiden sich nur in dem Punkt 3. Während wir in dem ersten Ansatz (siehe Listing 7.1) jede

neue ausgehende Verbindung erlaubt haben, werden hier die möglichen Verbindungen stark eingeschränkt. Zunächst erlaubt der Punkt 3a die DNS-Anfrage. Hier müssen Sie die DNS-Anfrage sowohl mit dem UDP- als auch mit dem TCP-Protokoll erlauben. DNS ist ein wenig ungewöhnlich. Es stellt zunächst immer die Anfrage in UDP. Können jedoch mit UDP nicht alle Daten übertragen werden, stellt der Client die Anfrage erneut mit TCP. Weitere Informationen finden Sie in Abschnitt 32.2. Der Punkt 3b erlaubt dann neue HTTP-Verbindungen. Da die Pakete mit dem Zustand NEW in der OUTPUT-Kette akzeptiert werden, ist nicht der Aufbau neuer Verbindungen von außen möglich. Der Rest der Regeln gleicht wieder dem ersten Ansatz.

STOP

Wenn Sie derart eine lokale Firewall aufsetzen, sollten Sie nie vergessen, dass auch lokale Prozesse untereinander über ein lokales Netz kommunizieren. Sobald die lokalen Prozesse Internet-Sockets[1] *und nicht UNIX-Sockets nutzen, werden hierbei auch Pakete erzeugt, die in der* `OUTPUT`- *und* `INPUT`*-Kette gefiltert werden. Wenn Sie nicht spezielle Regeln vorsehen, die diese Pakete akzeptieren, unterbinden Sie die lokale Kommunikation. Viele Dienste funktionieren dann nicht mehr!*

Dieser Regelsatz genügt jedoch meist nicht. Häufig existieren lokal genutzte Dienste wie eine grafische Oberfläche, die ebenfalls Netzwerkfunktionen nutzt. Damit diese Dienste auch weiterhin funktionieren, müssen Sie sicherstellen, dass deren Netzwerkpakete von Ihrer Firewall nicht verworfen werden. Die Firewall akzeptiert im Moment aber nur DNS- und HTTP-Verbindungen. Am einfachsten fügen Sie zu Beginn folgende Regeln grundsätzlich Ihren Firewall-Skripten hinzu:

```
# Akzeptiere den lokalen Verkehr
$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A OUTPUT -o lo -j ACCEPT
```

Diese Regeln akzeptieren den gesamten Verkehr, der über das Loopback-Device `lo` abgewickelt wird. Dies ist ein virtuelles Interface, das den gesamten lokalen Verkehr transportiert. Von außen ist kein Zugriff auf dieses Interface möglich. Daher stellt eine Akzeptanz des gesamten Verkehrs auf diesem Interface normalerweise auch kein Sicherheitsproblem dar.

Wenn auf Ihrem System mehrere Benutzer arbeiten, deren Netzwerkverkehr auch untereinander überwacht werden soll, dürfen Sie natürlich nicht den Verkehr blind akzeptieren. Dieses Szenario ist aber recht selten. Wenn Sie in Ihrer Firewall dieses Szenario implementieren wollen, lesen Sie bitte noch den nächsten Abschnitt über den Owner-Match. Dieser wird Ihnen die Arbeit erheblich vereinfachen.

Sie sollten nun bereits die Mächtigkeit der lokalen Firewall erkannt haben. Noch wichtiger wird die lokale Firewall, wenn Sie auf einem ansonsten ungeschützten System die Dienste schützen möchten. Auch hier können Sie eine lokale Firewall nutzen. Nur werden hier die

1 Ein UNIX-Socket ist eine lokale Datei im Verzeichnisbaum, die für die Kommunikation genutzt wird. Ein Beispiel ist `/dev/log`. Netzwerkverbindungen nutzen Internet-Sockets. Diese können auch von lokalen Prozessen für die lokale Kommunikation genutzt werden. Wenn Sie den Befehl `telnet localhost` aufrufen, kommunizieren Sie über einen derartigen Internet-Socket.

neuen Verbindungen von außen aufgebaut. Sie können aber entscheiden, mit welchem Protokoll auf welchen Port Sie die Verbindung erlauben. Wenn Sie zum Beispiel einen Root-Server betreiben und sich auf diesem System sowohl ein Webserver als auch ein SSH-Server befinden, können Sie die folgenden Regeln nutzen:

Listing 7.3: **Die lokale Firewall schützt einen Root-Server.**

```
# (1) Verwirf alle nicht explizit akzeptierten Pakete
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP

# (2) Lösche alle Regeln in den INPUT- und OUTPUT-Ketten
$IPTABLES -F INPUT
$IPTABLES -F OUTPUT

# (3) Akzeptiere den lokalen Verkehr
$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A OUTPUT -o lo -j ACCEPT

# (4a) Erlaube neue SSH-Verbindungen von außen
$IPTABLES -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT

# (4b) Erlaube neue HTTP-Verbindungen von außen
$IPTABLES -A INPUT -p tcp --dport 80 -m state --state NEW -j ACCEPT

# (5) Erlaube Antwortpakete und Fehlermeldungen für aufgebaute ↩
        Verbindungen
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# (6) Erlaube aufgebaute Verbindungen nach außen
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Den Großteil der Regeln kennen Sie bereits. Die meisten Regeln wurden nicht verändert. Hier ist noch einmal ihre Funktion in aller Kürze:

1. Hier werden die Grundregeln für die Ketten `INPUT` und `OUTPUT` auf `DROP` gesetzt.
2. Hier werden alle möglicherweise vorhandenen Regeln in den Ketten INPUT und OUTPUT gelöscht.
3. Diese Regeln akzeptieren den gesamten lokalen Verkehr auf dem Loopback-Interface.
4. Die Punkte 4a und 4b erlauben schließlich neue SSH- und HTTP-Verbindungen von außen.
5. Diese Regel erlaubt alle Pakete ab dem zweiten Paket in einer aufgebauten Verbindung von außen.
6. Diese Regel schließlich erlaubt es, dass der lokale Rechner auf die von außen aufgebauten Verbindungen reagieren darf.

Mit einem derartigen Firewall-Skript können Sie sicherstellen, dass von außen nur die von Ihnen gewünschten Dienste auf dem System genutzt werden dürfen, unabhängig von den installierten und gestarteten Diensten. Die lokale Nutzung der installierten Dienste ist davon aber nicht betroffen.

Wenn Sie zusätzlich den Zugriff einschränken möchten und immer von einer bestimmten IP-Adresse auf den SSH-Server zugreifen, können Sie dies durch eine einfache Änderung der Regel 4a erreichen:

```
REMOTE_ACCESS=3.0.0.1
# (4a) Erlaube neue SSH-Verbindungen von außen
$IPTABLES -A INPUT -p tcp -s $REMOTE_ACCESS --dport 22 -m state --state ↵
        NEW -j ACCEPT
```

Mit der Option `-s` definieren Sie, dass nur neue SSH-Verbindungen von dieser Source-IP-Adresse akzeptiert werden!

7.3 Der Owner-Erweiterung

Sie haben jetzt bereits sehr mächtige lokale Firewalls aufgebaut, die nur den Zugriff auf bestimmte Dienste zugelassen haben. Sie können diese weiter verbessern, indem Sie den `owner`-Match einsetzen. Hierbei handelt es sich um eine Iptables-Option, die Sie in der OUTPUT-Kette nutzen können. Diese Option erlaubt es festzustellen, welcher Benutzer oder welche Gruppe ein Paket erzeugt hat. Dies ist jedoch nur in der OUTPUT-Kette möglich, da Iptables diese Funktion natürlich nur für lokal erzeugte Pakete anbieten kann. Zusätzlich kann diese Option auch die Prozessnummer oder die Sitzungsnummer zur Auswahl verwenden. Neue Versionen von Iptables können auch den Kommandonamen verwenden!

Die genaue Syntax des Owner-Match können Sie in Abschnitt 7.3 nachlesen. Hier werden Sie einige Beispielregeln sehen.

Ein einfaches erstes Beispiel wird Ihnen die Möglichkeiten des Owner-Matches verdeutlichen. Stellen Sie sich vor, Sie betreiben einen Terminalserver, auf dem mehrere Benutzer gleichzeitig arbeiten dürfen. Einige Benutzer haben das Recht, auf das Internet zuzugreifen, während anderen Benutzern dieser Zugriff verwehrt werden soll. Mit den normalen Mitteln können Sie dies nicht implementieren, da alle Benutzer dieselbe Source-IP-Adresse nutzen. Der Owner-Match macht es nun doch möglich. Sie können neue Verbindungen eines Benutzers zulassen und einen anderen Benutzer ablehnen:

```
# Benutzer Ralf hat die UID 500
$IPTABLES -A OUTPUT -m owner --uid-owner 500 -m state --state NEW -j ↵
        ACCEPT

# Benutzer Thorsten hat die UID 501
$IPTABLES -A OUTPUT -m owner --uid-owner 501 -m state --state NEW -j ↵
        REJECT
```

Wenn Sie darauf achten, dass Sie ansonsten alle ESTABLISHED- und RELATED-Pakete in der INPUT- und OUTPUT-Kette akzeptieren, wird nun der Benutzer Ralf auf das Internet zugreifen dürfen, während der Benutzer Thorsten das nicht darf. Dabei können Sie dem Benutzer Ralf auch nur bestimmte Befehle erlauben. Wenn der Benutzer Ralf nur Verbindungen zum Port 22 aufbauen darf, um SSH-Sitzungen zu nutzen, können Sie folgende Zeile verwenden:

```
# Benutzer Ralf hat die UID 500 und darf ssh benutzen
$IPTABLES -A OUTPUT -m owner --uid-owner 500 -p tcp --dport 22 -m state ↵
        --state NEW -j ACCEPT
```

Diese Funktion können Sie auch verwenden, um einen Server zusätzlich zu sichern. Wenn Sie zum Beispiel einen kombinierten Mail- und Webserver betreiben, so können Sie sicherstellen, dass lediglich der Mailserver Verbindungen nach außen aufbauen darf und der Webserver nur Verbindungen entgegennimmt. Sollte es einem Angreifer gelingen, in den Webserver-Prozess einzubrechen, so verfügt er nicht über die Möglichkeit, zusätzliche Angriffswerkzeuge aus dem Internet nachzuladen. Hierzu müssen diese Dienste jedoch unterschiedliche Linux-Benutzer verwenden.

Listing 7.4: **Nur der E-Mailserver darf eine Verbindung nach außen aufbauen. Dem Webserver ist dies untersagt.**

```
# Der Webserver hat die UID 48 (apache)
# Der E-Mailserver hat die UID 89 (postfix)

# Der E-Mailserver baut (mindestens) DNS- und SMTP-Verbindungen auf
$IPTABLES -A OUTPUT -m owner --uid-owner 89 -p udp --dport 53 -m state -- ↵
        state NEW -j ACCEPT
$IPTABLES -A OUTPUT -m owner --uid-owner 89 -p tcp -m multiport --dport ↵
        25,53 -m state --state NEW -j ACCEPT

# Der Webserver darf keine Verbindung aufbauen
$IPTABLES -A OUTPUT -m owner --uid-owner 48 -m state --state NEW -j ↵
        REJECT
```

Die in diesem Regelsatz verwendete Option `multiport` erlaubt die gleichzeitige Angabe von bis zu 15 UDP- oder TCP-Ports. So können mehrere Regeln zu einer zusammengefasst werden und kann die Lesbarkeit erhöht werden. Weitere Hinweise zu `multiport` finden Sie in Abschnitt 16.9.11.

Zum Abschluss möchte ich Ihnen noch ein komplettes Beispielskript für einen Root-Server vorschlagen. Sie können es entsprechend Ihren Wünschen natürlich anpassen:

Listing 7.5: **Dieses Firewall-Skript schützt einen typischen Root-Server.**

```
#!/bin/bash
#
# Firewall-Skript für einen Root-Server
# (c) Ralf Spenneberg 15. Juni 2005
# Dieser Root-Server bietet die folgenden Dienste:
# HTTP Port 80/tcp
# HTTPS Port 443/tcp
# SMTP Port 25/tcp
# IMAPS Port 995/tcp
# DNS Port 53/tcp und 53/udp
#
# Dieser Root-Server benötigt Zugriff auf:
# DNS Port 53/tcp und 53/udp
# NTP Port 123/udp (Zeitsynchronisation)
# SMTP Port 25/tcp (zum Verschicken von E-Mail)
#
# Jeder lokale Dienst nutzt den lokalen DNS-Server zur Namensauflösung
# Keiner der Dienste versucht eigenständig, einen anderen DNS-Server zu ↵
        erreichen

# Definiere Variablen
IPTABLES=/sbin/iptables
ECHO=/bin/echo
SYSCTL=/sbin/sysctl

$ECHO "Starte Firewall"

$ECHO "Setze Kernelparameter"
$SYSCTL -w net.ipv4.tcp_syncookies=1
$SYSCTL -w net.ipv4.icmp_echo_ignore_broadcasts=1

$ECHO "Setze Regeln"
# Setze die Default-Policy auf DROP
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Akzeptiere Verkehr auf der Loopback-Schnittstelle
$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A OUTPUT -o lo -j ACCEPT
```

```
# Akzeptiere eingehende Verbindungen für SMTP, DNS, HTTP, HTTPS und IMAPS
$IPTABLES -A INPUT -p tcp -m multiport --dport 25,53,80,443,995 -m state ↵
        --state NEW -j ACCEPT
$IPTABLES -A INPUT -p udp --dport 53 -m state --state NEW -j ACCEPT

# Ausgehende Verbindungen

# Der DNS-Server baut Verbindungen als Benutzer named auf
UID=$(id -u named)
$IPTABLES -A OUTPUT -p udp --dport 53 --m owner --uid-owner $UID -m state ↵
         --state NEW -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 53 --m owner --uid-owner $UID -m state ↵
         --state NEW -j ACCEPT

# Der NTP-Server baut Verbindungen als Benutzer ntp auf
UID=$(id -u ntp)
$IPTABLES -A OUTPUT -p udp --dport 123 --m owner --uid-owner $UID -m ↵
        state --state NEW -j ACCEPT

# Der SMTP-Server baut Verbindungen als Benutzer postfix auf
UID=$(id -u postfix)
$IPTABLES -A OUTPUT -p tcp --dport 25 --m owner --uid-owner $UID -m state ↵
         --state NEW -j ACCEPT

# Alle bereits aufgebauten Verbindungen sollen erlaubt werden
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

$ECHO "Firewall erfolgreich gestartet"
```

Wenn Sie das Skript auf mehreren Rechnern einsetzen wollen, kann es sein, dass der Benutzer `postfix` auf diesen Systemen eine unterschiedliche UID aufweist. Der Befehl `id -u postfix` ermittelt die aktuelle UID. So können Sie das Skript leichter auf weitere Systeme übertragen.

Dieses Skript sollte Ihnen genug Material für erste Experimente mit einer lokalen Firewall geben. Weitere Anregungen und Hinweise erhalten Sie in den fortgeschrittenen Kapiteln.

7.4 Kombination mit Gateway-Regeln

Sie werden nur in wenigen Umgebungen den idealen Fall antreffen, dass eine Firewall keinerlei eigene Dienste anbieten wird. In vielen Fällen wird mindestens noch ein SSH-Daemon für die Administration auf dem System installiert sein. In einigen Konstellationen wird sich auf der Firewall auch ein VPN-Produkt befinden, das einen VPN-Zugang in das interne Netz

ermöglicht. Daher müssen Sie häufig auf einer klassischen Firewall, die als Gateway zwei Netze verbindet und den Verkehr kontrolliert, auch zusätzliche Regeln für die Filterung des lokalen Verkehrs vorsehen. Diese Regeln unterscheiden sich kaum von den bisher betrachteten Regeln, jedoch sollten Sie zusätzlich die Tatsache berücksichtigen, dass die Firewall über mehrere Netzwerkkarten verfügt, die Sie bei dem Aufsetzen Ihrer Regeln berücksichtigen können und sollten.

Beginnen wir mit dem einfachen Fall, dass Sie auf Ihrer Firewall, die bereits mit einem kompletten Regelsatz ausgestattet ist (siehe Abschnitt 1.3), nun auch noch einen SSH-Server installieren und auf diesen von innen zugreifen möchten. Dies ist sehr einfach durch das Hinzufügen der folgenden Regeln an das Ende des vorhandenen Skripts möglich:

Listing 7.6: **Ein Zugriff auf den SSH-Server soll von innen erlaubt werden.**

```
# Erlaube SSH-Zugriff von innen
$IPTABLES -A INPUT -i $INTDEV -p tcp --dport 22 -m state --state NEW -j ↵
        ACCEPT

# Erlaube alle bereits aufgebauten Verbindungen in der INPUT- und OUTPUT- ↵
        Kette
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Diese Regeln nutzen die Variable `$INTDEV`, die Sie hoffentlich bereits in Ihrem Firewall-Skript definiert haben. Der Wert dieser Variablen entspricht der internen Netzwerkkarte. So können Sie sehr einfach nur SSH-Verbindungen von innen zulassen. Wenn Sie den Zugriff weiter einschränken möchten und zum Beispiel nur bestimmte Administrationsrechner von innen zugreifen lassen möchten, können Sie die folgenden Regeln verwenden:

```
# Adminrechner, Liste durch Leerzeichen getrennt
ADMIN="192.168.0.5 192.168.0.8 192.168.0.17"

# Erlaube den Adminrechnern SSH-Zugriff von innen
for PC in $ADMIN
do
  $IPTABLES -A INPUT -s $PC -i $INTDEV -p tcp --dport 22 -m state --state ↵
          NEW -j ACCEPT
done
```

INFO

For-Schleife.

Jede Shell bietet Ihnen Möglichkeiten der Prozesssteuerung und auch eine For-Schleife. Die For-Schleife der Bourne-Shell (und damit auch der Bash- und Korn-Shell) erlaubt dabei die Angabe einer Werteliste, die anschließend durchlaufen wird. Damit Sie einen Eindruck von der Funktion der For-Schleife erhalten, können Sie den folgenden Befehl auf der Kommandozeile aufrufen:

INFO

```
[spenneb@bibo ~]$ for i in eins zwei drei vier; do echo $i; done

eins
zwei
drei
vier
```

So können Sie sehr einfach einen Befehl wiederholt mit unterschiedlichen Werten aufrufen. In dem Iptables-Listing oben wird der Iptables-Befehl mehrfach mit unterschiedlichen Source-Adressen aufgerufen. So wird für jede Source-Adresse eine Regel hinzugefügt, die eine SSH-Verbindung erlaubt.

INFO

Anstelle einer For-Schleife können Sie auch IP-Sets (`ipset`) verwenden. Dies ist eine Weiterentwicklung des ehemaligen IP-Pools. Damit können Sie mehrere IP-Adressen in einer Regel verwenden. Leider unterstützen noch nicht alle Distributionen automatisch diese Funktion, und Sie müssen den Kernel und den Iptables-Befehl patchen. Weitere Informationen über diese Funktion finden Sie in Kapitel 26.

Möchten Sie entsprechend den SSH-Zugriff von außen erlauben, so ersetzen Sie in der Regel `-i $INTDEV` durch `-i $EXTDEV`.

Häufig wünschen sich die Administratoren und damit vielleicht auch Sie, Verbindungen von der Firewall in das Internet aufbauen zu können. Zumindest soll es möglich sein, von der Firewall zu Testzwecken mit dem Ping-Befehl die Konnektivität testen zu können. Das können Sie ganz einfach erreichen, indem Sie die folgende Zeile hinzufügen:

```
# Erlaube Ping in das Internet
$IPTABLES -A OUTPUT -o $EXTDEV -p icmp --icmp-type echo-request -m state ↵
        --state NEW -j ACCEPT
```

Möchten Sie, dass die Firewall auch angepingt (Echo-Request) werden kann, um die Erreichbarkeit der Firewall sowohl von innen als auch von außen testen zu können, benötigen Sie noch die folgende Zeile:

```
# Erlaube, dass die Firewall per Ping getestet werden kann
$IPTABLES -A INPUT -p icmp --icmp-type echo-request -m state --state NEW ↵
        -j ACCEPT
```

Alle diese Regeln benötigen natürlich für eine korrekte Funktion zusätzliche Regeln in der INPUT- und OUTPUT-Kette, die ESTABLISHED- und RELATED-Pakete akzeptieren.

Nun sollten Sie über das Rüstzeug verfügen, um ein Standalone-Linux-System mit einer Firewall zu schützen. Außerdem können Sie ein Firewall-Gateway so konfigurieren, dass Sie lokale Dienste auf der Firewall nutzen können und diese Firewall selbst auf weitere Dienste zugreifen darf.

8. Aufbau einer DMZ

Eine demilitarisierte Zone (DMZ) ist ein eigenes Netzwerk, in das bestimmte Rechner ausgelagert werden, die einen unbeschränkteren Zugriff auf das Internet benötigen oder vom Internet erreicht werden sollen (z.B. Webserver). Dieser Aufbau einer demilitarisierten Zone bietet viele Vorteile. Wenn ein Angreifer erfolgreich in einen dieser Rechner einbrechen sollte, so hat er noch keinen Zugriff auf das immer noch geschützte interne LAN. Befände sich der Webserver nicht in der DMZ, sondern direkt im internen LAN, hätte der Angreifer Zugriff auf alle Systeme!

Dieses Kapitel beschreibt den Aufbau einer DMZ, zeigt unterschiedliche Architekturen und beispielhaft den Aufbau der Firewall-Skripten, um diese zu implementieren.

8.1 DMZ-Architekturen

Bereits in Kapitel 3, „Firewall-Architekturen“ haben wir verschiedene Möglichkeiten angesprochen, eine DMZ zu implementieren. Dabei wurden auch Vor- und Nachteile der verschiedenen Varianten erörtert. Im Folgenden soll die Implementierung einer DMZ als drittes Bein eines Paketfilters (siehe Abbildung 3.2) und als Netz zwischen zwei Paketfiltern (siehe Abbildung 3.3) exemplarisch durchgespielt werden. Während die Implementierung als drittes Bein einen relativ geringen Hardwareaufwand bedeutet, benötigt die Variante mit zwei Paketfiltern mindestens ein System als Paketfilter mehr. Dies bedeutet Geld-, Platz- und Kühlaufwand.

In der DMZ der Beispielfirma *Nohup.info* soll jeweils ein Web-, ein E-Mail-, ein cachender DNS- und ein Proxy-Server betrieben werden. Diese Systeme erhalten private IP-Adressen aus dem Bereich 192.168.255.0/24. Dabei soll aus dem Internet ein Zugriff auf den Web- und den E-Mailserver möglich sein. Aus dem internen Netz ist ein Zugriff auf den Proxy- und den E-Mailserver notwendig. Der Zugriff auf den Webserver wird über den Proxy-Server realisiert. Ein direkter Zugriff auf das Internet ist nicht vorgesehen. Der Paketfilter erhält als externe IP-Adresse eine feste statische IP-Adresse 3.0.0.1. Alle Dienste werden im Internet unter dieser IP-Adresse angebunden. Das bedeutet, dass ein DNS-Server bei der Frage nach der Adresse `www.nohup.info` die IP-Adresse 3.0.0.1 zurückliefert.

8.2 Dreibeiniger Paketfilter mit DMZ

Die Variante des dreibeinigen Paketfilters zur Implementierung einer demilitarisierten Zone ist die einfachste Variante einer DMZ. Der Hardwareaufwand ist gering, denn Sie benötigen hierfür nur eine zusätzliche Netzwerkkarte in Ihrem Paketfilter. Auch der weitere technische

Aufwand in Bezug auf Platz und Kühlung ist eher zu vernachlässigen, da Sie ja sowieso bereits ein System als Paketfilter einsetzen.

Falls Sie bereits ein Firewall-Skript für einen Paketfilter besitzen und dieses um die Funktionen für eine DMZ erweitern wollen, ist auch dies recht einfach machbar. In diesem Kapitel werden wir jedoch ein neues Skript entwickeln. Sie können die hier vorgestellten Vorschläge aber auch in jedes andere Skript übernehmen.

Zunächst sollten Sie die Netzwerkkarten zuordnen und in Ihrem Skript Variablen definieren, denen Sie die entsprechenden Werte zuweisen. Zusätzlich zu der Variable INTDEV und EXTDEV werden wir noch eine Variable DMZDEV verwenden. Dieser Variablen weisen Sie zunächst den Namen der Netzwerkkarte zu, über die die DMZ angebunden wird.

Listing 8.1: **Eine dreibeinige Firewall mit DMZ-Anschluss**

```
#!/bin/bash
# Autor : Ralf Spenneberg
# Version: 0.1
# Datum : 17. Dez 2004
# Dieses Skript lädt die Regeln für die Firewall

INTDEV="eth1"
# Achtung: Beim Einsatz von User-Mode-Linux lautet diese Variable
# INTDEV="tap0"

EXTDEV="eth0"

DMZDEV="eth2"

IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
```

Nun sollten Sie für die verschiedenen Dienste und Systeme Variablen definieren. Das erleichtert später die Wartung. Vielleicht installieren Sie zunächst nur ein System in der DMZ, das sowohl einen Webserver, als auch einen E-Mail- und einen Proxy-Server beherbergt. Später möchten Sie vielleicht den E-Mailserver auf einem eigenen System betreiben. Wenn Sie zu Beginn darauf geachtet haben, für alle Systeme Variablen zu verwenden, müssen Sie diese anschließend nur entsprechend anpassen und Ihr Skript neu starten.

```
MAIL=192.168.255.2
WEB=192.168.255.3
DNS=192.168.255.4
PROXY=192.168.255.5
EXTERN=3.0.0.1
```

Auch wenn in unserem Skript für den DNS-Server eine eigene IP-Adresse vorgesehen wurde, ist es natürlich nicht erforderlich, diesen auf eigener dedizierter Hardware zu installieren.

Wenn Sie den DNS-Server auf derselben Hardware wie den Proxy-Server installieren wollen, verwenden Sie einfach für beide Variablen dieselbe IP-Adresse!

Nun sollten Sie, bevor Sie irgendwelche Regeln hinzufügen, einen sauberen Grundzustand erzeugen.

```
# Setze Default-Regeln
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Lösche die Filter-Tabelle
$IPTABLES -F

# Lösche die NAT-Tabelle
$IPTABLES -t nat -F
```

Beginnen wir nun mit den NAT-Regeln. Insgesamt müssen wir die folgenden NAT-Regeln konfigurieren:

1. Der Proxy muss auf das Internet zugreifen können.
2. Der cachende DNS-Server muss auf das Internet zugreifen können.
3. Der E-Mailserver muss auf das Internet zugreifen können.
4. Das Internet muss auf den Webserver zugreifen können.
5. Das Internet muss auf den E-Mailserver zugreifen können.

Diese Anforderungen können mit den folgenden vier Regeln umgesetzt werden:

```
# (1) Proxy-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $PROXY -j SNAT --to-source ↵
        $EXTERN
# (2) DNS-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $DNS -j SNAT --to-source ↵
        $EXTERN
# (3) Mailserver-Zugriff nach außen
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $MAIL -j SNAT --to-source ↵
        $EXTERN
# (4) Webserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN -m multiport ↵
        --dport 80,443 -j DNAT --to-destination $WEB
# (5) Mailserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN --dport 25 -j ↵
         DNAT --to-destination $MAIL
```

Die Regeln Eins bis Drei führen eine Network Address Translation (NAT) der Absenderadresse durch. Dabei wird die originale Absenderadresse des Proxy oder des Mailservers bei einem Verbindungsaufbau nach außen durch die Adresse der Firewall ersetzt. Die Regeln Vier und Fünf sorgen dafür, dass bei Verbindungen von außen, die auf den Ports 25, 80 und 443 aufgebaut werden, Pakete entsprechend auf den E-Mail- und Webserver in die DMZ weitergeleitet werden. Dies erfolgt durch einen Austausch der Ziel-IP-Adresse in der NAT-PREROUTING-Kette. Die Portnummer wird dabei nicht modifiziert. Da aber die Weiterleitung (Port-Forwarding) in Abhängigkeit von der Portnummer erfolgen soll, muss zusätzlich auch das Protokoll TCP angegeben werden.

Wenn Sie aus bestimmten Gründen den Webserver in der DMZ auf einem anderen Port betreiben möchten, so wäre es auch möglich, die Portnummer zu modifizieren:

```
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN --dport 80 -j ↵
         DNAT --to-destination $WEB:8080
```

Die NAT-Regeln sind jedoch lediglich für die Adressumsetzung bei dem Zugriff aus dem Internet und in das Internet verantwortlich. Weitere Firewall-Regeln sind für die Funktion der Firewall erforderlich. Diese Regeln müssen nun auch den gewünschten Verkehr erlauben.

Wir benötigen zunächst eine Regel, die sämtliche aufgebauten Verbindungen und deren Fehlermeldungen akzeptiert:

```
# Akzeptiere alle aufgebauten Verbindungen
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Denken Sie daran: Die Zustandsüberwachung der Iptables-Firewall macht es möglich, dass Sie immer nur die Verbindungsaufbauten filtern und anschließend alle derart aufgebauten Verbindungen über eine einzige Regel akzeptieren. Nur Verbindungen, die Sie vorher explizit erlaubt haben (im Status NEW), werden von dieser Regel zugelassen. Der Aufbau der Regeln wird so wesentlich einfacher und übersichtlicher.

Nun benötigen wir eine Regel, um von innen auf den Proxy und den Mailserver zuzugreifen. Der Port eines Proxys ist nicht so festgelegt wie zum Beispiel der Port bei dem Zugriff auf einen Web- oder E-Mailserver. Hier werden vollkommen unterschiedliche Portnummern verwendet. Der Squid-Proxy-Server verwendet zum Beispiel die Portnummer 3128/tcp als Standardport. Es ist sinnvoll, auch hierfür eine Variable zu Beginn des Skripts zu definieren: PROXYPORT=3128. Dann können Sie diese Variable in Ihren Regeln nutzen und müssen bei einer Änderung nur zu Beginn des Skripts Änderungen vornehmen.

```
# Erlaube den Zugriff auf den Proxy
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $PROXY -p tcp --dport  ↵
        $PROXYPORT -m state --state NEW -j ACCEPT

# Erlaube den Zugriff auf den E-Mailserver
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m  ↵
        state --state NEW -j ACCEPT
```

Wenn der E-Mailserver in der DMZ auch Dienste zum Abholen der E-Mail bereitstellt, müssen Sie die letzte Regel so erweitern, dass auch diese Funktion unterstützt wird. Das Post Office Protocol (POP3) verwendet den Port 110, und das Internet Message Access Protocol (IMAP) verwendet den Port 143. Sinnvoller ist es aber, diese Dienste auf einem eigenen E-Mailserver in dem internen Netz anzubieten und den E-Mailserver in der DMZ nur als Relay zu nutzen. Dann können Sie natürlich auch den Zugriff auf den E-Mailserver in der DMZ auf den internen E-Mailserver beschränken, denn alle internen Anwender benutzen für ihren E-Mail-Zugriff nur den internen E-Mailserver.

```
MAILINTERN=192.168.0.100
# Erlaube den Zugriff auf den E-Mailserver
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -s $MAILINTERN -d $MAIL -p tcp ↵
         --dport 25 -m state --state NEW -j ACCEPT
```

Sobald eine E-Mail in das Internet gesendet werden muss, sendet der interne E-Mailserver diese E-Mail an den E-Mailserver in der DMZ, der sie in das Internet weiterleitet. Eingehende E-Mails nehmen den entgegengesetzten Weg. Für diesen Rückweg ist dann natürlich auch noch eine weitere Regel erforderlich, die den Transport aus der DMZ in das interne Netz erlaubt:

```
# Erlaube den Transport von E-Mail aus der DMZ in das interne Netz
$IPTABLES -A FORWARD -i $DMZDEV -o $INTDEV -d $MAILINTERN -s $MAIL -p tcp ↵
         --dport 25 -m state --state NEW -j ACCEPT
```

Des Weiteren benötigen natürlich der E-Mailserver, der DNS-Server und der Proxy einen Zugriff auf das Internet. Wenn kein eigener cachender DNS-Server in der DMZ betrieben werden soll, benötigen diese Systeme natürlich auch einen Zugriff auf einen DNS-Server in dem Internet.

```
# Erlaube dem DNS-Server einen Zugriff auf Nameserver im Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p tcp --dport 53 -m ↵
        state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p udp --dport 53 -m ↵
        state --state NEW -j ACCEPT

# Erlaube dem Proxy Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $PROXY -p tcp -m multiport ↵
        --dport 21,80,443 -m state --state NEW -j ACCEPT

# Erlaube dem E-Mailserver Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $MAIL -p tcp --dport 25 -m ↵
        state --state NEW -j ACCEPT
```

Wenn die Benutzer über den Proxy sowohl auf FTP- als auch auf HTTP- und HTTPS-Server zugreifen sollen, muss der Proxy auch auf entsprechende Ports zugreifen dürfen.

Das FTP-Protokoll ist ein sehr kompliziertes Protokoll. Daher wird es in einem eigenen Abschnitt genauer betrachtet (siehe Abschnitt 32.10). Dieses Protokoll verwendet eine Steuerungsverbindung, die üblicherweise auf dem TCP-Port 21 terminiert wird, und zusätzlich für jede zu übertragende Datei eine eigene Datenverbindung, deren Ports dynamisch ausgehandelt werden. Da es sehr schwer ist, statische Regeln für derartig dynamisch ausgehandelte und damit unbekannte Ports zu definieren, gibt es die Stateful-Inspection. Diese kann den Inhalt der Steuerungsverbindung mitlesen und die dynamisch ausgehandelten Ports erkennen. So können dann automatisch die ausgehandelten Datenverbindungen erlaubt werden. Sie müssen in Ihrem Skript dann nur noch die Steuerungsverbindung erlauben.

Damit das funktioniert, müssen Sie noch das Kernelmodul `nf_conntrack_ftp` laden:

```
MODPROBE=/sbin/modprobe $MODPROBE nf_conntrack_ftp
```

Wenn Sie auf Ihrer Firewall auch NAT verwenden, ist zusätzlich auch noch das weitere Kernel-Modul `nf_nat_ftp` erforderlich:

```
$MODPROBE nf_nat_ftp
```

Diese Zeilen fügen Sie am besten zu Beginn Ihres Skripts ein. Alle beobachteten Datenverbindungen werden dann auch mit dem Zustand `RELATED` versehen. Da Sie derartige Pakete in der `FORWARD`-Kette bereits akzeptieren, werden diese Verbindungen erlaubt. Weitere Informationen über das FTP-Protokoll und die Kernelmodule finden Sie in Abschnitt 32.10.

Nun fehlt noch der Zugriff aus dem Internet auf den Webserver und den E-Mailserver. Diese Zugriffe sind jetzt sehr einfach zu konfigurieren und erfolgen analog zu den bereits definierten Regeln. Der Zugriff erfolgt von außen auf die Systeme in der DMZ. Ein NAT wurde bereits konfiguriert. Es bleiben die folgenden Firewall-Regeln:

```
# Erlaube dem Internet den Zugriff auf den E-Mailserver
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m ↵
        state --state NEW -j ACCEPT

# Erlaube dem Internet den Zugriff auf den Webserver
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $WEB -p tcp -m multiport -- ↵
        dport 80,443 -m state --state NEW -j ACCEPT
```

Grundsätzlich sollte nun die komplette Funktionalität des Skripts bereits gewährleistet sein. Sinnvollerweise werden jedoch noch ein paar Protokollregeln und Ausnahmen hinzugefügt. Als Erstes sollten Sie sicherstellen, dass Anfragen auf dem Port 113/tcdp (Identd) abgelehnt und nicht verworfen werden. Ansonsten kann es zu Verzögerungen bei dem Aufbau von FTP- und SMTP-Verbindungen kommen.

```
$IPTABLES -A INPUT -p tcp --dport 113 -j REJECT
```

Damit Ihre internen Benutzer auch nicht auf einen Timeout warten müssen, wenn die Benutzer ein nicht erlaubtes Protokoll verwenden möchten, macht es Sinn, Anfragen von innen, die abgelehnt werden sollen, nicht zu verwerfen, sondern zu protokollieren und abzulehnen.

So behalten Sie immer den Überblick darüber, was Ihre Benutzer gerade treiben. Achten Sie jedoch auf die Datenschutzrichtlinien in Ihrem Unternehmen und auf die gesetzlichen Rahmenbedingungen.

```
$IPTABLES -A FORWARD -i
$INTDEV -j LOG --log-prefix
"Unerlaubt von innen: "
$IPTABLES -A FORWARD -i $INTDEV -j REJECT
$IPTABLES -A INPUT -i $INTDEV -j LOG --log-prefix
"Unerlaubt von innen: "
$IPTABLES -A INPUT -i $INTDEV -j REJECT
```

Da Sie diese Regeln an das Ende der Kette hängen, werden sie nur die Pakete betreffen, die nicht im Vorfeld der Kette von Ihren Regeln akzeptiert wurden.

Möglicherweise möchten Sie eine entsprechende Regel auch für Verbindungen von außen aufsetzen. Je nach Ihrer Internetanbindung möchte ich jedoch von einer derartigen grundsätzlichen Regel abraten, da Sie wahrscheinlich nicht die Zeit haben werden, Protokolle mit mehreren Zehntausend Einträgen pro Tag zu analysieren. Sehr aufschlussreich ist aber noch eine derartige Regel für Verbindungen aus der DMZ, die Sie nicht explizit in Ihren Regeln erlaubt haben. Damit sind Sie in der Lage, sehr früh Probleme in Ihrem Regelwerk oder auf Ihren Systemen in der DMZ zu erkennen.

```
$IPTABLES -A FORWARD -i $DMZDEV -j LOG --log-prefix
"Unerlaubt aus der DMZ: "
$IPTABLES -A FORWARD -i $DMZDEV -j REJECT
$IPTABLES -A INPUT -i $DMZDEV -j LOG --log-prefix
"Unerlaubt aus der DMZ: "
$IPTABLES -A INPUT -i $DMZDEV -j REJECT
```

Diese Regeln können ein Intrusion-Detection-System (IDS) sehr gut komplettieren und bei der Analyse eines Angriffs oder Einbruchs sinnvolle zusätzliche Informationen bieten.

Damit sollte das Skript schließlich fertig sein. Sicherlich haben Sie noch Ideen und Wünsche, wie Sie das Skript erweitern können. Vielleicht haben Sie auch eine vierbeinige Firewall mit zwei DMZ. Dieses Skript sollte Ihnen jedenfalls eine gute Ausgangsposition für eigene Anpassungen geben. Im Folgenden ist das komplette Skript noch einmal zur Referenz abgedruckt.

Listing 8.2: **Eine dreibeinige Firewall mit DMZ**

```
#!/bin/bash
# Autor : Ralf Spenneberg
# Version: 0.1
# Datum : 17. Dez 2004
# Dieses Skript lädt die Regeln für die Firewall
```

```
INTDEV="eth1"
# Achtung: Beim Einsatz von User-Mode-Linux lautet diese Variable
# INTDEV="tap0"

EXTDEV="eth0"

DMZDEV="eth2"

IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
MODPROBE=/sbin/modprobe

MAIL=192.168.255.2
WEB=192.168.255.3
DNS=192.168.255.4
PROXY=192.168.255.5
PROXYPORT=3128
EXTERN=3.0.0.1

# Lade die Kernelmodule für die Stateful-Inspection des FTP-Protokolls
$MODPROBE nf_conntrack_ftp
$MODPROBE nf_nat_ftp

# Setze Default-Regeln
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Lösche die Filter-Tabelle
$IPTABLES -F

# Lösche die NAT-Tabelle
$IPTABLES -t nat -F

## NAT-Regeln ##
# (1) Proxy-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $PROXY -j SNAT --to-source ↵
        $EXTERN
# (2) DNS-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $DNS -j SNAT --to-source ↵
        $EXTERN
```

```
# (3) Mailserver-Zugriff nach außen
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $MAIL -j SNAT --to-source ↩
        $EXTERN
# (4) Webserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN -m multiport ↩
        --dport 80,443 -j DNAT --to-destination $WEB
# (5) Mailserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN --dport 25 -j ↩
         DNAT --to-destination $MAIL

## Firewall-Regeln ##
# Akzeptiere alle aufgebauten Verbindungen
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
# Erlaube den Zugriff auf den Proxy
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $PROXY -p tcp --dport ↩
        $PROXYPORT -m state --state NEW -j ACCEPT

# Erlaube den Zugriff auf den E-Mailserver
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m ↩
        state --state NEW -j ACCEPT

### Bei Verwendung eines internen E-Mailservers:
# MAILINTERN=192.168.0.100
# Erlaube den Zugriff auf den E-Mailserver
# $IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -s $MAILINTERN -d $MAIL -p ↩
        tcp --dport 25 -m state --state NEW -j ACCEPT
# Erlaube den Transport von E-Mail aus der DMZ in das interne Netz
# $IPTABLES -A FORWARD -i $DMZDEV -o $INTDEV -d $MAILINTERN -s $MAIL -p ↩
        tcp --dport 25 -m state --state NEW -j ACCEPT
###
# Erlaube dem DNS-Server einen Zugriff auf Nameserver im Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p tcp --dport 53 -m ↩
        state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p udp --dport 53 -m ↩
        state --state NEW -j ACCEPT

# Erlaube dem Proxy den Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $PROXY -p tcp -m multiport ↩
        --dport 21,80,443 -m state --state NEW -j ACCEPT

# Erlaube dem E-Mailserver den Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $MAIL -p tcp --dport 25 -m ↩
        state --state NEW -j ACCEPT
```

```
# Erlaube dem Internet den Zugriff auf den E-Mailserver
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m ↵
        state --state NEW -j ACCEPT

# Erlaube dem Internet den Zugriff auf den Webserver
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $WEB -p tcp -m multiport -- ↵
        dport 80,443 -m state --state NEW -j ACCEPT

# Lehne Identd-Anfragen ab
$IPTABLES -A INPUT -p tcp --dport 113 -j REJECT

# Protokolliere alle anderen Anfragen von innen
$IPTABLES -A FORWARD -i $INTDEV -j LOG --log-prefix "Unerlaubt von innen: ↵
         "
$IPTABLES -A FORWARD -i $INTDEV -j REJECT $IPTABLES -A INPUT -i $INTDEV - ↵
        j LOG --log-prefix "Unerlaubt von innen: "
$IPTABLES -A INPUT -i $INTDEV -j REJECT
$IPTABLES -A FORWARD -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der ↵
        DMZ: "
$IPTABLES -A FORWARD -i $DMZDEV -j REJECT $IPTABLES -A INPUT -i $DMZDEV - ↵
        j LOG --log-prefix "Unerlaubt aus der DMZ: "
$IPTABLES -A INPUT -i $DMZDEV -j REJECT
```

In seiner jetzigen Form ist das Skript bereits durchaus lang und wahrscheinlich im ersten Moment auch für den einen oder anderen Administrator durchaus unübersichtlich. Im nächsten Abschnitt werden Sie sehen, wie diese Skripten lesbarer und wartbarer gestaltet werden können. Wir werden das Skript in Bezug auf seine Lesbarkeit und die Geschwindigkeit der Verarbeitung optimieren.

8.3 Optimierung mit benutzerdefinierten Ketten

Im letzten Abschnitt haben wir bereits ein recht umfangreiches Skript für eine dreibeinige Firewall mit DMZ entwickelt. Dieses Skript war bereits fast 100 Zeilen lang. Da das Skript für fast jede veränderliche Information (wie IP-Adressen) Variablen nutzt, ist eine weitere Wartung recht einfach. Dennoch ist es durchaus schwierig, einen Überblick zu behalten und den genauen Paketfluss nachzuvollziehen. Einfacher kann das geschehen, wenn Sie sich Ihre eigenen Ketten für jede Funktion erzeugen. Iptables bietet, wie früher schon Ipchains, die Möglichkeit, neue Ketten mit einem eigenen Namen zu definieren und diese dann mit Regeln zu füllen.

Diese benutzerdefinierte Kette weist mehrere Vorteile auf:

» Sie können durch eine sinnvolle Strukturierung der Regeln und anschließend durch eine entsprechende Auswahl der Ketten die Gesamtzahl der auszuwertenden Regeln pro Paket

senken und damit die Abarbeitung der Regeln pro Paket stark beschleunigen. Stellen Sie sich vor, Ihre Firewall verfügte über 100 Regeln. Davon beträfen jeweils 50 Regeln TCP- und UDP-Pakete. Im besten Fall trifft die erste Regel bereits auf das aktuell betrachtete Paket zu, im schlechtesten Fall aber erst die Regel 100. Dabei müssten Sie bei einem TCP-Paket die UDP-Regeln gar nicht betrachten und umgekehrt. Also erzeugen Sie sich zwei neue Ketten: `mein_udp` und `mein_tcp`. Anschließend verschieben Sie alle TCP-Regeln in die Kette `mein_tcp` und alle UDP-Regeln in die Kette `mein_udp`. Nun müssen Sie nur noch in der originalen Kette prüfen, ob es sich um ein TCP-Paket handelt. Dann springen Sie in die Kette `mein_tcp`. In dem anderen Fall springen Sie in die Kette `mein_udp`. Nun wird Ihr Paket im besten Fall von der Regel Zwei bereits akzeptiert (die erste Regel prüft ja, ob es ein TCP- oder UDP-Paket ist). Im schlechtesten Fall müssen aber nur 51 Regeln abgearbeitet werden. Die Gesamtzahl der pro Paket auszuwertenden Regeln sinkt also drastisch. Diesen Arbeitsaufwand können Sie möglicherweise durch eine weitere Aufteilung weiter senken.

» Die benutzerdefinierten Ketten erlauben Ihnen auch eine Optimierung der Lesbarkeit. Sie können zum Beispiel drei Ketten erzeugen: `mein_extern`, `mein_dmz` und `mein_intern`. Sobald ein Paket aus dem entsprechenden Netz kommt, springen Sie in die jeweilige Kette. Dies erlaubt Ihnen eine logische Gruppierung der Regeln. Wenn Sie wissen möchten, was die DMZ verlassen darf, so müssen Sie nur noch in eine Kette schauen.

» Schließlich ist es auch in bestimmten Umgebungen mit benutzerdefinierten Ketten leichter möglich, Veränderungen an den Firewall-Regeln vorzunehmen. Vor einigen Jahren wurde ich gebeten, eine Firewall für ein größeres IT-Schulungsunternehmen aufzusetzen. Diese Firewall sollte neben dem Netz für die Personalverwaltung und den Vertrieb und dem Labornetz auch die Netze der einzelnen Schulungsräume untereinander und im Zugriff auf das Internet kontrollieren und regeln. Aufgrund der verschiedenen Themen, die in dem Unternehmen geschult wurden, waren häufig Änderungen an den Firewall-Regeln erforderlich. Die Administration sollte jedoch von Personen durchgeführt werden, die über keinerlei oder nur geringe Kenntnisse verfügten. Daher implementierte ich für jeden Schulungsraum eine eigene benutzerdefinierte Kette, in der die Regeln für den Zugriff der Rechner dieses Raumes auf das Internet definiert wurden. Zusätzlich schrieb ich ein einfaches Skript, das den Benutzer nach der Raumnummer fragte und anschließend fünf vordefinierte Firewall-Regelsätze anbot. Nach der Auswahl löschte das Skript nur die Regeln in der benutzerdefinierten Kette dieses Raumes und fügte die neuen Regeln ein. Dadurch war sichergestellt, dass alle weiteren Räume und auch die weiteren Netze zur Verwaltung der Firma ungestört weiterarbeiten konnten. Dieser Vorgang wäre ohne benutzerdefinierte Ketten ungleich komplizierter gewesen, denn ein kompletter Neustart des Firewall-Skripts kam nicht in Frage.

Bevor wir uns aber mit einer praktischen Implementierung beschäftigen, wollen wir zunächst schauen, wie die benutzerdefinierten Ketten arbeiten. Wenn Sie die Beispiele nachstellen wollen, sollten Sie zunächst sämtliche auf Ihrem System geladenen Firewall-Regeln löschen und die Default-Policies der Ketten auf ACCEPT setzen. Auf einem Fedora/RedHat-System kann das zum Beispiel recht komfortabel mit dem Befehl `service Iptables stop` erfolgen. Ansonsten können Sie ein Skript wie Listing 5.2 auf Seite 89 verwenden.

Sie können eine neue benutzerdefinierte Kette mit dem Befehl `iptables` erzeugen. Hierzu verwenden Sie einfach:

```
[root@bibo ~]# iptables -N meinekette
[root@bibo ~]# iptables -vnL Chain FORWARD (policy ACCEPT 0 packets, 0 ↵
        bytes)
pkts bytes target prot opt in out source destination

Chain INPUT (policy ACCEPT 93 packets, 58921 bytes)
pkts bytes target prot opt in out source destination

Chain OUTPUT (policy ACCEPT 92 packets, 7147 bytes)
pkts bytes target prot opt in out source destination

Chain meinekette (0 references)
pkts bytes target prot opt in out source destination
```

Sie sehen am unteren Ende der Ausgabe des Befehls `iptables -vnL` Ihre neue Kette. Sofort fällt auf, dass diese Kette scheinbar keine Policy hat. Anstelle der Policy wird hier die Anzahl der Referenzen angegeben. Damit Sie erkennen, was eine Referenz ist, erzeugen wir nun eine Regel, die die Kette nutzt:

```
[root@bibo ~]# iptables -A INPUT -j meinekette
[root@bibo ~]# iptables -vnL Chain FORWARD (policy ACCEPT 0 packets, 0 ↵
        bytes)
pkts bytes target prot opt in out source destination

Chain INPUT (policy ACCEPT 654 packets, 243K bytes)
pkts  bytes  target      prot  opt  in  out  source     destination
299   38464  meinekette  all   --   *   *    0.0.0.0/0  0.0.0.0/0

Chain OUTPUT (policy ACCEPT 649 packets, 60506 bytes)
 pkts bytes target prot opt in out source destination

Chain meinekette (1 references)
pkts bytes target prot opt in out source destination
```

In der Kette INPUT ist nun eine Regel vorhanden, die auf die Kette `meinekette` verweist. Dadurch wird die Kette `meinekette` in einer weiteren Kette referenziert. Dies wird in dem Referenzzähler in der Kette `meinekette` angezeigt. Solange die Kette `meinekette` noch irgendwo referenziert wird, können Sie diese nicht entfernen. Die benutzerdefinierte Kette können Sie genauso entfernen, wie Sie diese auch erzeugen. Hierfür verwenden Sie lediglich die Option `-X`.

```
[root@bibo ~]# iptables -X meinekette
iptables: Too many links
```

Hier werden Sie darauf hingewiesen, dass noch eine Verknüpfung mit dieser Kette existiert. Diese müssen Sie zuerst löschen. Es gibt keine Suchfunktion für diese Referenz. Sie können sich nur durch den `grep`-Befehl unterstützen lassen: `iptables -vnL | grep meinekette`.

Nun fügen wir der Kette `meinekette` zunächst eine Regel hinzu:

```
[root@bibo ~]# iptables -A meinekette -j DROP
```

Dann löschen wir die Referenz:

```
[root@bibo ~]# iptables -F INPUT
[root@bibo ~]# iptables -vnL Chain FORWARD (policy ACCEPT 0 packets, 0 ↵
        bytes)
pkts bytes target prot opt in out source destination

Chain INPUT (policy ACCEPT 1427 packets, 612K bytes)
pkts bytes target prot opt in out source destination

Chain OUTPUT (policy ACCEPT 1455 packets, 135K bytes)
pkts bytes target prot opt in out source destination

Chain meinekette (0 references)
pkts  bytes  target  prot  opt  in  out  source     destination
0     0      DROP    all   --   *   *    0.0.0.0/0  0.0.0.0/0
```

Wenn Sie nun versuchen, die Kette zu löschen, wird der Versuch wieder fehlschlagen, da die Kette noch nicht leer ist. Erst nach dem Flush der Kette kann die benutzerdefinierte Kette gelöscht werden.

```
[root@bibo ~]# iptables -X meinekette
iptables: Directory not empty
[root@bibo ~]# iptables -F meinekette
[root@bibo ~]# iptables -X meinekette
```

Damit ist schon fast alles zur Administration der benutzerdefinierten Ketten gesagt. Es bleibt der Verweis auf die Möglichkeit, dass Sie eine benutzerdefinierte Kette umbenennen können. Hierzu gibt es den folgenden Befehl:

```
[root@bibo ~]# iptables -E meinekette neuername
```

Eine Frage ist aber dennoch offen: Was passiert am Ende der benutzerdefinierten Kette, wenn es keine Standardrichtlinien (Policy) gibt? Antwort: Die benutzerdefinierte Kette arbeitet ähnlich einem Unterprogramm. Nach ihrer Abarbeitung springt sie in die aufrufende Kette zurück, die weiter abgearbeitet wird. Wenn Sie dieses Verhalten verhindern möchten, müssen Sie am Ende der benutzerdefinierten Kette eine eigene Regel vorsehen, die alle nicht von expliziten Regeln betrachteten Pakete betrifft. Das könnte zum Beispiel der folgende Iptables-Befehl erreichen:

```
[root@bibo ~]# iptables -A meinekette -j DROP
```

Diese Regel würde alle weiteren Pakete in dieser Kette verwerfen. Ein Rücksprung in die aufrufende Kette würde nicht mehr erfolgen, da für alle Pakete in dieser Kette eine Entscheidung getroffen wurde. Ob Sie die restlichen Pakete verwerfen (`DROP`), ablehnen (`REJECT`) oder akzeptieren (`ACCEPT`), hängt sicherlich von der Logik Ihrer Firewall ab. Genauso kann es jedoch sein, dass Sie bei bestimmten Paketen einen sofortigen Rücksprung in die aufrufende Kette veranlassen möchten. Auch dies ist möglich. Hierfür gibt es das Ziel `RETURN`. Damit können Sie die benutzerdefinierte Kette sofort verlassen und in der aufrufenden Kette die weiteren Regeln betrachten.

```
[root@bibo ~]# iptables -A meinekette -j RETURN
```

8.3.1 Anwendung der benutzerdefinierten Ketten

Nun wollen wir die benutzerdefinierten Ketten nutzen, um das Skript der dreibeinigen Firewall übersichtlicher und wartbarer zu gestalten. Wenn Sie das entsprechende Kapitel[TS d] nicht gelesen haben, möchte ich Ihnen nun ans Herz legen, es kurz querzulesen, damit Sie den Sinn des folgenden Skripts und des Umbaus kennen.

Zunächst sollten wir uns überlegen, welche Ketten wir benutzerdefiniert anlegen möchten. Es gibt grundsätzlich mehrere konzeptionelle Möglichkeiten:

1. Wir erzeugen für jedes Netz eine Kette. In dieser Kette wird definiert, welche Verbindungen das Netz aufbauen darf. Dies trennt die Regeln also entsprechend der Herkunft der Verbindungen auf. Die drei Ketten heißen dann: `DMZ`, `LAN` und `INTERNET`.
2. Wir erzeugen für jedes IP-Protokoll eine Kette. So gibt es drei Ketten: `TCP`, `UDP` und `ICMP`. Dies kann in bestimmten Umgebungen von Vorteil sein. Hier ist es das sicher nicht.
3. Wir erzeugen für jeden Informationsfluss eine Kette. Das bedeutet, dass wir eine Kette für den Zugriff auf unseren Webserver erzeugen, eine Kette für den E-Mail-Verkehr und eine Kette für den Proxy. Sobald wir Änderungen an der Art und Weise der E-Mail-Zustellung vornehmen wollen, müssen wir nur in der entsprechenden Kette die Änderungen vornehmen.

STOP

In der Vergangenheit war es nicht möglich, eine Kette `icmp` zu erzeugen und zu benutzen, da dieses Schlüsselwort bereits belegt ist. Auch heute sollten Sie davon Abstand nehmen, um Namenskonflikte zu vermeiden. Sie können diese Namenskonflikte sehr einfach vermeiden, indem Sie Großbuchstaben für Ihre Ketten verwenden oder jeder Kette einen Buchstaben voranstellen.

Welche der drei Konzepte Sie für Ihre zukünftigen Firewalls verwenden, möchte ich Ihnen überlassen. Ich wähle für die weitere Erläuterung den Fall Eins, bei dem drei Ketten (`DMZ`, `LAN` und `INTERNET`) erzeugt werden.

Das Skript 8.2 kann nun umgeschrieben werden. Dabei werden die ersten Zeilen inklusive der NAT-Regeln unverändert übernommen und anschließend die benutzerdefinierten Ketten erzeugt.

TS d Bitte Nummer ergänzen.

```
#!/bin/sh
# Autor : Ralf Spenneberg
# Version: 0.1
# Datum : 17. Dez 2004
# Dieses Skript lädt die Regeln für die Firewall

INTDEV="eth1"
# Achtung: Beim Einsatz von User-Mode-Linux lautet diese Variable
# INTDEV="tap0"

EXTDEV="eth0"

DMZDEV="eth2"

IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
MODPROBE=/sbin/modprobe

MAIL=192.168.255.2
WEB=192.168.255.3
DNS=192.168.255.4
PROXY=192.168.255.5
PROXYPORT=3128
EXTERN=3.0.0.1

# Lade die Kernelmodule für die Stateful-Inspection des FTP-Protokolls
$MODPROBE nf_conntrack_ftp
$MODPROBE nf_nat_ftp

# Setze Default-Regeln
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Lösche die Filter-Tabelle
$IPTABLES -F

# Lösche die NAT-Tabelle
$IPTABLES -t nat -F
```

```
## NAT-Regeln ##
# (1) Proxy-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $PROXY -j SNAT --to-source ↵
        $EXTERN
# (2) DNS-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $DNS -j SNAT --to-source ↵
        $EXTERN
# (3) Mailserver-Zugriff nach außen
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $MAIL -j SNAT --to-source ↵
        $EXTERN
# (4) Webserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN -m multiport ↵
        --dport 80,443 -j DNAT --to-destination $WEB
# (5) Mailserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN --dport 25 -j ↵
         DNAT --to-destination $MAIL

# Erzeuge benutzerdefinierte Ketten
$IPTABLES -N DMZ
$IPTABLES -N LAN
$IPTABLES -N INTERNET

## Firewall-Regeln ##
# Akzeptiere alle aufgebauten Verbindungen
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Nun können die Regeln auf die Ketten verteilt werden und die Ketten angesprungen werden.

```
##########################################################
#
# Zugriff aus dem LAN
#
# Erlaube den Zugriff auf den Proxy
$IPTABLES -A LAN -o $DMZDEV -d $PROXY -p tcp --dport $PROXYPORT -m state ↵
        --state NEW -j ACCEPT

# Erlaube den Zugriff auf den E-Mailserver
$IPTABLES -A LAN -o $DMZDEV -d $MAIL -p tcp --dport 25 -m state --state ↵
        NEW -j ACCEPT

### Bei Verwendung eines internen E-Mailservers:
# MAILINTERN=192.168.0.100
# Erlaube den Zugriff auf den E-Mailserver
```

```
# $IPTABLES -A LAN -o $DMZDEV -s $MAILINTERN -d $MAIL -p tcp --dport 25 - ↵
        m state --state NEW -j ACCEPT

# Protokolliere alle anderen Anfragen von innen
$IPTABLES -A LAN -j LOG --log-prefix "Unerlaubt von innen: "
$IPTABLES -A LAN -j REJECT

#######################################################
#
# Zugriff aus der DMZ
#
# Erlaube den Transport von E-Mail aus der DMZ in das interne Netz
# $IPTABLES -A DMZ -o $INTDEV -d $MAILINTERN -s $MAIL -p tcp --dport 25 - ↵
        m state --state NEW -j ACCEPT
###
# Erlaube dem DNS-Server einen Zugriff auf Nameserver im Internet
$IPTABLES -A DMZ -o $EXTDEV -s $DNS -p tcp --dport 53 -m state --state  ↵
        NEW -j ACCEPT
$IPTABLES -A DMZ -o $EXTDEV -s $DNS -p udp --dport 53 -m state --state  ↵
        NEW -j ACCEPT

# Erlaube dem Proxy den Zugriff auf das Internet
$IPTABLES -A DMZ -o $EXTDEV -s $PROXY -p tcp -m multiport --dport  ↵
        21,80,443 -m state --state NEW -j ACCEPT

# Erlaube dem E-Mailserver den Zugriff auf das Internet
$IPTABLES -A DMZ -o $EXTDEV -s $MAIL -p tcp --dport 25 -m state --state  ↵
        NEW -j ACCEPT

# Protokolliere alle anderen Zugriffe aus der DMZ
$IPTABLES -A DMZ -j LOG --log-prefix "Unerlaubt aus der DMZ: "
$IPTABLES -A DMZ -j REJECT

#############################################
#
# Zugriff aus dem Internet
#
# Erlaube dem Internet den Zugriff auf den E-Mailserver
$IPTABLES -A INTERNET -o $DMZDEV -d $MAIL -p tcp --dport 25 -m state -- ↵
        state NEW -j ACCEPT
```

```
# Erlaube dem Internet den Zugriff auf den Webserver
$IPTABLES -A INTERNET -o $DMZDEV -d $WEB -p tcp -m multiport --dport ↵
        80,443 -m state --state NEW -j ACCEPT
# Rufe die Ketten auf

# FORWARD
$IPTABLES -A FORWARD -i $INTDEV -j LAN
$IPTABLES -A FORWARD -i $DMZDEV -j DMZ
$IPTABLES -A FORWARD -i $EXTDEV -j INTERNET

# INPUT
$IPTABLES -A INPUT -p tcp --dport 113 -j REJECT
$IPTABLES -A INPUT -i $INTDEV -j LAN
$IPTABLES -A INPUT -i $DMZDEV -j DMZ
```

Wenn Sie sich nun Ihre `FORWARD`-Kette ansehen, werden Sie feststellen, dass diese sehr übersichtlich geworden ist. Es sind nur noch 4 Regeln vorhanden:

```
[root@bibo ~]# iptables -vnL FORWARD
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts  bytes  target    prot  opt  in    out   source     destination  state
0     0      ACCEPT    all   --   *     *     0.0.0.0/0  0.0.0.0/0     ↵
        RELATED,ESTABLISHED
0     0      LAN       all   --   eth1  *     0.0.0.0/0  0.0.0.0/0
0     0      DMZ       all   --   eth2  *     0.0.0.0/0  0.0.0.0/0
0     0      INTERNET  all   --   eth0  *     0.0.0.0/0  0.0.0.0/0
```

Sie können sehr leicht feststellen, welchen Weg das erste Paket einer neuen Verbindung durch Ihr Regelwerk nimmt. Denken Sie daran, dass alle Pakete, die zu aufgebauten Verbindungen gehören, bereits von der ersten Regel in dieser Kette akzeptiert werden. Alle Verbindungsaufbauten werden entsprechend der Netzwerkkarte, über die diese Pakete eingehen, auf die verschiedenen Ketten verteilt. Erreicht das Paket über die Netzwerkkarte `eth2` das System, so wird es an die Kette `DMZ` übergeben, die das Paket prüft, gegebenenfalls akzeptiert oder protokolliert und verwirft.

Falls Sie nun die genauen Regeln betrachten möchten, die das Paket dann prüfen, so genügt es, die Kette `DMZ` anzuzeigen.

```
[root@bibo ~]# iptables -vnL DMZ
Chain DMZ (2 references)
pkts  bytes  target    prot  opt  in    out   source         destination  ↵
                                   state
0     0      ACCEPT    tcp   --   *     eth0  192.168.255.4  0.0.0.0/0 tcp  ↵
        dpt:53                  NEW
0     0      ACCEPT    udp   --   *     eth0  192.168.255.4  0.0.0.0/0 udp  ↵
        dpt:53                  NEW
```

```
0     0      ACCEPT    tcp   --   *    eth0 192.168.255.5  0.0.0.0/0  ↵
        multiport dports 21,80,443  NEW
0     0      ACCEPT    tcp   --   *    eth0 192.168.255.2  0.0.0.0/0 tcp  ↵
        dpt:25                  NEW
0     0      LOG       all   --   *    *    0.0.0.0/0      0.0.0.0/0  ↵
                                   LOG flags 0 level 4 prefix ‘ ↵
        Unerlaubt aus der DMZ:’
0     0      REJECT    all   --   *    *    0.0.0.0/0      0.0.0.0/0  ↵
        reject-with                icmp-port-unreachable
```

Der Vorteil der benutzerdefinierten Ketten in puncto Lesbarkeit sollte Ihnen nun klar geworden sein. Gleichzeitig haben wir aber auch etwas für die Optimierung der Geschwindigkeit getan. Ein Paket, das eine neue Verbindung aus dem Internet zu dem Webserver starten möchte, hätte bei dem alten Firewall-Skript von 10 Regeln betrachtet werden müssen, bevor es akzeptiert worden wäre. Dies kostet Zeit. Nun handelt es sich lediglich um 4 Regeln in der `FORWARD`-Kette und dann um 2 Regeln in der Kette `INTERNET`. Insgesamt sind es also nur 6 Regeln. Bei einer derart geringen Regelzahl ist der Vorteil nur minimal, und häufig ist nicht das Firewall-System das Nadelöhr bei dem Pakettransport, sondern die Bandbreiten der beteiligten Netzwerkverbindungen. Stellen Sie sich aber eine Firewall mit mehreren Hundert Regeln vor, die mit Gigabit-Geschwindigkeit die Pakete verarbeiten soll. Hier kann die Firewall sehr wohl ein Nadelöhr darstellen.

Natürlich ist streng genommen nicht die Anzahl der Regeln, sondern die Anzahl der durchzuführenden Prüfungen interessant. Die meisten Regeln prüfen nicht nur eine Eigenschaft des Pakets, sondern meist gleich mehrere. Diese Prüfungen dauern leider nicht alle gleich lang, sodass objektive Berechnungen nicht möglich sind. Die Anzahl der Prüfungen erlaubt jedoch eine grobe Abschätzung des Aufwands.

8.4 DMZ mit zwei Paketfiltern

Eine dreibeinige Firewall mit DMZ bietet bereits einen recht ordentlichen Schutz. Jedoch besteht immer die Gefahr, dass ein Angreifer auf dem System einen Konfigurationsfehler oder einen Bug in der verwendeten Software (zum Beispiel in dem Linux-Kernel) findet. Erlangt der Angreifer so die Kontrolle über das System, kann er von einer dreibeinigen Firewall aus jedes System in der DMZ, aber auch jedes System in dem internen LAN erreichen. Bei den Systemen in der DMZ muss ein möglicher Angriff mit anschließendem Einbruch immer als Risiko akzeptiert werden, da das Internet (mit Einschränkungen) auf das System zugreifen darf. CE Dies soll die Firewall für die internen Systeme im LAN aber gerade verhindern.

Sie können dazu eine zweistufige Firewall mit zwei Paketfiltern einsetzen. Fällt der erste Paketfilter in die Hände des Angreifers, so schützt der zweite Paketfilter noch das interne LAN.

Natürlich besteht die Gefahr, dass der Angreifer auch in den zweiten Paketfilter eindringt. Als Firewall-Administrator hofft man jedoch, dass dies so viel Zeit kostet, dass man noch

CE Bitte prüfen.

Editor's or typesetter's annotations (will be removed before the final TEX run)

Gegenmaßnahmen einleiten kann. Das kann zum Beispiel eine Unterbrechung der Netzwerkverbindung sein. Häufig wird aus diesem Grund auch empfohlen, zwei unterschiedliche Produkte als Paketfilter einzusetzen.[1] So kann ein Angreifer nicht denselben Softwarefehler für den Einbruch in beiden Paketfiltern nutzen, sondern muss nach neuen Lücken suchen. Auch Konfigurationsfehler treten selten auf beiden Systemen gleichzeitig auf. Dies erhöht also die Schwierigkeit für den Angreifer. Da dies jedoch ein Iptables-Buch ist, möchte ich mich hier darauf beschränken, die Empfehlung zu erwähnen, und es Ihnen überlassen, ob Sie dieser Empfehlung folgen und welches zusätzliche Produkt Sie auswählen.

Die Implementierung der zweistufigen Firewall erfolgt so wie in Abbildung 8.1.

Wie bereits zu Beginn des Kapitels erwähnt wurde (siehe Abschnitt 8.1), befinden sich in der DMZ der Beispielfirma *Nohup.info* jeweils ein Web-, ein E-Mail-, ein cachender DNS- und ein Proxy-Server. Diese Systeme erhalten private IP-Adressen aus dem Bereich 192.168.255.0/24. Dabei soll aus dem Internet ein Zugriff auf den Web- und den E-Mailserver möglich sein. Aus dem internen Netz ist ein Zugriff auf den Proxy- und den E-Mailserver notwendig. Der Zugriff auf den Webserver wird über den Proxy-Server realisiert. Ein direkter Zugriff auf das Internet ist nicht vorgesehen. Der äußere Paketfilter erhält als externe IP-Adresse eine feste statische IP-Adresse, 3.0.0.1. Alle Dienste werden im Internet unter dieser IP-Adresse angebunden. Das bedeutet, dass ein DNS-Server bei der Frage nach der Adresse `www.nohup.info` die IP-Adresse 3.0.0.1 zurückliefert.

Teilen wir nun die Implementierung auf. Zunächst besprechen wir die Regeln für den äußeren Paketfilter, anschließend die für den inneren Paketfilter.

Wenn Sie den Hardwareaufwand für zwei Paketfilter scheuen und Ihr System über mindestens drei Netzwerkkarten verfügt, können Sie die beiden Paketfilter auch als virtuelle Systeme auf der Hardware laufen lassen. Hierfür können Sie VMware, User-Mode-Linux, KVM oder Xen verwenden. Alle diese Systeme sind in der Lage, die Hardware zu virtualisieren und voneinander getrennte Linux-Systeme auf derselben Hardware zu ermöglichen. Die Sicherheit ist gewiss nicht so hoch wie bei zwei physikalisch getrennten Systemen, aber besser als bei einer dreibeinigen Firewall.

8.4.1 Der äußere Paketfilter

Der äußere Paketfilter muss den Zugriff des Internets auf den Webserver und den E-Mailserver in der DMZ erlauben. Außerdem muss er dem Proxy, dem DNS-Server und dem E-Mailserver den Zugriff auf die Ressourcen des Internets ermöglichen.

Das Skript beginnt wieder mit der üblichen Definition der Variablen. Die zwei vorhandenen Netzwerkkarten erhalten die Variablennamen `$EXTDEV` und `$DMZDEV`. Ansonsten kann der Kopf des Skripts für die dreibeinige Firewall unverändert übernommen werden.

1 Das BSI (Bundesamt für Sicherheit in der Informationstechnik) spricht zum Beispiel diese Empfehlung aus (`https://www.bsi.bund.de//ContentBSI/Themen/Internet_Sicherheit/Sicherheitskomponenten/Sicherheitsgateway/konzsichgateway.html`)

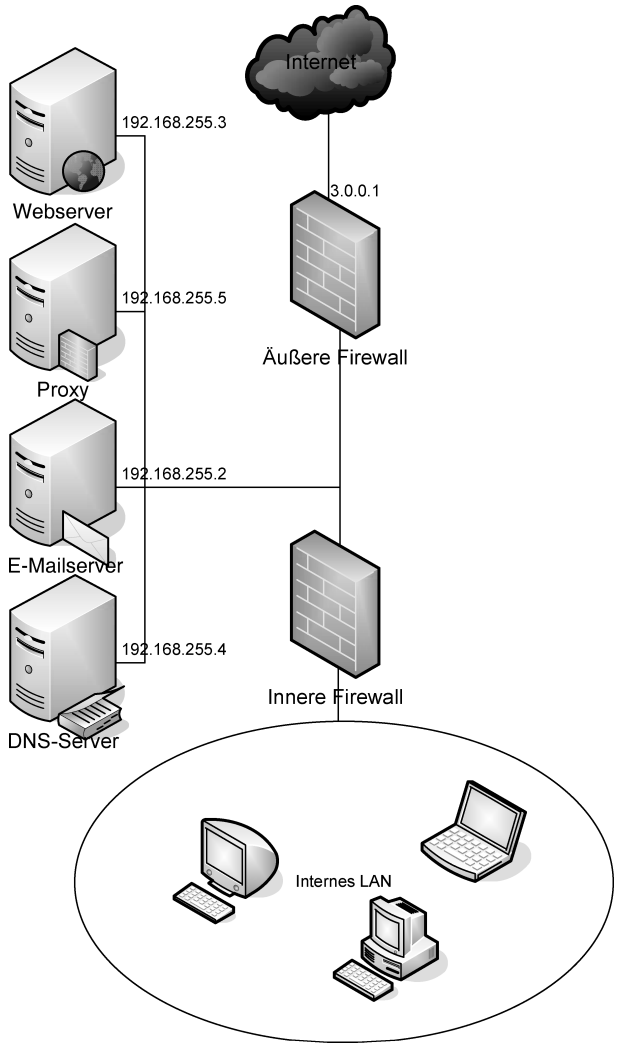


Abbildung 8.1: **Eine zweistufige Firewall schützt die DMZ.**

```
#!/bin/sh
# Autor : Ralf Spenneberg
# Version: 0.1
# Datum : 17. Dez 2004
# Dieses Skript lädt die Regeln für die Firewall

EXTDEV="eth0"

DMZDEV="eth1"
```

```
IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
MODPROBE=/sbin/modprobe

MAIL=192.168.255.2
WEB=192.168.255.3
DNS=192.168.255.4
PROXY=192.168.255.5
EXTERN=3.0.0.1

# Lade die Kernelmodule für die Stateful-Inspection des FTP-Protokolls
$MODPROBE nf_conntrack_ftp
$MODPROBE nf_nat_ftp

# Setze Default-Regeln
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Lösche die Filter-Tabelle
$IPTABLES -F

# Lösche die NAT-Tabelle
$IPTABLES -t nat -F
```

Die äußere Firewall ist auch für ein NAT der Adressen zum Internet hin zuständig. Hier können die NAT-Regeln aus dem Skript für die dreibeinige Firewall unverändert übernommen werden:

```
## NAT-Regeln ##
# (1) Proxy-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $PROXY -j SNAT --to-source ↵
        $EXTERN
# (2) DNS-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $DNS -j SNAT --to-source ↵
        $EXTERN
# (3) Mailserver-Zugriff nach außen
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $MAIL -j SNAT --to-source ↵
        $EXTERN
# (4) Webserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN -m multiport ↵
        --dport 80,443 -j DNAT --to-destination $WEB
# (5) Mailserver-Zugriff von außen
```

```
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN --dport 25 -j ↵
         DNAT --to-destination $MAIL
```

Diese Regeln sorgen dafür, dass bei einem Zugriff auf das Internet die Quell-IP-Adressen des DNS-, E-Mail- und Proxy-Servers genattet werden. Auch bei einem Zugriff aus dem Internet auf den E-Mail- und den Webserver wird der Zugriff genattet.

Es fehlen nun noch die Filterregeln. Auch diese können fast unverändert übernommen werden.

```
# Erlaube dem DNS-Server einen Zugriff auf Nameserver im Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p tcp --dport 53 -m  ↵
        state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p udp --dport 53 -m  ↵
        state --state NEW -j ACCEPT

# Erlaube dem Proxy den Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $PROXY -p tcp -m multiport  ↵
        --dport 21,80,443 -m state --state NEW -j ACCEPT

# Erlaube dem E-Mailserver den Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $MAIL -p tcp --dport 25 -m  ↵
        state --state NEW -j ACCEPT

# Erlaube dem Internet den Zugriff auf den E-Mailserver
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m  ↵
        state --state NEW -j ACCEPT

# Erlaube dem Internet den Zugriff auf den Webserver
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $WEB -p tcp -m multiport --  ↵
        dport 80,443 -m state --state NEW -j ACCEPT

# Lehne Identd-Anfragen ab
$IPTABLES -A INPUT -p tcp --dport 113 -j REJECT
$IPTABLES -A FORWARD -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der  ↵
        DMZ: "
$IPTABLES -A FORWARD -i $DMZDEV -j REJECT
$IPTABLES -A INPUT -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der DMZ: ↵
         "
$IPTABLES -A INPUT -i $DMZDEV -j REJECT
```

Dieses Skript genügt für die Konfiguration des äußeren Paketfilters. Ein direkter Zugriff von dem inneren Paketfilter wird nicht erlaubt. So ist der äußere Paketfilter auch vor Angriffen von innen geschützt. Wenn Sie einen zusätzlichen Administrationszugang wünschen, so können Sie zum Beispiel einen Zugang per SSH erlauben:

```
$IPTABLES -A INPUT -i $DMZDEV -p tcp --dport 22 -m state --state NEW, ↵
        ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -O $DMZDEV -m state --state ESTABLISHED -j ACCEPT
```

Natürlich könnten Sie auch diesen Zugang auf eine bestimmte IP-Adresse begrenzen. Dann ist es aber sinnvoll, für diese Adresse auch eine Variable, z.B. `ADMIN`, zu verwenden.

8.5 Der innere Paketfilter

Die Aufgabe des inneren Paketfilters ist es, den Zugriff des internen LANs auf die Systeme in der DMZ zu regeln. Je nach der gewählten E-Mail-Struktur muss er auch eine Verbindung des DMZ-E-Mailservers in das LAN ermöglichen.

Ich beschränke mich nun auf die Darstellung des kompletten Skripts und einige wenige Erklärungen. Die einzelnen Regeln wurden in den vorangegangenen Kapiteln bereits ausreichend erläutert. Die Netzwerkkarten in diesem Skript heißen `DMZDEV` und `INTDEV`. Das folgende Skript implementiert den inneren Paketfilter:

```
#!/bin/sh
# Autor : Ralf Spenneberg
# Version: 0.1
# Datum : 17. Dez 2004
# Dieses Skript lädt die Regeln für den inneren Paketfilter

INTDEV="eth1"

DMZDEV="eth2"

IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
MODPROBE=/sbin/modprobe

MAIL=192.168.255.2
PROXY=192.168.255.5
PROXYPORT=3128

# Setze Default-Regeln
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Lösche die Filter-Tabelle
$IPTABLES -F
```

```
## Firewall-Regeln
# Akzeptiere alle aufgebauten Verbindungen
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#########################################################
# Zugriff aus dem LAN
# Erlaube den Zugriff auf den Proxy
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $PROXY -p tcp --dport  ↵
        $PROXYPORT -m state --state NEW -j ACCEPT

# Erlaube den Zugriff auf den E-Mailserver
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m  ↵
        state --state NEW -j ACCEPT

### Bei Verwendung eines internen E-Mailservers:
# MAILINTERN=192.168.0.100
# Erlaube den Zugriff auf den E-Mailserver
# $IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -s $MAILINTERN -d $MAIL -p  ↵
        tcp --dport 25 -m state --state NEW -j ACCEPT

# Protokolliere alle anderen Anfragen von innen
$IPTABLES -A FORWARD -i $INTDEV -j LOG --log-prefix "Unerlaubt von innen: ↵
         "
$IPTABLES -A FORWARD -i $INTDEV -j REJECT

#########################################################
# Zugriff aus der DMZ
# Erlaube den Transport von E-Mail aus der DMZ in das interne Netz
# $IPTABLES -A FORWARD -i $DMZDEV -o $INTDEV -d $MAILINTERN -s $MAIL -p  ↵
        tcp --dport 25 -m state --state NEW -j ACCEPT
##

# Protokolliere alle anderen Zugriffe aus der DMZ
$IPTABLES -A FORWARD -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der  ↵
        DMZ: "
$IPTABLES -A FORWARD -i $DMZDEV -j REJECT
```

Dieser Paketfilter führt kein NAT durch, da kein Zugriff auf das Internet erfolgt. Dafür muss dann natürlich sichergestellt sein, dass die beteiligten Systeme über entsprechende Einträge in ihren Routingtabellen verfügen. Bei den Systemen im LAN ist dies recht einfach.

Bei ihnen wird als Default-Gateway der Paketfilter eingetragen. Bei den Systemen in der DMZ ist es erforderlich, als Default-Gateway den äußeren Paketfilter einzutragen. Lediglich für die

Verbindung zum internen LAN ist eine zusätzliche Route auf diesen Systemen erforderlich. Hier muss als Gateway der interne Paketfilter eingetragen werden.

Ich hoffe, dass ich Ihnen mit diesen Skripten ein paar Beispiele geliefert habe, die Sie nun bei der Erstellung Ihrer eigenen Skripten unterstützen. Natürlich handelt es sich hier nur um grobe Gerüste, die häufig um spezielle Anpassungen erweitert werden müssen. Diese hängen aber immer sehr von der Umgebung ab und werden daher hier nicht näher besprochen. Wenn Sie hierzu Hilfe benötigen, so schauen Sie in den Kapiteln zur fortgeschrittenen Konfiguration der Firewall nach. Dort werden dann die einzelnen Protokolle betrachtet und Beispielregeln für deren Filterung gezeigt. Auch fortgeschrittene Fähigkeiten des Iptables-Befehls werden dort besprochen. Ich bin mir sicher, dass Sie dort auch Anhaltspunkte für Ihr Problem finden werden.

Teil IV

Werkzeuge

Dieser Teil stellt Ihnen verschiedene wertvolle Werkzeuge für die Administration, Pflege, Wartung, Fehlersuche und für den Test Ihrer Firewall-Struktur vor. Dabei handelt es sich um zentrale Protokollserver, Server zur Zeitsynchronisation, Werkzeuge zur Protokollanalyse, zur Administration der Regeln und zum Test der Firewall.

9. Zentrale Protokollserver

Sobald viele UNIX- oder Linux-Rechner überwacht werden müssen, stellen die meisten Administratoren fest, dass eine manuelle Überwachung der Protokolle auf jedem einzelnen Rechner sehr mühsam und fehlerträchtig ist. Daher werden häufig zentrale Protokollserver für die Sammlung der Daten eingesetzt.

Es gibt aber noch einen weiteren, unter Sicherheitsaspekten wichtigeren Grund für eine zentrale Protokollierung: Wenn tatsächlich ein Einbrecher in der Lage war, in Ihren Rechner einzudringen, so wird er zunächst versuchen, die Spuren zu entfernen, die der Einbruch hinterlassen hat. Dies betrifft häufig auch entsprechende Einträge in den Protokollen. Werden diese Protokolle lediglich lokal und unverschlüsselt gespeichert, so hat er ein leichtes Spiel.

Protokolliert jedoch der Rechner seine Meldungen nicht nur lokal, sondern über das Netzwerk auch zentral auf einem Protokollserver, so kann der Einbrecher zwar die Spuren in den lokalen Protokolldateien entfernen, nicht jedoch die Meldungen auf dem zentralen Protokollserver löschen. Hierzu müsste er erst in den Protokollserver einbrechen.

Für die Auswertung der Protokolle ist es sehr wichtig, dass die überwachten Rechner eine gemeinsame Zeitbasis verwenden. Ansonsten besteht die Gefahr, dass bei einem Angriff die Meldungen der unterschiedlichen Rechner nicht miteinander korreliert werden können. Die Meldung eines Angriffs durch einen Snort-Sensor und die Modifikation von Dateien und die Installation eines Rootkits auf einem anderen Rechner können nicht in einen kausalen Zusammenhang gebracht werden, wenn nicht die Zeiten der Meldungen bzw. Modifikationen vergleichbar sind. Achten Sie daher darauf, dass Ihre Systeme über eine Zeitsynchronisation verfügen!

9.1 Einrichtung eines zentralen Protokollservers

Die meisten Linux-Distributionen verwenden für die Systemprotokollierung eine Kombination aus zwei Diensten: Syslogd und Klogd. Syslogd ist der zentrale Protokolldienst unter Linux. Dieser Protokolldienst nimmt die Meldungen von verschiedenen anderen Systemen entgegen. Hierbei handelt es sich unter anderen um Crond, E-Mail, News, Login, Lpd etc. Der Syslogd ist typischerweise nicht zuständig für Dienste wie zum Beispiel einen Apache-Webserver, einen Squid-Proxy oder Samba. Der Klogd nimmt vom Kernel die Meldungen entgegen und leitet sie üblicherweise an den Syslogd weiter. Damit werden auch Meldungen des Linux-Kernels durch den Syslogd protokolliert.

Im Folgenden möchte ich kurz die wesentlichen Möglichkeiten der Konfiguration des Syslogd und des Klogd beschreiben. Anschließend werde ich zwei weitere Syslogd-Daemons beschreiben, die ersatzweise eingesetzt werden können. Der Rsyslogd ist der neue Default-Syslog-Daemon der Fedora- und Debian-Distribution. SuSE-Distributionen nutzen den Syslog-ng. Der Modular-Syslogd[1] wurde seit 2003 nicht mehr weiterentwicklt und wird daher hier nicht mehr berücksichtigt.

Sowohl der Rsyslogd als auch der Syslog-ng verfügen über wesentlich mehr Funktionen als der normale BSD-Syslogd.

Der BSD-Syslogd benötigt lediglich eine Konfigurationsdatei. Dies ist die Datei `/etc/syslog.conf`. Sie enthält Informationen darüber, welche Meldungen wo zu protokollieren sind. Hierzu wird die folgende Syntax verwendet:

```
facility.priority location
```

Der Standard-BSD-Syslogd, der in den meisten aktuellen Linux-Distributionen eingesetzt wird, ist in der Lage, die folgenden Facilities zu unterscheiden:

- `auth`
- `authpriv`
- `cron`
- `daemon`
- `kern`
- `lpr`
- `mail`
- `mark` (lediglich für die interne Verwendung)
- `news`
- `security` (identisch mit `auth`, sollte jedoch nicht mehr eingesetzt werden)
- `syslog`
- `user`
- `uucp`
- `local0` bis `local7`

Als `priority` werden die folgenden Werte unterstützt:

- `debug`
- `info`
- `notice`
- `warning` und `warn` (obsolet)
- `err`, `error` (obsolet)

1 `http://sourceforge.net/projects/msyslog`

» crit

» alert

» emerg, panic (obsolet)

Bei einer Angabe wie zum Beispiel cron.info werden nun alle Meldungen von cron mit der Wertigkeit info und höher von dieser Regel protokolliert. Möchten Sie spezifisch nur die Meldungen mit der Wertigkeit info protokollieren, so können Sie dies mit der Angabe cron.=info erreichen. Außerdem ist die Negation möglich: cron.!=info. Zusätzlich besteht die Möglichkeit, sowohl für die Facility als auch für die Priority den Stern (*) als Wildcard zu verwenden.

Leider besteht beim Standard-BSD-Syslog nicht die Möglichkeit, in Abhängigkeit von einem Muster, zum Beispiel einem regulären Ausdruck, die Protokollierung zu konfigurieren.

Nachdem Sie die zu protokollierenden Meldungen in der linken Spalte der Datei /etc/syslog.conf ausgewählt haben, müssen Sie den Ort der Protokollierung definieren. Hier stehen ebenfalls mehrere Möglichkeiten zur Verfügung:

» normale Datei, z.B. /var/log/messages

» Named Pipe, z.B. |/var/log/my_fifo. Diese Named Pipe muss zuvor mit dem Befehl mkfifo erzeugt werden. Hiermit besteht die Möglichkeit, die Informationen ähnlich wie über eine normale Pipe an einen anderen Prozess weiterzuleiten.

» Terminal bzw. Konsole, z.B. /dev/tty10

» zentraler Protokollserver, z.B. @logserver.example.com

» Benutzer, z.B. root. Diese Personen erhalten die Meldungen auf ihrem Terminal, falls sie angemeldet sind. Es wird keine E-Mail versandt!

» alle angemeldeten Benutzer *

Ein paar Beispiele verdeutlichen dies:

```
# Kritische Kernelmeldungen auf der 10. virtuellen Konsole
kern.crit                                                /dev/tty10

# Notfälle direkt an root
kern.emerg                                                     root

# Authentifizierungsvorgänge in einer geschützten Datei und zusätzlich
# zentral
authpriv.*              /var/log/secure authpriv.* @remotesyslog

# E-Mail getrennt
mail.*                                              /var/log/maillog

# Alles andere in einer Datei
*.info;mail.none;authpriv.none                    /var/log/messages
```

Der Syslog erhält üblicherweise die zu protokollierenden Meldungen über den UNIX-Socket `/dev/log`. Dieser Socket wird vom Syslog beim Start erzeugt. Dies genügt in den meisten Fällen, da sämtliche lokalen Anwendungen auf diesen Socket zugreifen können. Bei einer Anwendung, die sich in einer Chroot-Umgebung befindet, besteht jedoch das Problem, dass ein derartiger Zugriff auf `/dev/log` nicht möglich ist. Der Syslogd muss dann bei seinem Start einen zusätzlichen UNIX-Socket in der Chroot-Umgebung erzeugen. Dies kann mit der Option `-a socket` erfolgen. Hiermit können bis zu 19 weitere Sockets definiert werden.

Mithilfe dieser UNIX-Sockets ist jedoch der Syslogd nicht in der Lage, Meldungen über das Netzwerk entgegenzunehmen. Hierzu muss zusätzlich ein Internet-Socket geöffnet werden. Die Protokollierung über das Netzwerk erfolgt beim Standard-BSD-Syslogd an den UDP-Port 514. Die Konfiguration dieser Portnummer erfolgt beim Syslogd in der Datei `/etc/services`. Diese Datei enthält hierzu üblicherweise folgenden Eintrag:

```
syslog 514/udp
```

Leider erfolgt die Protokollierung nur mit UDP. Daher kann der Syslog nicht sicherstellen, dass sämtliche Meldungen tatsächlich den Empfänger erreichen. Einzelne Pakete können zum Beispiel aufgrund einer falsch konfigurierten Firewall oder einer Überlastung des Netzwerks verloren gehen. Auch bietet UDP keinen Schutz vor Spoofing.

Damit der Syslogd nun diesen Port öffnet und Meldungen über diesen Internet-Socket annimmt, müssen Sie ihn mit der Option `-r` starten. Dann nimmt er jede Meldung, die an diesen Socket gesendet wird, an und protokolliert sie entsprechend seiner eigenen Konfigurationsdatei `/etc/syslog.conf`. Es besteht leider nicht die Möglichkeit, im Syslogd die Clients, die die Meldungen senden, zu authentifizieren oder aufgrund ihrer IP-Adresse einzuschränken. Dies kann lediglich durch das Setzen intelligenter Paketfilterregeln (Iptables) geschehen. Ein Angreifer könnte daher den Protokollserver mit Meldungen (auch mit gefälschten Absenderadressen[2]) bombardieren und so die Protokolle mit unsinnigen Meldungen füllen.

Der Klogd besitzt keine eigene Konfigurationsdatei. Die Meldungen des Kernels werden vom Klogd an den Syslogd weitergeleitet, der sie aufgrund seiner Konfigurationsdatei protokolliert. Der Klogd verfügt jedoch über eine interessante Startoption: `-c X`. Diese Option erlaubt die Definition einer Priority X. Kernel-Meldungen, die mindestens diese Wertigkeit aufweisen, werden direkt vom Klogd auf der Konsole protokolliert. Dies ist jedoch den Geschwindigkeitsbeschränkungen der Konsole unterworfen. Die Konsole wird als serielles Terminal mit einer üblichen Geschwindigkeit von 38.400 Baud emuliert. Bei dem Überschreiten der Geschwindigkeit kann es zum Verlust von Meldungen kommen.

Die unterschiedlichen Distributionen verwalten die Startoptionen des Klogd und des Syslogd an unterschiedlichen Positionen. Fedora und OpenSUSE verwenden die Datei `/etc/sysconfig/syslog`.

Der Standard-BSD-Syslogd ist also bereits in der Lage, seine Meldungen über das Netzwerk auf einem zentralen Logserver zu protokollieren. Leider weist er weder eine Mustererken-

2 Dann hilft auch Iptables nicht mehr.

nung mit regulären Ausdrücken noch eine Authentifizierung oder Verschlüsselung auf. Daher handelt es sich hierbei nicht um einen idealen Protokolldienst für sicherheitsrelevante Umgebungen.

Wenn Sie eine Protokollmeldung aus einem Skript erzeugen wollen, können Sie einfach den Befehl `logger` verwenden. Dieser Befehl schreibt die Meldung über den Syslog-Socket `/dev/log` in die Protokolle.

9.2 Rsyslogd

Der Rsyslogd ist wahrscheinlich der mächtigste Open-Source-Syslog-Daemon, der aktuell verfügbar ist. Da die Entwicklung in sehr schnellen Schritten vorangeht, fast täglich neue Funktionen hinzukommen und alle paar Wochen neue Versionen veröffentlicht werden, möchte ich in diesem Artikel Ihnen nur die Funktionen des Daemons vorstellen und am Beispiel einer recht aktuellen Version eine Einführung in die Konfiguration geben.

Der Rsyslogd ist der Default-Syslog-Daemon der Distributionen Debian, Fedora und Red Hat. Ich kann seinen Einsatz uneingeschränkt empfehlen.

9.2.1 Rsyslogd-Funktionen

Die aktuellsten Versionen unterstützen unter anderem die folgenden Funktionen:

- native Unterstützung der Protokollierung in MySQL, PostgreSQL, Firebird/Interbase, OpenTDS (MS SQL, Sybase), SQLLite, Ingres, Oracle und mSQL via libdbi-Datenbanken
- native Unterstützung des Versands von E-Mail-Nachrichten
- Unterstützung von TCP als Transportprotokoll
- Unterstützung komprimierter Syslog-Meldungen
- Unterstützung eines Spool-Verzeichnisses auf der Festplatte zur Zwischenspeicherung von Nachrichten
- Überwachung von Textdateien und Konvertierung ihres Inhaltes in Syslog-Nachrichten
- Konfiguration von Backup-Servern (Syslog und Datenbank)
- Unterstützung des Empfangs von verlässlichen Nachrichten nach RFC3195
- automatische Erzeugung von Dateien und Verzeichnisstrukturen für die Protokollierung
- frei konfigurierbares Protokollformat einschließlich des Zeitstempels
- automatische Rotation von Protokolldateien
- Verschlüsselung der übertragenen Daten mit TLS
- granulare Filterfunktionen mit regulären Ausdrücken
- Unterstützung von IPv6
- Versand von SNMP-Trap-Nachrichten

9.2.2 Rsyslogd-Anwendung

Zunächst ist der Rsyslogd ein Drop-In-Replacement für das klassische Paar aus Klogd und Syslogd. Er kann auch mit deren Konfigurationsdatei arbeiten. Dann stehen aber viele Funktionen nicht zur Verfügung.

Die wichtigsten Erweiterungen werden bei dem Rsyslogd über Module zur Verfügung gestellt. Diese müssen zu Beginn der Konfigurationsdatei geladen werden. Die Namen der Module beginnen mit `om` für ein Output-Modul und `im` für ein Input-Modul. Während die Input-Module Meldungen aus den unterschiedlichsten Quellen entgegennehmen, können die Output-Module diese Meldungen an die verschiedensten Empfänger zustellen.

Aktuelle Versionen stellen die folgenden Module bereit:

- `omsnmp`
- `omgssapi`
- `ommysql`
- `omrelp`
- `ompgsql`
- `omlibdbi`
- `imfile`
- `imudp`
- `imtcp`
- `imrelp`
- `imgssapi`
- `immark`
- `imklog`
- `imuxsock`

Um Meldungen per UNIX-Socket für die lokale Protokollierung entgegenzunehmen, laden Sie das `imuxsock`-Modul. Mit dem Modul `immark` werden `--MARK--`-Meldungen in den Protokollen erzeugt. Das Modul `imklog` nimmt die Meldungen des Kernels entgegen. Um zusätzlich Meldungen per TCP entgegenzunehmen, laden Sie noch das Modul `imtcp`. Mit der Option `$InputTCPServerRun` setzen Sie den Port.

```
$ModLoad imuxsock.so
$ModLoad imklog.so
#$ModLoad immark.so

$ModLoad imtcp.so
$InputTCPServerRun 514
```

Um nun sämtliche nicht sensitiven Meldungen in einer Datei zu protokollieren, können Sie eine Syntax verwenden, die der von Syslog entspricht:

```
*.info;mail.none;authpriv.none;cron.none        /var/log/messages
authpriv.*                                        /var/log/secure
```

Gleichzeitig protokolliert diese Konfiguration sensitive Informationen in `/var/log/secure`.

Um mit regulären Ausdrücken weitere Meldungen getrennt zu speichern, können Sie die folgende Syntax verwenden:

```
:HOSTNAME, regex, 'station[0-9]+'                /var/log/stations
```

Hier wird der Hostname des Rechners mit dem angegebenen regulären Ausdruck verglichen. Stimmt er überein, so wird die Meldung in der angegebenen Datei gespeichert.

Dies soll Ihnen einen Überblick über die Möglichkeiten geben. Bitte lesen Sie die ausführliche Dokumentation Ihres Rsyslogd, wenn Sie diesen nutzen möchten.

9.3 Syslog-ng

Der Syslog-ng (`http://www.balabit.com/products/syslog_ng/`) ist eine weitere Alternative zu dem BSD-Syslog. Der Syslog-ng wird von der ungarischen Firma Balabit für ihr Firewall-Produkt Zorp entwickelt. Der Syslog-ng ist Open Source und wird unter der GPL-Lizenz vertrieben. Wenn Sie den Syslog-ng einsetzen möchten, müssen Sie sich von dem üblichen Format der Syslog-Konfiguration verabschieden. Der Syslog-ng verwendet ein eigenes, vollkommen anderes Format. Dafür gewinnen Sie aber auch einige sehr mächtige Vorteile gegenüber dem BSD-Syslog:

- bessere Filterfunktionen inklusive regulären Ausdrücken
- Transport der Meldungen per UDP oder TCP
- Protokollierung in einer MySQL- oder PostgreSQL-Datenbank
- Die Unterstützung langer Rechnernamen erlaubt eine einfachere Zuordnung der Meldungen.
- saubere und klare Konfigurationsdatei

Die Installation des Syslog-ng ist recht einfach, da es für fast jede Distribution bereits fertige binäre Pakete gibt.

Wenn Sie dennoch Syslog-ng aus seinen Quellen übersetzen möchten, können Sie das einfach mit den folgenden Befehlen erreichen:

```
$ ./configure
$ make
$ sudo make install
```

Um den Syslog-ng komfortabel über ein Init-Skript zu starten, können Sie eines der mitgelieferten Init-Skripten aus dem Verzeichnis `./contrib` verwenden.

Für die Übersetzung benötigen Sie zusätzlich das libol-Paket. Dieses Paket erhalten Sie auf `http://www.balabit.com/downloads/libol/0.3/`. Übersetzen und installieren Sie dieses Paket zuerst! Außerdem gibt es teilweise Probleme mit der Flex-Version. Ab Version 2.5.4 schlägt die Kompilierung teilweise fehl. Die Syslog-ng-FAQ gibt hier weitere Hinweise (`http://www.campin.net/syslog-ng/faq.html`).

Bei der Konfiguration des Syslog-ng-Quelltext-Pakets können Sie einige optionale Funktionen aktivieren. Hierbei handelt es sich sowohl um die Unterstützung für Sun Solaris-Betriebssysteme (`--enable-sun-streams` und `--enable-sun-door`) als auch um die Unterstützung für TCPwrapper (`--enable-tcp-wrapper`) und Source-Spoofing (`--enable-spoof-source`).

Nun müssen Sie noch den Syslog-ng konfigurieren. Der Syslog-ng verfolgt in seiner Konfigurationsdatei eine andere Philosophie als der BSD-Syslog und der modulare Syslog. Sie definieren zunächst die Quelle (Source) einer Meldung. Eine typische Quelle ist:

```
source src { unix-stream("/dev/log"); };
```

Anschließend definieren Sie einen oder mehrere Filter und dann die möglichen Ziele:

```
filter mail { facility("mail") and level ("info"); };
destination mail_file { file("/var/log/mail"); };
```

Wenn Sie Quellen, Filter und Ziele definiert haben, können Sie einen Protokollpfad (*log path*) definieren. Dieser Protokollpfad beschreibt, dass Meldungen aus einer bestimmten Quelle, auf die ein bestimmter Filter zutrifft, an einem bestimmten Ziel protokolliert werden sollen:

```
log { source(src);
      filter(mail);
      destination(mail_file); };
```

Da das Syslog-ng-Handbuch sehr ausführlich ist, werde ich mich hier nur auf die Beschreibung einiger spezieller Punkte beschränken. Zunächst möchte ich Ihnen eine typische Syslog-ng-Konfigurationsdatei für ein Linux-System vorstellen und erklären:

```
#####
# options

options {
         # disable the chained hostname format in logs
         # (default is enabled)
         chain_hostnames(no);

         # enable fully qualified hostnames
         # (default is disabled)
         use_fqdn(yes);
```


CE: Vielleicht wäre es hier günstig, die Kommentare zu übersetzen?

Editor's or typesetter's annotations (will be removed before the final TeX run)

```
# keep the original hostname
# (default disabled)
keep_hostname(yes);

# which time to log: original (default) or received
#use_time_recvd(no);

# the time to wait before a died connection is re-established
# (default is 60)
time_reopen(10);

# the time to wait before an idle destination file is closed
# (default is 60)
time_reap(360);

# the number of lines buffered before written to file you might
# want to increase this if your disk isn't catching with all the
# log messages you get or if you want less disk activity
# (say on a laptop)
# (default is 0)
#sync(0);

# the number of lines fitting in the output queue
log_fifo_size(2048);

# enable or disable directory creation for destination files
create_dirs(yes);

# default owner, group, and permissions for log files
# (defaults are 0, 0, 0600)
#owner(root);
#group(root);
#perm(0600);

# default owner, group, and permissions for created directories
# (defaults are 0, 0, 0700)
#dir_owner(root);
#dir_group(root);
dir_perm(0700);

# enable or disable DNS usage
# syslog-ng blocks on DNS queries, so enabling DNS may lead to a
# Denial of Service attack
```

```
        # (default is yes)
        use_dns(no);

        # maximum length of message in bytes
        # this is only limited by the program listening on the /dev/log
        # Unix socket, glibc can handle arbitrary length log messages,
        # but -- for example -- syslogd accepts only 1024 bytes
        # (default is 2048)
        #log_msg_size(2048); };

# sources

source s_local {
               # message generated by Syslog-NG
               internal();
               # standard Linux log source (this is the default place
               # for the syslog() function to send logs to)
               unix-stream("/dev/log");
               # messages from the kernel
               file("/proc/kmsg" log_prefix("kernel: ")); };

source s_external {
                  # tcp socket
                  tcp ( ip(0.0.0.0) port(5140) keep-alive(yes)); };

# destinations
# some standard log files
destination df_auth { file("/var/log/auth.log"); };
destination df_syslog { file("/var/log/syslog"); };
destination df_cron { file("/var/log/cron.log"); };
destination df_daemon { file("/var/log/daemon.log"); };
destination df_kern { file("/var/log/kern.log"); };
destination df_mail { file("/var/log/mail.log"); };
destination df_user { file("/var/log/user.log"); };

destination df_messages { file("/var/log/messages"); };

# consoles
# this will send messages to everyone logged in
destination du_all { usertty("*"); };

# all messages from external
destination d_external { file("/var/log/external"); };
```

```
#####
# filters

# all messages from the auth and authpriv facilities
filter f_auth { facility(auth, authpriv); };

# all messages except from the auth and authpriv facilities filter
f_syslog { not facility(auth, authpriv); };

# respectively: messages from the cron, daemon, kern, lpr, mail, news,
# user, and uucp facilities
filter f_cron { facility(cron); };
filter f_daemon { facility(daemon); };
filter f_kern { facility(kern); };
filter f_mail { facility(mail); };
filter f_user { facility(user); };

filter f_at_least_info { level(info..emerg); };

# all messages of info, notice, or warn priority not coming form the auth ↵
        ,
# authpriv, cron, daemon, mail, and news facilities
filter f_messages {
                   level(info,notice,warn)
                   and not facility(auth,authpriv,cron,daemon,mail,news) ↵
                           ;};

# messages with priority emerg
filter f_emerg { level(emerg); };

#####
# logs
# order matters if you use "flags(final);" to mark the end of processing
# in a "log" statement

log {
     source(s_local);
     filter(f_auth);
     destination(df_auth); };

log {
     source(s_local);
     filter(f_syslog);
```

```
        destination(df_syslog); };

log {
        source(s_local);
        filter(f_cron);
        destination(df_cron); };

log {
        source(s_local);
        filter(f_daemon);
        destination(df_daemon); };

log {
        source(s_local);
        filter(f_kern);
        destination(df_kern); };

log {
        source(s_local);
        filter(f_mail);
        filter(f_atleast_info);
        destination(df_mail); };

log {
        source(s_local);
        filter(f_user);
        destination(df_user); };

log {
        source(s_local);
        filter(f_messages);
        destination(df_messages); };

log {
        source(s_local);
        filter(f_emerg);
        destination(du_all); };

log {
        source(s_external);
        destination(d_external); };
```

Sie beginnen mit der Konfiguration der Optionen des Syslog-ng. Einige Optionen sind beispielhaft aufgeführt. In dem Syslog-ng-Handbuch finden Sie alle unterstützten Optionen. Auf

einem Syslog-ng-Logserver, der auch von außen Meldungen entgegennimmt, ist die Option `keep_hostname` interessant. Damit protokolliert der Syslog-ng den originalen Hostnamen des Rechners, der die Meldung gesendet hat, anstelle seines eigenen. Die Option `use_fqdn` protokolliert dann auch den vollqualifizierten Hostnamen, sodass die Rechner leichter auseinandergehalten werden können.

Anschließend definieren Sie die Protokollquellen. Hier wurden eine Quelle für lokale Meldungen (`s_local`) und eine Quelle für Meldungen definiert, die über TCP transportiert werden. Der Syslog-ng öffnet nun einen TCP-Socket auf dem Port 5140 und nimmt hier Protokollmeldungen von außen entgegen. Wenn Sie den Syslog-ng mit der Option `--enable-tcp-wrapper` übersetzt haben, können Sie in den Dateien `/etc/hosts.allow` und `/etc/hosts.deny` festlegen, welche Rechner Protokollmeldungen zustellen dürfen. Damit zum Beispiel nur das lokale Netzwerk 192.168.0.0/24 auf den Syslog-ng-Server zugreifen darf, verwenden Sie folgende Zeile in der Datei `/etc/hosts.deny`:

```
syslog-ng: ALL EXCEPT 192.168.0.0/255.255.255.0
```

Nach den Quellen definieren Sie die Filter und die Ziele, wo die Protokollmeldungen gespeichert werden sollen. Hier wurde nur eine kleine Menge von typischen Filtern und Zielen definiert. Sämtliche über das Netzwerk empfangenen Nachrichten werden komplett in einer eigenen Datei abgelegt. Die lokalen Meldungen werden wie üblich auf verschiedene Dateien aufgeteilt.

Der Syslog-ng kann auch eine Protokollierung direkt in einer Datenbank vornehmen. Hierzu unterstützt der Syslog-ng die MySQL- und die PostgreSQL-Datenbank. Damit Sie die Meldungen anschließend komfortabel betrachten können, existieren verschiedene in PHP geschriebene Oberflächen für die Verwaltung der Meldungen. Mein Favorit PHP-Syslog-NG ist leider in Logzilla-Pro aufgegangen. Dieses Werkzeug ist jetzt nur noch unter einer kommerziellen Lizenz verfügbar. Dennoch können Sie eine der letzten Open-Source-Varianten von 2006 einsetzen. Alternativ können Sie auch die kommerzielle Variante Logzilla-Pro (`http://www.logzilla.pro`) testen. Die letzte Open-Source-Variante erhalten Sie unter `http://sourceforge.net/projects/php-syslog-ng`.TS9

Um diese Funktion zu nutzen, müssen Sie zunächst den Syslog-ng so konfigurieren, dass er seine Meldungen in die Datenbank schreibt. Hierzu benötigen Sie zunächst ein Ziel (Destination), das die Meldungen in die Datenbank schreibt:

```
# pipe messages to /var/log/mysql.pipe to be processed by mysql
destination d_mysql {
    pipe("/var/log/mysql.pipe"
    template("INSERT INTO
    logs (host, facility, priority, level, tag, datetime, program, msg)
    VALUES ( '$HOST', '$FACILITY', '$PRIORITY', '$LEVEL', '$TAG', '$YEAR-
            $MONTH-$DAY $HOUR:$MIN:$SEC','$PROGRAM', '$MSG' );\n")
    template-escape(yes));
};
```


TS9 Bitte auf Bild 9.1 im Text referenzieren.

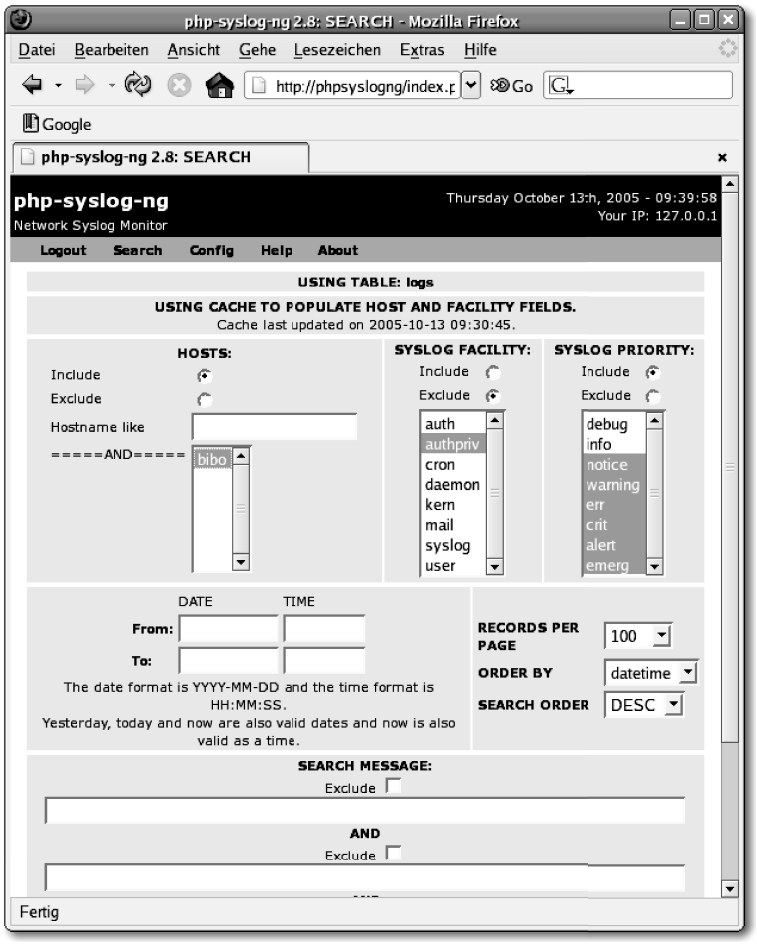


Abbildung 9.1: **Mit dem PHP-Frontend php-syslog-ng können Sie komfortabel nach Meldungen suchen und sie betrachten.**

INFO

Bevor Sie diese Zeilen abtippen, möchte ich Sie darauf hinweisen, dass sich in dem Unterverzeichnis `./scripts` in dem Paket `phpsyslogng` eine Beispielkonfiguration befindet, und zwar in der Datei `./scripts/syslog.conf`.

Dieses Ziel schreibt nun die Meldungen in eine Datei namens `/var/log/mysql.pipe`. Dabei handelt es sich jedoch nicht um eine normale Datei, sondern um eine Pipe. Diese müssen Sie manuell anlegen:

```
$ sudo mkfifo /var/log/mysql.pipe
```

Eine derartige Named-Pipe ähnelt der Pipe (|) auf der Kommandozeile. Ein Prozess schreibt in die Pipe, während ein anderer Prozess die Daten in derselben Reihenfolge ausliest. Da die Daten bereits in der SQL-Sprache in die Pipe geschrieben werden, benötigen Sie jetzt nur noch ein Programm, das die Daten aus der Pipe ausliest und in die Datenbank schreibt. Das

Abbildung 9.1: **(Fortsetzung)**

Datenbank-Management-System (DBMS) MySQL selbst kann dies nicht. Hierfür genügt aber das folgende einfache Skript `syslog2mysql.sh`:

```
#!/bin/bash
if [ ! -e /var/log/mysql.pipe ]
then
     mkfifo /var/log/mysql.pipe
fi
while [ -e /var/log/mysql.pipe ]
do
     mysql -u syslogfeeder --password=PW_HERE syslog < /var/log/mysql. ↵
              pipe >/dev/null
done
```


ceh Was tut man, wenn man keinen Vixie-Crond hat?

Editor's or typesetter's annotations (will be removed before the final TeX run)

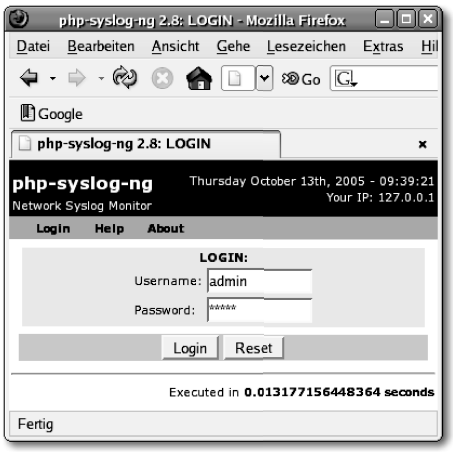


Abbildung 9.2: **Bei dem ersten Login melden Sie sich mit admin/admin an.**

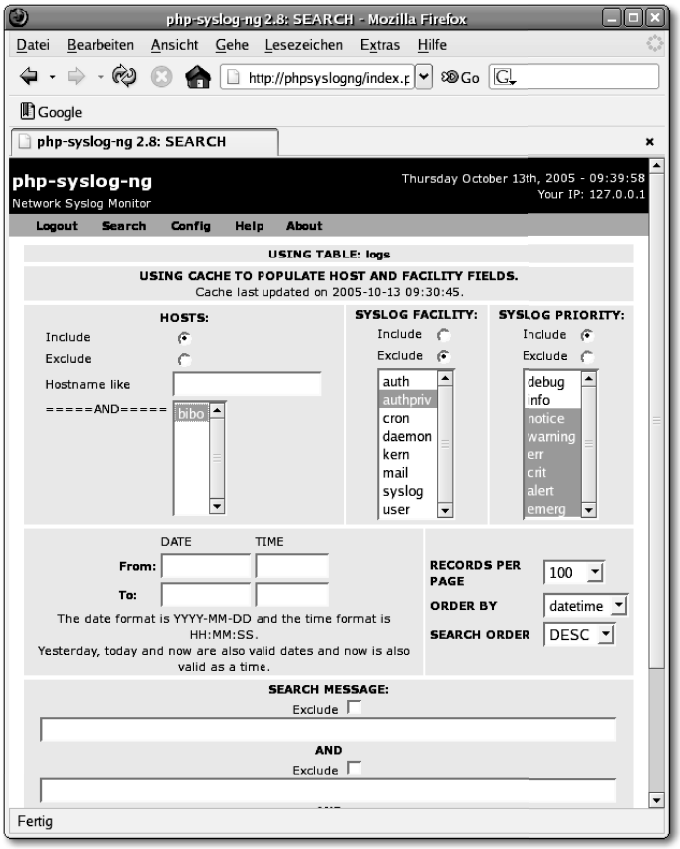


Abbildung 9.3: **Eine mächtige Suchmaske erlaubt Ihnen das Eingrenzen der gewünschten Meldungen.**

Dieses Skript wird ebenfalls mit `phpsyslogng` als `./scripts/syslog2mysql.sh` mitgeliefert. Damit dieses Skript jedes Mal nach einem Neustart automatisch aufgerufen wird, kann es sehr einfach über den Cron-Daemon eingebunden werden. Vielleicht verwenden Sie ja den Vixie-Crond auf Ihrem Linux-System. Dieser besitzt seit über 10 Jahren die Möglichkeit, jedes Mal nach einem Reboot des Systems einen Befehl zu starten. Tragen Sie einfach folgende Zeile in der Datei `/etc/crontab` ein:

```
@reboot root /usr/local/bin/syslog2mysql.sh >> /var/log/syslog2mysql 2>&1
```

Jetzt müssen Sie noch die Datenbank anlegen. Starten Sie Ihr MySQL-DBMS. Für die Erzeugung der Datenbank enthält die aktuelle Version des `phpsyslogng`-Pakets in dem Unterverzeichnis `./scripts` eine Datei names `scripts/dbsetup.sql`. Sie müssen diese Datei nur in Bezug auf die zu verwendenen Kennwörter anpassen und können dann das Skript aufrufen:

```
$ mysql -u root -p < scripts/dbsetup.sql
```

Abbildung 9.4: **Die Ergebnisse werden in Farbe dargestellt.**

Nun müssen Sie nur noch im Syslog-ng die Meldungen auswählen, die in der Datenbank protokolliert werden sollen. Hierzu fügen Sie einen Protokollpfad in der Konfigurationsdatei des Syslog-ng hinzu. Am einfachsten ist der folgende Protokollpfad ohne Filter:

```
log { source(src); destination(d_mysql); };
```

Damit Sie später auch der Menge der Protokolleinträge in der Datenbank Herr werden, hat Claus Lund, der Entwickler von `phpsyslogng` ein Rotationsskript namens `./scripts/logrotate.php` geschrieben, das täglich die Log-Tabelle in der Datenbank umbenennt und eine neue Tabelle beginnt. Dieses Skript müssen Sie in Bezug auf den Installationsort von `phpsyslogng` anpassen:

```
$APP_ROOT = '/var/www/phpsyslogng';
```

Zusätzlich empfehle ich auch noch folgendes Skript, das ein Backup der Datenbank durchführt:

```
#!/bin/bash
# make sure you have directory /var/log/backup created with mask 600
/usr/bin/mysqlhotcopy --user=syslogadmin --password=g3h31m --allowold ↵
        syslog
/var/log/backup
```

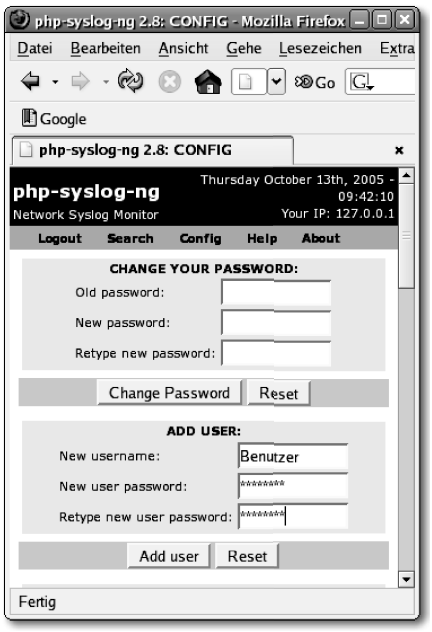


Abbildung 9.5: **Für die tägliche Arbeit können Sie auch zusätzliche Benutzer anlegen.**

Diese Skripten können Sie ebenfalls über den Cron-Daemon aufrufen. Fügen Sie dazu folgende Zeilen Ihrer Cron-Tabelle `/etc/crontab` hinzu:

```
@daily root /usr/local/bin/mysql-backup.sh >> /var/log/syslog2mysql 2>&1
```

Alternativ können Sie diese Zeile natürlich auch in einem Skript in dem Verzeichnis `/etc/cron.daily` speichern.

Da sämtliche Skripten das Kennwort für den Zugriff auf die Datenbank enthalten, sollten Sie die Rechte so setzen, dass nur root diese Skripten lesen und verwenden darf!

Die restliche Konfiguration von `phpsyslogng` ist recht einfach. Beginnen Sie damit, dass Sie den `phpsyslogng`-Quelltextbaum in das DocumentRoot-Verzeichnis Ihres Webservers kopieren. Vielleicht legen Sie auch ein neues Verzeichnis an und erzeugen in dem Apache einen VirtualHost für dieses Verzeichnis. Anschließend müssen Sie lediglich die Datei `./config/config.php` anpassen. Hier sollten Sie darauf achten, dass Sie sämtliche Kennwörter wieder richtig anpassen. Haben Sie alles richtig gemacht, können Sie sich bereits mit einem Browser verbinden und sehen die Login-Seite von `phpsyslogng` (siehe Abbildung 9.2).

Bei dem Login können Sie sich mit `admin/admin` anmelden. Anschließend können Sie in der Datenbank suchen (siehe Abbildung 9.3), die Ergebnisse betrachten (siehe Abbildung 9.4) und weitere Benutzer anlegen (siehe Abbildung 9.5).

10. Zentrale Zeitsynchronisation

Wenn Sie Ihre Firewall-Protokolle später auswerten möchten und vielleicht auch noch mit Protokollen Ihres Webservers und Ihres Intrusion-Detection-Systems korrelieren möchten, ist es wichtig, dass die Protokolle über eine einheitliche Zeitbasis verfügen. Um dies zu erreichen, führt kein Weg an einem zentralen Zeitserver vorbei, der die Systeme mit der korrekten Uhrzeit versorgt. Dieses Kapitel zeigt Ihnen, wie Sie einen Zeitserver aufsetzen und Ihre Systeme damit synchronisieren.

10.1 Das Zeitsynchronisationsprotokoll NTP

Viele Systemadministratoren verschwenden keinen Gedanken an zentrale Zeitsynchronisation. Dabei ist die zentrale Zeitsynchronisation sämtlicher Netzwerkkomponenten im Falle eines sicherheitsrelevanten Ereignisses (Incident) von größter Bedeutung. Wie wollen Sie die Protokolle auswerten, wenn die Zeiten in den Protokollen des angegriffenen Webservers, der Datenbank, der Firewall und den Intrusion-Detection-Systems nicht zusammenpassen? Wollen Sie raten, was zuerst passiert ist? Auch die normale Auswertung von Protokollen ist bei nicht synchroner Zeit schwierig. Wie möchten Sie die Laufzeit einer E-Mail über zwei oder drei Mail-Relays ermitteln, wenn alle Systeme eine unterschiedliche Zeitbasis verwenden?

Um dieses Problem zu lösen, ist bereits vor langer Zeit das Network Time Protocol (NTP) geschaffen worden. Aktuell ist die Version 4 des Protokolls, das mit der Version 3 (RFC 1305) und den Versionen 1 (RFC1059) und 2 (RFC1119) kompatibel ist. Die wesentliche Neuerung des Protokolls in der Version 4 ist die Unterstützung von asymmetrischer Kryptografie und IPv6. Die Version 4 ist bisher nicht in einem RFC beschrieben worden. Es existiert lediglich ein Draft[1], der das Protokoll beschreibt. Alternativ zu dem NTP-Protokoll existiert auch das Simple Network Time Protocol (SNTP), das dem NTP-Protokoll ähnlich ist, aber weniger Funktionen besitzt und einfacher zu implementieren ist.

Das NTP-Protokoll verwendet zum Transport der Informationen den UDP-Port 123. Ungewöhnlicherweise verwendet das Protokoll diesen Port sowohl als Client als auch als Server. Die Zeitinformationen können per Unicast, Multicast und Broadcast verteilt werden. Leider bietet das UDP-Protokoll keinerlei Schutz vor Spoofing oder Modifikation der transportierten Informationen. Hierfür kann ab der Version 4 die asymmetrische Kryptografie zum Schutz der Integrität und Authentizität eingesetzt werden.

1 http://tools.ietf.org/wg/ntp/draft-ietf-ntp-ntpv4-proto/draft-ietf-ntp-ntpv4-proto-00.txt

Das NTP-Protokoll wird bei dem Zugriff eines NTP-Clients auf einen NTP-Server eingesetzt. Als NTP-Server kann jedes System eingesetzt werden, das über eine genaue Zeit verfügt. Die Zeitquelle kann eine DCF-77-Uhr sein, wie zum Beispiel die Expert Mouseclock von GUDE ANALOG- und DIGITALSYSTEME GmbH (`http://www.gude.info`). Alternativ können Sie auch eine GPS-Maus nutzen. Achten Sie nur darauf, dass das Produkt über Linux-Unterstützung verfügt. Natürlich können Sie auch einen oder mehrere andere NTP-Server als Zeitquelle nutzen. Eine Liste von öffentlich frei verfügbaren NTP-Servern finden Sie auf `http://www.pool.ntp.org/` und `http://support.ntp.org/bin/view/Servers/WebHome`. Die Server werden in Stratum 1,2,3...-Server eingeteilt. Ein Stratum-1-Server ist direkt mit einer Hochpräzisionszeitquelle (z.B. Atomuhr) verbunden. Ein Stratum-3-Server synchronisiert sich mit einem Stratum-2-Server, der sich wieder mit einem Stratum-1-Server synchronisiert. Je weiter weg sich ein Server von einem Stratum-1-Server befindet, desto ungenauer ist seine Zeit. Wenn Sie selbst einen Server mit DCF-77-Uhr aufsetzen, ist dies ein Stratum-1-Server. Alle Systeme, die diesen Rechner zur Synchronisation verwenden, sind Stratum-2-Systeme.

Wenn Sie für die Synchronisation Ihres NTP-Servers weitere NTP-Server im Internet verwenden, achten Sie bitte darauf, dass Sie mehrere global verteilte Systeme verwenden. Ansonsten besteht die Gefahr, dass beim Ausfall eines Systems oder bei einer falschen Uhrzeit auf diesem System Ihre Zeitsynchronisation in Mitleidenschaft gezogen wird. Alle Systeme können mehrere Zeitserver zur Synchronisation einsetzen!

10.2 Der ntpd-Zeitserver

Die am häufigsten eingesetzte Software für den Aufbau eines Zeitservers auf der Basis von Linux ist der `ntpd` (früher `xntpd`). Er ist in den meisten Distributionen enthalten. Der Quelltext wird unter `http://support.ntp.org/bin/view/Main/SoftwareDownloads` vorgehalten. Eine weitere Möglichkeit für den Aufbau eines NTP-Servers ist die Software `openntp` von dem OpenBSD-Projekt (`http://www.openntpd.org`). Dieser Zeitserver ist wesentlich einfacher in seinen Funktionen und in vielen Konfigurationseinstellungen kompatibel. Er unterstützt bisher jedoch nicht NTP Version 4 und damit auch keine Authentifizierung.

Da der `ntpd` in allen aktuellen Distributionen enthalten ist, spare ich mir hier die Beschreibung der Installation aus den Quellen. Sie finden eine Installationsanleitung in dem Quellpaket, falls Sie tatsächlich das Paket manuell installieren möchten. Ansonsten stellen Sie bitte sicher, dass das `ntp`-Paket Ihrer Distribution installiert ist. Zunächst betrachten wir die Konfiguration des NTP-Servers als Client und anschließend als Server.

10.2.1 Der Client

Das `ntp`-Paket enthält zwei Möglichkeiten, um ein System als Client mit einem Zeitserver zu synchronisieren. Der Kommandozeilenbefehl `ntpdate` führt eine einmalige Synchronisierung durch. Der Server `ntpd` kann auch als Client eine ständige Synchronisierung des Clients mit einem NTP-Server ermöglichen. Sinnvoll ist daher der Einsatz des `ntpd` als Client, um die ständige Synchronisierung zu gewährleisten. Der `ntpdate`-Befehl wird auch in zukünftigen

Versionen der Software entfernt werden, da er bereits jetzt durch den Aufruf `ntpd -q` ersetzt werden kann.

Wenn die Zeit des Clients um mehr als 1000 Sekunden von der Zeit des Servers abweicht, weigert sich der `ntpd`, die Zeit zu synchronisieren. Daher wird häufig der Befehl `ntpdate` vor dem Start des `ntpd` aufgerufen. Wenn Sie diesen Befehl durch `ntpd -q` ersetzen möchten, müssen Sie hier zusätzlich `-g` angeben. Diese Option schaltet die 1000-Sekunden-Prüfung ab.

Die Konfiguration des `ntpd` als Client ist sehr einfach. Erzeugen Sie lediglich die folgende Konfigurationsdatei `/etc/ntp.conf`:

```
# Erlaube per Default niemandem die Modifikation, die Abfrage oder das
# Monitoring des Zeitservers
restrict default ignore

# Erlaube alle Funktionen über das Loopback-Interface
restrict 127.0.0.1

# Lokale Drift-Datei, muss schreibbar sein.
driftfile /var/lib/ntp/drift

# Expert Mouseclock
# Generiere Link: ln -s /dev/ttyS0 /dev/refclock-0
# server 127.127.8.0 mode 5

# Server im Internet
server 0.pool.ntp.org
server 1.pool.ntp.org
server 2.pool.ntp.org

# Lokale Uhr
server 127.127.1.0          # local clock
fudge 127.127.1.0 stratum 10
```

Ausgestattet mit dieser Konfigurationsdatei, können Sie Ihren NTP-Server bereits probehalber starten. Natürlich sollten Sie die eingetragenen Server im Internet prüfen und bei Bedarf durch andere ersetzen. Um den NTP-Server probehalber zu starten, geben Sie auf der Kommandozeile zunächst den Befehl `ntpd -q -g` ein. Dies führt eine einmalige Synchronisation durch. Dieser Vorgang kann einige Sekunden dauern:

```
# ntpd -q -g
ntpd: time set -6.640906s
```

Anschließend geben Sie das Kommando `ntpd` auf der Kommandozeile ein. Es sollte sich scheinbar sofort beenden. Prüfen Sie, ob der `Ntpd` richtig im Hintergrund läuft, indem Sie sich die laufenden Prozesse anzeigen lassen.

Sie können die Funktion des `Ntpd` nun mit dem Befehl `ntpdc` auf dem lokalen System überprüfen.

```
# ntpdc
ntpdc> peers
remote           local            st  poll reach delay   offset   disp
==============================================================================
216.154.195.60   192.168.255.100  3   64   3     0.15869 0.003562 1.98438
LOCAL(0)         127.0.0.1        10  64   3     0.00000 0.000000 1.98436
gatekeeper.no-s  192.168.255.100  1   64   3     0.23141 0.000760 1.98444
frigg.interstro  192.168.255.100  3   64   3     0.07040 0.001728 1.98438
ntpdc>
```

Der Befehl `ntpdc` erlaubt die komplette Administration des `Ntpd`-Servers. Sie können Server hinzufügen und entfernen, den aktuellen Zustand betrachten, Rechte ändern etc. Die Manpage gibt Ihnen weitere Auskünfte. Der Befehl `peers` ist der häufigste Befehl, den Sie wahrscheinlich brauchen. Sie können diesen Befehl auch direkt mit `ntpdc -p` aufrufen. Dieser Befehl zeigt Ihnen die aktuellen Peers des Servers und ihre Zustände an. Die erste Spalte gibt Ihnen Auskunft über den Zustand des Peers:

- +: Symmetrisch aktiv. Der Rechner sendet regelmäßig Nachrichten an die Adresse und zeigt seine Synchronisationsfähigkeit.
- -: Symmetrisch passiv. Der Rechner empfängt symmetrisch aktive Nachrichten.
- =: Der Peer wird im Clientmode abgefragt.
- ^: Broadcasts werden an die Adresse versandt.
- : Broadcasts werden von dieser Adresse empfangen.
- : Mit diesem Server erfolgt aktuell die Synchronisation.

Sobald Sie ein * in der ersten Spalte erkennen können, erfolgt eine aktive Zeitsynchronisation mit dem entsprechenden System.

```
# ntpdc -p
remote           local            st  poll reach delay   offset   disp
==============================================================================
*216.154.195.60  192.168.255.100  3   64   37    0.15869 0.003562 0.66310
LOCAL(0)         127.0.0.1        10  64   37    0.00000 0.000000 0.66202
gatekeeper.no-s  192.168.255.100  1   64   37    0.23141 0.000760 0.66327
frigg.interstro  192.168.255.100  3   64   37    0.07040 0.001728 0.66351
```

Sobald der NTP-Server als Client zu Ihrer Zufriedenheit läuft, können Sie ihn mit `kill` beenden und über die Startskripten Ihrer Distribution starten. Prüfen Sie bitte, ob der Server

anschließend auch tatsächlich läuft. Viele Distributionen verwenden einen eigenen Benutzer für den Betrieb des NTP-Servers und starten ihn in einem Chroot (siehe Abschnitt 10.3). Dabei kann es zu Rechteproblemen kommen. Prüfen Sie die Protokolle Ihrer Distribution, um mögliche Fehlermeldungen zu finden.

10.2.2 Der Server

Die bisher erstellte Konfigurationsdatei erlaubt den Betrieb eines NTP-Servers als Client. Dieser Client wird sich nun ständig mit den verfügbaren Zeitquellen synchronisieren. Häufig möchten Sie aber auch einen eigenen Zeitserver betreiben, sodass sich weitere Clients mit diesem Zeitserver synchronisieren können.

Die bisherige Konfiguration erlaubt es keinem Client, Synchronisationsanfragen an diesen Server zu schicken, und der Server versendet auch keine Broadcast-Pakete. Die Verwendung des Broadcast- oder Multicast-Transports kann ich auch nur empfehlen, wenn Sie gleichzeitig eine Authentifizierung aktivieren (siehe Abschnitt 10.3). Die Gefahr eines gespooften NTP-Angriffs ist ansonsten zu hoch.

Um einen Broadcast-Server zu konfigurieren, müssen Sie die Konfigurationsdatei nur um eine Zeile ergänzen:

```
broadcast 192.168.0.255
```

Wenn ein Client diesen Broadcast-Server nutzen soll, tragen Sie auf dem Client die folgenden beiden Zeilen ein:

```
broadcastclient
broadcastdelay 0.008
```

Um Unicast-Clients zu unterstützen, fügen Sie eine zusätzliche Zeile mit dem `restrict`-Parameter in der Konfigurationsdatei hinzu. Achten Sie darauf, dass Sie bei der Angabe den Clients nur die Abfrage der Synchronisationsinformationen erlauben.

```
restrict 192.168.0.0 mask 255.255.255.0 nomodify notrap
```

Nun dürfen Clients in dem Netzwerk 192.168.0.0/24 diesen NTP-Server als Zeitserver zur Synchronisation nutzen.

10.3 Sicherheit

In einem Firewall-Buch muss bei dem Einsatz von NTP auch über die Sicherheit des Protokolls, der Server und des Clients gesprochen werden. Dabei müssen zwei Aspekte betrachtet werden:

» Der NTP-Server verwendet den Port 123/udp. Dies ist ein privilegierter Port (<1024), und er steht daher nur dem Benutzer root zur Verfügung. Der NTP-Server muss daher mit root-Rechten gestartet werden!

» Das NTP-Protokoll nutzt das UDP-Protokoll für den Transport der Informationen. Dieses Protokoll bietet keinen Schutz der Integrität und Authentizität. Ein Spoofing-Angriff ist leicht möglich.

Um die Sicherheit des Servers auf Port 123 zu erhöhen, ist der `ntpd` in der Lage, nach seinem Start die root-Privilegien abzugeben und in ein Chroot-Verzeichnis zu wechseln. Hiermit reduzieren Sie mögliche Gefahren bei einem Angriff auf den Dienst enorm. Damit der `ntpd` nach dem Start die root-Privilegien abgibt, müssen Sie bei dem Start mit der Option `-u ntp:ntp` einen unprivilegierten Benutzer und eine unprivilegierte Gruppe übergeben. Sobald sich der NTP-Server an den Port 123/udp gebunden hat, wird er seinen Benutzerkontext entsprechend ändern.

Damit der NTP-Server in einem Chroot-Verzeichnis arbeitet, müssen Sie das Verzeichnis vorbereiten. Hierzu müssen Sie alle Dateien, die der `ntpd` für seine Funktion benötigt, in das Verzeichnis kopieren. Hierbei handelt es sich mindestens um die Datei `ntp.conf` und die Drift-Datei. In Abhängigkeit der Konfiguration sind auch noch die Verzeichnisse `/var/run`, `/var/log` und das `/dev`-Verzeichnis mit lokalen Zeitquellen erforderlich.

Anschließend starten Sie den NTP-Server mit der Option `-T /var/chroot-dir`. Diese Funktion erhöht die Sicherheit nur, wenn der Prozess auch die root-Privilegien abgibt!

Zum Schutz vor direkten Angriffen auf das Protokoll wie Spoofing unterstützt der NTP-Server in der Version 4 des NTP-Protokolls kryptografische Methoden. Um diese zu nutzen, müssen Sie zunächst Schlüssel erzeugen.

Am einfachsten ist die Authentifizierung mit symmetrischen Schlüsseln. Daher soll diese Authentifizierung hier zuerst besprochen werden.

10.3.1 Symmetrische Authentifizierung

Bei der symmetrischen Authentifizierung verfügen beide Authentifizierungspartner über identische Schlüssel. Dies ist sowohl ein Vorteil, denn die Verteilung ist sehr einfach, aber auch gleichzeitig ein Nachteil, da der Schlüssel vertraulich transportiert und gespeichert werden muss und der Austausch mit Dritten problematisch sein kann.

Es gibt vier verschiedene Arten von Schlüsseln:

1. `A`: einen ASCII-Schlüssel aus maximal 8 Zeichen
2. `M`: einen ASCII-Schlüssel mit maximal 31 Zeichen
3. `S`: einen 64-Bit-Wert mit dem niedrigstwertigsten Bit pro Byte als Parität (DES)
4. `N`: einen 64-Bit-Wert mit dem höchstwertigsten Bit pro Byte als Parität

Die Schlüssel `A` und `M` sind am einfachsten zu handhaben. Diese Schlüssel müssen in der Datei `/etc/ntp/keys` abgespeichert werden:

```
1 A ABCDEFGH
2 M qwertzuiopasdfghjklyxcvbnmqwer
```

Jeder Schlüssel in dieser Datei erhält eine Nummer und einen Typ. Über die Nummer des Schlüssels werden diese nun in der Konfigurationsdatei referenziert. Damit ein Client einen Schlüssel für die Verbindung zu einem Server nutzt, muss er zunächst dem Schlüssel vertrauen (*trust*) und wissen, welchen Schlüssel er für welchen Server verwenden soll:

```
trustedkey 1 2
server 192.168.0.5 key 1
server 192.168.0.7 key 2
```

Der Server benötigt identische Schlüssel und muss ebenfalls den Schlüsseln vertrauen. Wenn Sie die Zeit per Broadcast verteilen möchten, müssen Sie auf dem Server den Schlüssel in der `broadcast`-Zeile definieren und ebenfalls auf dem Client und dem Server den Schlüsseln vertrauen:

```
trustedkey 1 2
broadcast 192.168.0.255 key 1
```

Auf dem Client genügen dann die folgenden Zeilen:

```
broadcastclient
trustedkey 1 2
keys /etc/ntp/keys
```

10.3.2 Asymmetrische Authentifizierung

Die asymmetrische Authentifizierung ist in vielen Umgebungen die bessere Lösung. Sie müssen nun nur die öffentlichen Schlüssel (*Public Keys*) sämtlicher Systeme untereinander austauschen. So ist es auch einfach möglich, zu dritten Instanzen sichere Verbindungen aufzubauen.

Die asymmetrische Authentifizierung wird bei dem `ntpd` als Autokey bezeichnet. Bei der Konfiguration wird zwischen Broadcast- und Multicast-Autokey und Unicast-Autokey unterschieden. Während das Unicast-Autokey-Verfahren auf dem Client konfiguriert wird, wird das Broadcast- und Multicast-Autokey-Verfahren auf dem Server konfiguriert.

Für das Broadcast-Autokey-Verfahren tragen Sie auf dem Server folgende Zeile ein:

```
broadcast 192.168.0.255 autokey
```

Bei dem Unicast-Autokey-Verfahren tragen Sie auf dem Client folgende Zeile ein:

```
server 192.168.0.5 autokey
```

Zusätzlich benötigen beide Systeme noch die folgenden Zeilen:[2]

```
crypto pw (client|server)password
keysdir /etc/ntp
```

2 Wählen Sie `client` oder `server` entsprechend der Funktion.

Dann müssen Sie noch die Schlüssel erzeugen und verteilen. Autokey unterstützt drei verschiedene Identitätsschemata: IFF, GQ und MV.[3]

» **IFF**. Bei dem IFF-Identitätsschema werden für jeden Server spezifische Schlüssel erzeugt. Sie müssen diesen Schlüssel für jeden Client exportieren. Dabei kann der Schlüssel wieder mit einem Client-Passwort verschlüsselt transportiert werden.

  Um die IFF-Parameter zu erzeugen, verwenden Sie:

```
cd /etc/ntp
ntp-keygen -T -I -p serverpassword
```

» **GQ**. Bei dem GQ-Identitätsschema wird ein Schlüssel erzeugt, der von allen Systemen in der Gruppe genutzt wird. Die Erzeugung erfolgt mit:

```
cd /etc/ntp
ntp-keygen -T -G -p serverpassword
```

» **MV**. Bei dem MV-Schema erzeugen Sie einen Schlüssel für den Server und N-1 Schlüssel für die Clients.

```
cd /etc/ntp
ntp-keygen -V N -p serverpassword.
```

Für die Erzeugung der Parameter werden ein Server Key und ein Zertifikat erzeugt. Diese sind nur für ein Jahr gültig und müssen anschließend neu erzeugt werden. Dies erfolgt mit `cd /etc/ntp; ntp-keygen -T -q serverpassword`.

Auf dem Client müssen Sie zunächst auch einen Schlüssel und ein Zertifikat erzeugen. Anschließend müssen Sie die entsprechenden Schlüssel der Server nun auch auf dem Client importieren und installieren. Um den Schlüssel und das Zertifikat für den Client zu erzeugen, verwenden Sie:

```
cd /etc/ntp
ntp-keygen -H -p clientpassword
```

» **IFF**. Bei den IFF-Gruppenschlüsseln müssen Sie diese auf den Servern zunächst exportieren. Dazu verwenden Sie den folgenden Befehl:

```
ntp-keygen -e -q serverpassword -p clientpassword
```

  Diesen so mit dem Client-Passwort verschlüsselten Schlüssel können Sie nun per E-Mail verschicken oder per Diskette auf den Client transportieren. Speichern Sie den Schlüssel auf dem Client in dem Verzeichnis `/etc/ntp` ab, und erzeugen Sie einen symbolischen Link:

```
cd /etc/ntp
ln -s ntpkey_IFFkey_server.3301264563 ntpkey_iff_server
```

3 Neue Versionen unterstützen auch PC.

» **GQ**. Kopieren Sie den GQ-Gruppen-Schlüssel sicher auf den Client, und erzeugen Sie ebenfalls eine Verknüpfung.

```
cd /etc/ntp
ln -s ntpkey_GQpar_server.3301145293 ntpkey_gq_server
```

» **MV**. Kopieren Sie den auf dem Server erzeugten Client-MV-Schlüssel auf den Client, und erzeugen Sie ebenfalls die Verknüpfung.

```
cd /etc/ntp
ln -s ntpkey_MVkey1_server.3301144193 ntpkey_mv_server
```

Nach einem Neustart der `ntpd`-Daemons auf den Servern und Clients sollten Sie mit dem Kommando `ntpq -p` den Zustand der Synchronisation beobachten.

Mit `ntpq -c` erhalten Sie Informationen über die Authentifizierung:

```
# ntpq -c
nd assID status conf reach auth condition last_event cnt
===========================================================
1 26132  f694   yes  yes  ok    sys.peer  reachable  9
```

11. Protokollanalyse

Wenn Sie eine Firewall betreiben, wollen Sie auch wissen, was passiert. Die Protokollanalyse ist eine der wichtigsten Aufgaben des Firewall-Administrators. Daher sollten Sie sich nur mit den besten Werkzeugen zufriedengeben. Leider ist die eierlegende Wollmilchsau in diesem Bereich noch nicht erfunden oder programmiert worden. Dieses Kapitel zeigt Ihnen die Werkzeuge, die ich bei meiner Arbeit schätzen gelernt habe, und einige weitere, die ich sehr interessant finde.

11.1 Fwlogwatch

Fwlogwatch ist ein Echtzeit-Protokoll-Analysator speziell für Firewall-Protokolle. Fwlogwatch wird von Boris Wesslowski unter der GPL-Lizenz vertrieben (`http://fwlogwatch.inside-security.de/`). Firewall-Protokolle weisen häufig eine sehr spezielle Syntax auf. Auch wenn Sie ein Linux-Guru sind, gestaltet sich die Erstellung von Zusammenfassungen und speziellen Alarmmeldungen mit Linux-Hausmitteln wie `grep` und `awk` unter Umständen recht kompliziert. Mit Fwlogwatch sind Sie in der Lage, verschiedenste Protokollformate zu analysieren: Linux ipchains, Linux/Iptables, Solaris/BSD/Irix/HP-UX ipfilter, Cisco IOS, Cisco PIX, Windows XP Firewall und Snort IDS. Ich werde mich in diesem Buch aus naheliegenden Gründen auf Iptables beschränken. Darüber hinaus können Sie mit diesem Werkzeug (gzip-)komprimierte Protokolldateien lesen, direkt Namensauflösungen durchführen und Berichte und Meldungen im Text- und HTML-Format erstellen. Diese Berichte können automatisch oder interaktiv erstellt werden. Für die Überwachung eines laufenden Fwlogwatch-Dienstes existiert ein Web-Interface. Die Konfiguration können Sie ab der Version 0.9 webbasiert über ein PHP-Skript (`fwlogwatch.php`) durchführen, das in dem `contrib/`-Verzeichnis mitgeliefert wird.

11.1.1 Installation von Fwlogwatch

Fwlogwatch installieren Sie üblicherweise auf der Firewall. Wenn Sie aus Sicherheitsgründen die Installation auf einem anderen System durchführen möchten, müssen Sie entweder die Protokolle manuell kopieren, oder Sie richten eine zentrale Protokollierung über das Netzwerk ein. Dies ist recht einfach und wird in Kapitel 9 beschrieben.

Fwlogwatch ist meines Wissens nur in der Debian-Distribution fest enthalten. Die Benutzer anderer Distributionen müssen Fwlogwatch von der Homepage `http://fwlogwatch.inside-security.de/` laden. Dort finden Sie aber sowohl RPM-Pakete als auch Quelltextarchive. Ich verwende daher immer das Source-RPM-Paket und übersetze es für die entsprechende Distribution neu:

```
# rpmbuild --rebuild fwlogwatch-<version>.src.rpm
```

Anschließend können Sie das erzeugte Paket installieren.

INFO

Da Sie für die Übersetzung einen Compiler und viele weitere Werkzeuge wie `make` und `patch` benötigen, sollten Sie das Paket auf einem anderen System übersetzen und anschließend auf die Firewall kopieren und dort installieren.

Damit Fwlogwatch automatisch als Dienst startet, rufen Sie unter Suse oder Fedora den `chkconfig`-Befehl auf. Auf einem Debian Lenny-System können Sie diesen Befehl nachinstallieren oder den Befehl `update-rc.d` nutzen.

```
# chkconfig --add fwlogwatch
```

Wenn Sie Fwlogwatch direkt aus den Quellen übersetzen möchten, müssen Sie zunächst das Paket von der Homepage laden, auspacken und mit `make; sudo make install` übersetzen. Mit dem Befehl `sudo make install-config` installieren Sie eine Beispielkonfigurationsdatei und ein Template in `/etc`.

11.2 Konfiguration von Fwlogwatch

Fwlogwatch unterstützt in der Version 1.2 zwei verschiedene Modi: Zusammenfassung (*Summary*) und Echtzeitantwort (*Realtime Response*). Ältere Versionen unterstützen auch noch einen interaktiven Berichtsmodus (*Report*). Der Modus Zusammenfassung erlaubt es, aus einem Firewall-Protokoll mit mehreren Tausend Einträgen in wenigen Sekunden einen Bericht zu erzeugen, in dem die Ereignisse zusammengefasst werden. Eine typische Zusammenfassung sieht folgendermaßen aus:

```
fwlogwatch-Zusammenfassung
Generiert Mit Mai 29 10:01:08 CEST 2002 von root.
18184 von 18362 Einträgen in der Datei "/home/xxx_firewall/messages.2"  ↵
        sind Paketfiltereinträge, 139 sind eindeutig.
Erster Paketfiltereintrag: Mai 12 13:21:27, letzter: Mai 17 13:44:10.
Alle Einträge wurden vom diesem Rechner geloggt: "fwsteinfurt".
Alle Einträge haben das selbe Ziel: "-".
Mai 17 13:56:24 03:04:25:40 - eth1 3001 udp Pakete (187968 Bytes) von  ↵
        192.168.222.200 (-) bis 111.112.222.222 (-) Port 53
Mai 17 13:56:37 03:04:25:44 - eth1 2991 udp Pakete (187172 Bytes) von  ↵
        192.168.222.200 (-) bis 192.33.4.12 (c.root-servers.net) Port 53
Mai 17 13:56:33 03:04:25:50 - eth1 2983 udp Pakete (186434 Bytes) von  ↵
        192.168.222.200 (-) bis 192.112.36.4 (G.ROOT-SERVERS.NET) Port  ↵
        53
Mai 17 13:56:37 03:04:25:44 - eth1 2976 udp Pakete (185906 Bytes) von  ↵
        192.168.222.200 (-) bis 192.5.5.241 (f.root-servers.net) Port 53
Mai 17 13:56:29 03:04:25:49 - eth1 2353 udp Pakete (142615 Bytes) von  ↵
        192.168.222.200 (-) bis 192.36.148.17 (i.root-servers.net) Port  ↵
        53
```


CE! Bitte Änderung bestätigen.

Editor's or typesetter's annotations (will be removed before the final TEX run)

```
Mai 17 11:14:21 03:20:24:24 AntiSpoofing: eth4 866 icmp Pakete (48496 ↵
        Bytes)
  von xxx.yyy.200.249 (-) bis 192.168.201.5 (-) Port 0
```

Die deutsche Variante der Zusammenfassung gibt leider als Übersetzung des englischen *to* das deutsche Wort *bis* anstatt sinnvollerweise *nach* an.

Sie können das verbessern, indem Sie in der Datei `po/de.po` im Fwlogwatch-Quelltext in Zeile 793 die folgende Ersetzung durchführen:

```
#: ../output.c:262
#, c-format
msgid " to %s"
msgstr " nach %s"
```

Mit dieser Zusammenfassung können Sie den Zustand der Firewall in den letzten Stunden überblicken. Sie erkennen leichter, welche Rechner die Richtlinien verletzen, als es die manuelle Analyse der Protokolldatei ermöglicht.

Im interaktiven Berichtsmodus[1] ist Fwlogwatch in der Lage, automatisch E-Mails zu generieren, die an ein CERT oder an die verantwortlichen Administratoren der „angreifenden" Rechner gesendet werden. Diese Funktion sollte zur Vermeidung von SPAM daher mit Vorsicht eingesetzt werden!

```
------------------------------------------------------------------------
From: root@kermit.spenneberg.de
To: [Insert address of abuse contact or CERT here]
Subject: Ereignisbericht 20020514-192.168.222.200

Dear Madam or Sir,

we would like to bring to your attention that [your organization's ↵
        networks] have been subject to heavy scans from a/several host/s ↵
         in your domain:

Angreifende IP-Adresse: 192.168.222.200
Ziel-IP-Adresse: 111.112.222.222
Anzahl der geloggten Versuche: 3001
Verfolgungsnummer: 20020514-192.168.222.200

Please take the appropriate steps to stop these scans.  In case you need ↵
        more information we would be glad to provide you with the ↵
        appropriate log file excerpts as far as they are available.
```

1 Ab Version 1.2 wird dieser Modus nicht mehr unterstützt.

```
Thank you.
Yours sincerely
[Your signature]
-------------------------------------------------------------------------
```

STOP

Leider besteht hier das Problem, dass die sinnvollerweise in Englisch gehaltene E-Mail mit deutschen Informationen gefüllt wird, wenn Fwlogwatch in einer deutschen Umgebung verwendet wird. Dieses Problem (und auch das oben angesprochene) lässt sich vermeiden, wenn vor dem Aufruf die entsprechende Umgebungsvariable für das Locale modifiziert wird:

```
LANG="en_US" fwlogwatch -i 1000
```

Allerdings werden dann auch die Zusammenfassungen nicht mehr übersetzt. Natürlich können Sie die Umgebungsvariable `LANG` *immer nur für den jeweiligen Lauf anpassen.*

Schließlich unterstützt Fwlogwatch auch einen Echtzeit-Modus. Hier überwachen Sie mit Fwlogwatch in Echtzeit die Protokolldatei.

Um diese Funktionen anzubieten, besitzt fwlogwatch eine Konfigurationsdatei: `fwlogwatch.config`.

Diese mitgelieferte Datei ist sehr gut dokumentiert und relativ selbsterklärend. Im Folgenden möchte ich Ihnen nur kurz die wichtigsten Optionen der verschiedenen Modi erklären. Die meisten Optionen können sowohl auf der Kommandozeile als auch in der Konfigurationsdatei definiert werden.

Allgemeine Optionen

Zunächst stehen Ihnen einige Funktionen in allen Modi zur Verfügung. Sinnvollerweise geben Sie die Parameter in der Konfigurationsdatei an. Mit den angegebenen Kommandozeilenoptionen können Sie aber auch die Funktion auf der Kommandozeile nutzen.

- `include_file`. Hiermit können Sie weitere Konfigurationsdateien einlesen. Damit lässt sich die Konfigurationsdatei aufsplitten.
- `verbose = <yes|no>, -v`. Diese Option ist nur auf der Kommandozeile sinnvoll. Die Angabe von zwei `-v` verstärkt die Wirkung.
- `resolve_hosts = <yes|no>, -n`. Hiermit schalten Sie die Namensauflösung an. Denken Sie daran, dass diese Namensauflösung Zeit kostet, Ihre Internetverbindung belastet und möglicherweise auch zu Einträgen in den Protokollen führt.
- `resolve_services = <yes|no>, -N`. Lesen Sie lieber Portnummern oder -namen?
- `input = <datei>, -f`. Hier geben Sie die zu verarbeitenden Protokolldateien an. Ab der Version 0.9.1 können Sie mehrere Dateien auf einzelnen Zeilen angeben. Ist die Datei mit `gzip` komprimiert worden (`.gz`), kann Fwlogwatch sie dekomprimieren. Bei Verwendung des Echtzeitmodus müssen Sie die Datei mit ihrem absoluten Pfad angeben. Der Einfachheit halber geben Sie die Datei immer mit ihrem absoluten Pfad an.

» `parser = infcpels, -P`. Hier wählen Sie das Protokollformat. Fwlogwatch kann viele Formate lesen. Das Netfilter/Iptables-Format wählen Sie mit `parser = n`.

» `src_ip = on, -S; dst_ip = on , -D; protocol = on, -p; src_port = on, -s; dst_port = on, -d; tcp_opts = on, -y`. Fwlogwatch sortiert und aggregiert die Einträge in der Protokolldatei. Sie können die Kriterien in der Konfigurationsdatei bestimmen.

» `exclude_src_host =, exclude_src_port =, exclude_dst_host =, exclude_dst_port =, include_src_host =, include_src_port =, include_dst_host =, include_dst_port =, -E`. Hiermit können Sie bestimmte Rechner und Ports von der Analyse durch Fwlogwatch ausnehmen. Dies ist zum Beispiel sinnvoll, wenn Sie einen bestimmten Rechner für Nmap-Audit-Scans (siehe Abschnitt 15.3) verwenden. Sie können auch auf der Kommandozeile die zu ignorierenden Rechner angeben. Die Kommandozeilenoption `-E` verwendet `e` für *exclude* und `i` für *include*. Die Buchstaben `hpcb` stehen für *host*, *port*, *chain* und *branch*. In den ersteren beiden Fällen wird mit den Buchstaben `s` oder `d` noch *source* oder *destination* angegeben. Beispiel: `-Eehs127.0.0.1`. Ab der Version 1.0 können Sie hier auch die CIDR-Notation verwenden (Beispiel: `192.168.1.0/25`).

» `exclude_chain, exclude_branch, ... , -E`. Hiermit können Sie ganze Ketten oder Aktionen in der Fwlogwatch-Auswertung ignorieren. Das Format der Kommandozeilenoption `-E` wurde bereits bei der letzten Option erklärt. Beispiel: `-EecOUTPUT`.

» `sort_order = [ctezpbSsDd][ad], -O.`. Auch die Sortierung der Protokolleinträge in der Auswertung können Sie bestimmen. Der erste Buchstabe definiert das Sortierkriterium: `c` count, `t` Startzeit, `e` Endzeit, `z` Dauer, `n` Zielname, `p` Protokoll, `b` Anzahl der Bytes, `S` Source, `s` Sourceport, `D` Ziel, `d` Zielport. Der zweite Buchstabe definiert die Sortierrichtung: aufsteigend (`a`, *ascending*) oder abfallend (`d`, *descending*). Es können mehrere Sortierungen angegeben werden, die in der angegebenen Reihenfolge genutzt werden. Default ist `tacd`.

Weitere Optionen erlauben ab der Version 0.9.3 die Anpassung der Ausgabe an die eigenen Wünsche:

» `title =`. Hiermit kann der Titel für die Ausgabe definiert werden. Default ist „fwlogwatch summary“ im Zusammenfassungsmodus oder „fwlogwatch status“ im Echtzeitmodus.

» `stylesheet =`. Diese Option erlaubt die Angabe eines alternativen Stylesheets. Im Echtzeitmodus muss dies ein externer URL sein, der mit `http://` beginnt.

» `textcolor =, bgcolor =, rowcolor1 =, rowcolor2 =`. Wenn die Farben zusätzlich angepasst werden sollen, können Sie das mit diesen Optionen einstellen.

Fwlogwatch-Zusammenfassung

Im Zusammenfassungsmodus können Sie folgende weitere Optionen nutzen:

» `data_amount, -b`. Hiermit erhalten Sie bei jedem Angriff die Summe der Pakete.

» `start_times = yes, -t; end_times = yes, -e`. Dies zeigt Ihnen den Start und das Ende des Angriffs an.

» `duration = yes`, `-z`. Dauer eines Angriffs.

» `html = yes`, `-w`. Erzeugt eine HTML-Ausgabe (siehe Abbildung 11.1, ab der Version 1.0 in XHTML 1.1).

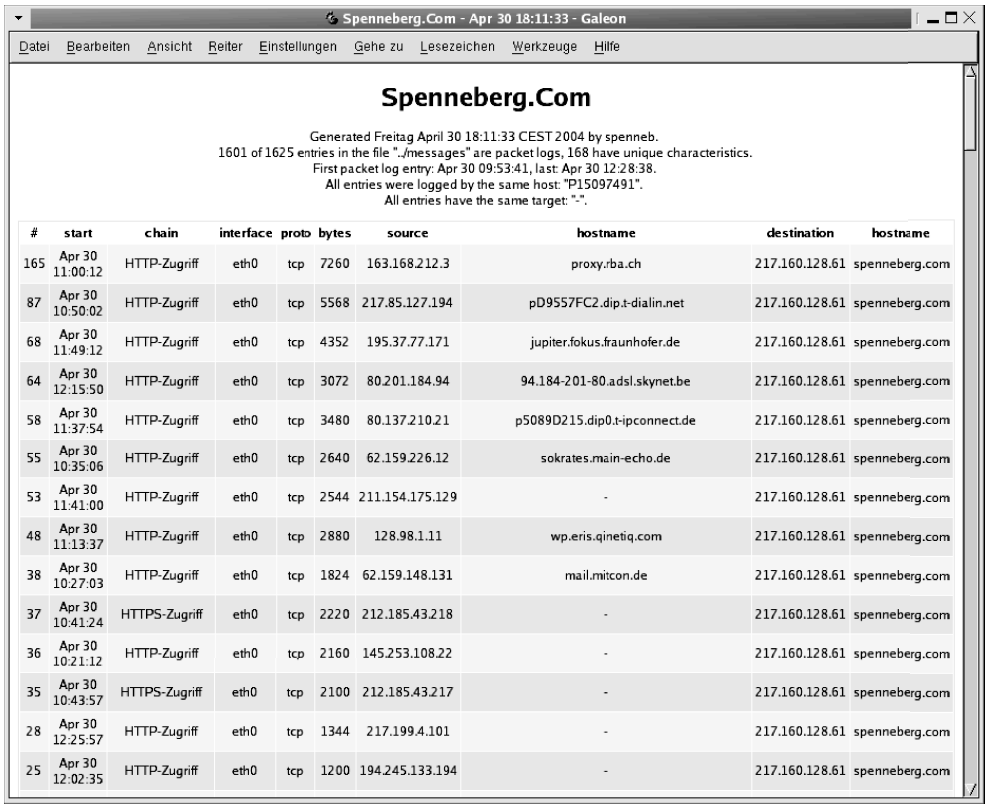
» `output = <datei>`, `-o`. Fwlogwatch schreibt die Ausgabe in eine Datei.

» `recent = <zeitraum>[mhdwy]`, `-l`. Wenn Sie sich nur für die letzte Woche interessieren, können Sie als Zeitraum `recent=w` verwenden.

» `at_least = <anzahl>`, `-m`. Versteckt Einträge, die nur selten vorkommen.

» `maximum = <anzahl>`, `-M`. Hiermit kann die Anzahl der Einträge in dem Bericht beschränkt werden.

» `whois_lookup = no`, `-W`. Diese Option weist Fwlogwatch an, Informationen über die Source-IP-Adressen in der Whois-Datenbank zu ermitteln. Diese Option sollte nur mit Vorsicht eingesetzt werden, da sie sehr langsam ist und die Whois-Datenbanken stark belastet.



#	start	chain	interface	proto	bytes	source	hostname	destination	hostname
165	Apr 30 11:00:12	HTTP-Zugriff	eth0	tcp	7260	163.168.212.3	proxy.rba.ch	217.160.128.61	spenneberg.com
87	Apr 30 10:50:02	HTTP-Zugriff	eth0	tcp	5568	217.85.127.194	pD9557FC2.dip.t-dialin.net	217.160.128.61	spenneberg.com
68	Apr 30 11:49:12	HTTP-Zugriff	eth0	tcp	4352	195.37.77.171	jupiter.fokus.fraunhofer.de	217.160.128.61	spenneberg.com
64	Apr 30 12:15:50	HTTP-Zugriff	eth0	tcp	3072	80.201.184.94	94.184-201-80.adsl.skynet.be	217.160.128.61	spenneberg.com
58	Apr 30 11:37:54	HTTP-Zugriff	eth0	tcp	3480	80.137.210.21	p5089D215.dip0.t-ipconnect.de	217.160.128.61	spenneberg.com
55	Apr 30 10:35:06	HTTP-Zugriff	eth0	tcp	2640	62.159.226.12	sokrates.main-echo.de	217.160.128.61	spenneberg.com
53	Apr 30 11:41:00	HTTP-Zugriff	eth0	tcp	2544	211.154.175.129	-	217.160.128.61	spenneberg.com
48	Apr 30 11:13:37	HTTP-Zugriff	eth0	tcp	2880	128.98.1.11	wp.eris.qinetiq.com	217.160.128.61	spenneberg.com
38	Apr 30 10:27:03	HTTP-Zugriff	eth0	tcp	1824	62.159.148.131	mail.mitcon.de	217.160.128.61	spenneberg.com
37	Apr 30 10:41:24	HTTPS-Zugriff	eth0	tcp	2220	212.185.43.218	-	217.160.128.61	spenneberg.com
36	Apr 30 10:21:12	HTTP-Zugriff	eth0	tcp	2160	145.253.108.22	-	217.160.128.61	spenneberg.com
35	Apr 30 10:43:57	HTTPS-Zugriff	eth0	tcp	2100	212.185.43.217	-	217.160.128.61	spenneberg.com
28	Apr 30 12:25:57	HTTP-Zugriff	eth0	tcp	1344	217.199.4.101	-	217.160.128.61	spenneberg.com
25	Apr 30 12:02:35	HTTP-Zugriff	eth0	tcp	1200	194.245.133.194	-	217.160.128.61	spenneberg.com

Abbildung 11.1: **Fwlogwatch kann im Zusammmenfassungsmodus auch eine HTML-Seite erzeugen.**

Wenn Sie anfangen, mit dem Zusammenfassungsmodus zu experimentieren, können Sie zu Beginn die folgende Konfigurationsdatei verwenden und dann an die eigenen Bedürfnisse anpassen:

```
resolve_hosts = yes
parser = n s
rc_ip = on
dst_ip = on
data_amount = yes
duration = yes
protocol =on
dst_port = on
sort_order = tacd
```

Fwlogwatch-Meldung

In bestimmten Umgebungen kann es sinnvoll sein, direkt bei der Analyse der Protokolle gewisse Ereignisse zu melden. Im Berichtsmodus besteht die Möglichkeit, eine E-Mail für jedes Angriffsereignis zu versenden. Dieser Modus steht allerdings ab der Version 1.2 nicht mehr zur Verfügung.

Diese Meldung kann an ein CERT oder an die verantwortlichen Administratoren des angreifenden Rechners gesendet werden. Die Meldung erfolgt nicht in Echtzeit, sondern zeitversetzt bei der Analyse der Protokolle mit Fwlogwatch.

Natürlich sollten Sie mit dieser Funktion sehr vorsichtig sein. E-Mail-Adressen können gefälscht werden. Der scheinbare Angriff kann harmlos sein. Sie können sich lächerlich machen. Prüfen Sie daher genau die Ereignisse, bevor Sie eine E-Mail versenden.

Für die Konfiguration des E-Mail-Versands stehen Ihnen die folgenden Funktionen zur Verfügung:[2]

- `interactive = <anzahl>, -i`. Dies aktiviert den Meldungsmodus. Für alle Angriffe mit mehr Paketen als der angegebenen Anzahl werden Meldungen erzeugt.
- `sender = <email>, -F`. Die E-Mail-Adresse des Absenders. Hier sollten Sie Ihre E-Mail-Adresse oder ein Funktionskonto eintragen (z.B. `firewall-admin@spenneberg.net`), das Sie verwenden.
- `recipient = <email>, -T`. Die E-Mail-Adresse des Empfängers. Dies kann Ihr internes CERT sein. Senden Sie bitte nichts[TS: j] ohne Aufforderung an ein externes CERT.
- `cc = <email>, -C`. Carbon Copy. Die E-Mail wird zusätzlich an diese Adresse gesendet. Diese Funktion ermöglicht es Ihnen, einfach jede E-Mail zu archivieren oder an mehrere CERTs zu versenden.

2 Die Optionen werden ab der Version 1.2 der Konfigurationsdatei für den Versand der Zusammenfassungen verwendet.

[TS: j] Bitte bestätigen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

» `template = <datei>, -I`. Template für die E-Mail (Default: `/etc/fwlogwatch.template`). Diese Datei können Sie frei nach Ihren Wünschen anpassen.

Damit Sie direkt loslegen können, habe ich hier eine einfache Beispielkonfiguration für Sie:

```
# Meldungsmodus interactive = 3000
parser = n
sender = ralf@spenneberg.de
recipient = incident@firma.de
cc = incident@spenneberg.de
template = /etc/fwlogwatch.incident
```

Echtzeit

Im Echtzeitmodus ist Fwlogwatch in der Lage, Echtzeitbenachrichtigungen zu versenden und sogar aktiv zu reagieren. Hierzu konfigurieren Sie zwei mitgelieferte Skripten, die Sie zuvor anpassen können. Das erste Skript, `fwlw_notify`, versendet Echtzeit-Alarmierungen per E-Mail oder Windows-Nachrichtendienst. Das zweite Skript, `fwlw_respond`, kann Firewall-Regeln mit `iptables` hinzufügen, um weitere Angriffe des Hosts zu verhindern. Den Einsatz des zweiten Skripts sollten Sie abwägen, da ansonsten ein Angreifer einen Denial-of-Service-Angriff auf Ihre Firewall durchführen kann. Hierzu würde es genügen, wenn der Angreifer die IP-Adressen der DNS-Root-Server oder einiger wichtiger Webserver fälschen würde. Fwlogwatch würde bei Verwendung des Skripts `fwlw_respond` diese IP-Adressen sperren und Ihnen keine Kommunikation mit den Systemen erlauben, deren IP-Adressen beim Angriff gefälscht wurden.

» `realtime_response = yes, -R`. Diese Option schaltet den Echtzeitmodus an.

» `ipchains_check = no`. Diese Option überprüft die Korrektheit der `ipchains`-Regeln beim Start. Beim Einsatz von `iptables` ist diese Funktion obsolet.

» `pidfile = <datei>`. Hiermit schreibt Fwlogwatch seine PID in die angegebene Datei. Damit ist es einfacher, einem laufenden Fwlogwatch-Prozess Signale zu senden.

» `run_as =`. Diese Option weist Fwlogwatch an, seinen Benutzerkontext zu ändern. Dabei wird der `fwlogwatch`-Befehl zunächst als `root` gestartet. So kann der Prozess sowohl auf die Protokolldatei als auch auf den privilegierten Port für den Webserver zugreifen. Anschließend werden diese `root`-Privilegien abgegeben.

» `alert_threshold = <anzahl>, -a`. Beim Überschreiten dieses Schwellenwertes löst Fwlogwatch Alarm aus (Default 5).

» `recent = , -l`. Lebensdauer der Ereignisse. Nach Ablauf vergisst Fwlogwatch die Ereignisse (siehe Zusammenfassungsmodus).

» `notify = yes, -A`. Führt das Notify-Skript aus.

» `respond = yes, -B`. Führt das Respond-Skript aus.

» `notification_script = <datei>; response_script = <datei>`. Name des entsprechenden Skripts (`fwlw_(notify|respond)`). In den mitgelieferten Skripten (`contrib/`) befinden sich weitere Beispiele und Kommentare.

» `known_host = <ip-address>, -k`. Die hier in CIDR-Notation (192.168.0.0/24) angegebenen Rechner oder Netzwerke lösen keine Warnmeldung aus.

Im Echtzeitmodus kann ein Webserver in Fwlogwatch aktiviert werden. Damit überwachen Sie den Echtzeitmodus. Hierfür stehen die folgenden Optionen zur Verfügung:

» `server_status = yes, -X`. Damit aktivieren Sie den Webserver.

» `bind_to = <ip-address>`. Hiermit kann die IP-Adresse angegeben werden, auf der anschließend der Webserver seine Dienste anbieten soll (Default: 127.0.0.1).

» `listen_port = <port>`. Diese Angabe definiert den TCP-Port, der verwendet wird.

» `listen_to = <ip-address>`. Lediglich die angegebene IP-Adresse darf sich mit dem Webserver verbinden.

» `status_user = <user>, status_password = <crypt>`. Fwlogwatch verlangt von dem Benutzer bei der Anmeldung eine Authentifizierung. Diese wird mit diesen Angaben definiert. Das angegebene Kennwort muss verschlüsselt angegeben werden. Fwlogwatch verwendet wie die klassischen UNIX-Systeme die `crypt`-Verschlüsselung. Sie können das Kennwort sehr leicht mit dem Befehl `htpasswd -nb <user kennwort>` erzeugen. Dieser Befehl ist üblicherweise in den Linux Apache-Paketen enthalten.

» `refresh = <Sekunden>`. Schließlich können Sie einen automatischen Refresh der Webseite mit dieser Option konfigurieren.

Eine Beispielkonfiguration mit der Konfiguration des Webservers für die Überwachung kann folgendermaßen aussehen:

```
realtime_response = yes parser = n run_as = fwloguser alert_threshold = 5 ↵
         notify = yes
notification_script = /usr/local/sbin/fwlw_notify server_status = yes  ↵
        bind_to = 192.168.0.1
listen_port = 8888 status_user = ralf status_password = gieOlzYkkk9sQ  ↵
        refresh = 10
```

Ein Screenshot der Web-Überwachung im Echtzeitmodus ist in Abbildung 11.2 zu sehen. Ab der Version 1.0 kann über das Web-Interface auch eine Konfiguration von Fwlogwatch erfolgen (siehe Abbildung 11.3).

INFO

Wenn Sie Fwlogwatch im Echtzeitmodus als Daemon einsetzen und ein anderes Werkzeug die Protokolldatei rotiert, verliert Fwlogwatch den Zugang zu den Protokollen. Sie müssen, damit Fwlogwatch nun die neue Datei liest, Fwlogwatch ein Signal schicken. Ein einfaches SIGHUP reicht jedoch nicht aus. Hier liest Fwlogwatch nur seine Konfigurationsdatei neu. Sie müssen ein SIGUSR1 an den Prozess schicken:

```
kill -USR1 $(cat /var/run/fwlogwatch-pid)
```

Abbildung 11.2: **Im Echtzeitmodus kann Fwlogwatch seinen Status über ein Web-Interface anzeigen.**

Weitere Web-Funktionen

Fwlogwatch weist noch einige weitere Funktionen auf, mit denen Sie komfortabel über ein Web-Interface Berichte erzeugen können. So enthält das `contrib/`-Verzeichnis der Fwlogwatch-Distribution zwei CGI-Programme und ein PHP-Programm. Diese können Sie nach Anpassung auf einem Webserver installieren und auf diese Weise mit ihnen weitere Berichte erzeugen.

Das `fwlogsummary.cgi`-Skript erzeugt insgesamt 8 verschiedene HTML-Berichte, die anschließend mit einem Webbrowser betrachtet werden können. Hierzu ist es erforderlich, dass der Webserver das CGI-Skript und den Befehl `fwlogwatch` ausführen darf und Leserechte

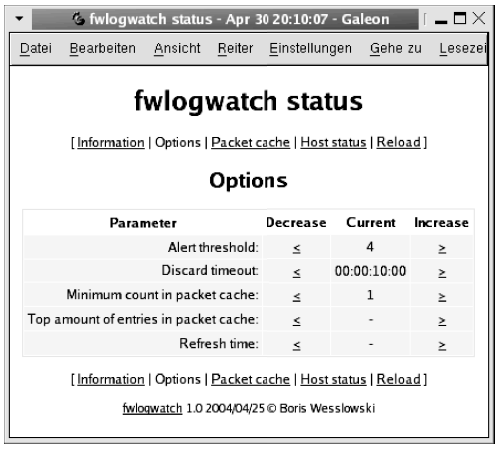


Abbildung 11.3: **Das Web-Interface erlaubt auch eine Konfiguration von Fwlogwatch.**

an der Protokolldatei hat. Es ist aber auch möglich, das Skript zu bestimmten Uhrzeiten automatisch über Cron aufzurufen. Die so erzeugten Berichte werden nach verschiedenen Aspekten sortiert und auf dem Webserver abgelegt. Sie erhalten so sehr einfach 8 verschiedene Berichte der letzten Stunde, die die Einträge sortiert nach der Absender-Adresse, Zieladresse, Ports etc. aufführen. Die Erstellung all dieser Berichte dauert eine gewisse Zeit. Daher empfiehlt sich der Aufruf per Cron-Daemon.

Das zweite CGI-Skript, `fwlogsummary_small.cgi`, erzeugt lediglich einen Bericht, der die letzte Stunde zusammenfasst. Hiermit können Sie bei einer Alarmierung sehr schnell einen Überblick über den Zustand der Firewall erhalten.

Das PHP-Skript erlaubt schließlich online die Auswahl der Optionen für die Anzeige des Berichts. Dabei können fast alle Möglichkeiten von Fwlogwatch ausgereizt werden. Abbildung 11.4 zeigt die möglichen Optionen.

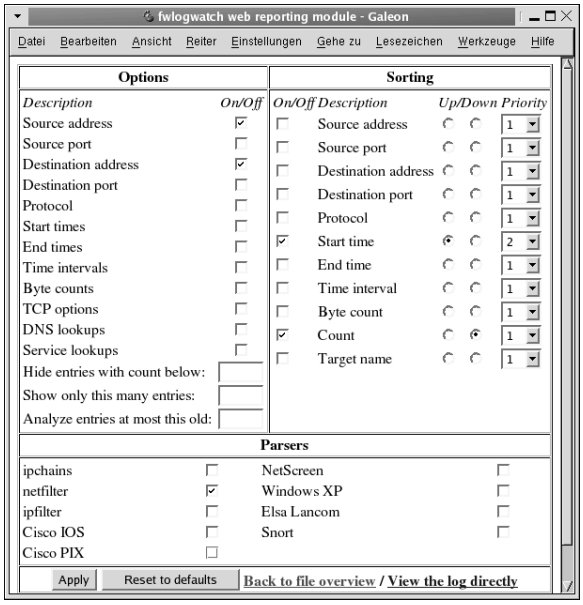


Abbildung 11.4: **Mit dem PHP-Skript wählen Sie komfortabel die Optionen für die Erzeugung des Berichts.**

11.3 IP Tables State (IPTState)

Der Befehl `iptstate` (`http://www.phildev.net/iptstate/index.html`) ist sehr hilfreich bei der täglichen Administration und Wartung einer Iptables-Firewall. Er ist mit dem Befehl `top` vergleichbar, der die Prozesse in einer sortierten Liste darstellt. Allerdings sollten Sie diesen Befehl nicht ununterbrochen auf Ihrer Firewall laufen lassen, da er das System stark belastet. Um jedoch bei nur wenigen Verbindungen diese leicht lesbar anzuzeigen und zu sortieren, ist der Befehl sehr gut geeignet.

Bitte auf Abbildung 11.5 im Text verweisen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

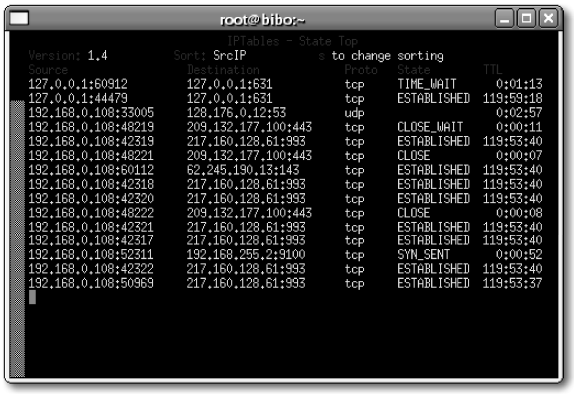


Abbildung 11.5: **Der Befehl iptstate zeigt die aktuellen Verbindungen der Firewall an.**

Der Befehl ist inzwischen in den meisten Distributionen enthalten und kann dort recht einfach über die Paketverwaltung installiert werden. Um den Befehl aus den Quellen zu installieren, benötigen Sie auf Ihrer Firewall lediglich das Ncurses-Paket. Dieses ist in den meisten Distributionen enthalten und auch schon installiert. Sie können IPTState nicht auf einem anderen System als der Firewall betreiben, da IPTState den aktuellen Zustand der Firewall direkt aus der Datei `nf_conntrack` liest. Auf älteren Kerneln ist dies die Datei `ip_conntrack`.

Falls bei der manuellen Installation mit `make; make install` ein Fehler auftritt, prüfen Sie bitte, ob das `ncurses-devel`-Paket Ihrer Distribution installiert ist.

Zur manuellen Installation nutzen Sie die folgenden Befehle:

```
$ cd /usr/local/src
$ tar xjf iptstate-<version>tar.bz2
$ cd iptstate-<version>
$ make
$ sudo make install
```

Nun sollten Sie bereits mit `man iptstate` die Handbuchseite des Befehls anzeigen können. Mit `iptstate` starten Sie bereits den Befehl, der Ihnen die Verbindungen über Ihre Firewall anzeigt.

Bei dem Aufruf des Programms können Sie über Kommandozeilenoptionen die Sortierreihenfolge und Filterfunktionen definieren. So erlaubt die Option `-b` die Sortierung der Verbindungen nach der Zieladresse (`-bd`), nach dem Protokoll (`-bp`), nach dem Zustand (`-bs`) oder nach ihrer Lebensdauer (`-bt`). Leider lassen sich diese nicht miteinander kombinieren. Mit den Optionen `-S` und `-D` können Sie nur die Verbindungen mit einer bestimmten Absender- oder Ziel-IP-Adresse anzeigen lassen. Die Option `-f` zeigt keinerlei Loopback-Verbindungen an. Die Option `-l` führt für jede Adresse eine Namensauflösung durch. Dabei werden die Namen der Clients von rechts abgeschnitten, da die meisten Clients aus Ihrer eigenen Domäne kommen werden, und die Namen der Server werden von links abgeschnitten, da Sie wahrschein-

lich am meisten an den Domänen interessiert sind. Wenn Sie die Namensauflösung nutzen, ist die gleichzeitige Angabe von `-L` sinnvoll. Hiermit werden alle DNS-Verbindungen zur Namensauflösung in der Anzeige unterdrückt. Sobald Sie sich in dem interaktiven Modus befinden, können Sie

- mit der Leertaste einen sofortigen Refresh der Anzeige auslösen,
- mit `r` rückwärts sortieren,
- mit `l` die DNS-Auflösung an- und abschalten,
- mit `n` die Anzeige der DNS-Verbindung an- und abschalten,
- mit `s` die Sortierung nach der nächsten Spalte durchführen und
- mit `q` den Befehl beenden.

Häufig benötigen Sie jedoch keine sich immer neu aktualisierende Anzeige, sondern möchten nur den aktuellen Zustand der Firewall in einer leicht lesbaren Liste anzeigen. Hierfür bietet IPTState die Option `-s`. Dies ist der Single-Run-Modus. Das Werkzeug gibt den aktuellen Zustand aus und beendet sich. Sie können die Ausgabe leicht mit `grep` und `awk` weiterbearbeiten oder auch per E-Mail verschicken.

```
# iptstate -s
IP Tables State Top -- Sort by: SrcIP
Source                Destination         Proto State       TTL
192.168.255.100:52661 217.160.128.61:993  tcp   ESTABLISHED 119:54:02
192.168.255.100:52662 217.160.128.61:993  tcp   ESTABLISHED 119:54:02
192.168.255.100:35957 209.132.177.100:443 tcp   CLOSE       0:00:05
192.168.255.100:35956 209.132.177.100:443 tcp   CLOSE       0:00:04
192.168.255.100:52668 217.160.128.61:993  tcp   ESTABLISHED 119:54:01
192.168.255.100:50090 213.30.31.52:80     tcp   TIME_WAIT   0:00:52
192.168.255.100:56979 217.160.128.61:993  tcp   ESTABLISHED 119:54:02
192.168.255.100:59425 193.201.52.189:80   tcp   SYN_SENT    0:00:39
192.168.255.100:52667 217.160.128.61:993  tcp   ESTABLISHED 119:54:01
192.168.255.100:56978 217.160.128.61:993  tcp   ESTABLISHED 119:54:01
192.168.255.100:38752 192.168.255.1:53    udp               0:00:04
192.168.255.100:38759 192.168.255.1:53    udp               0:00:04
192.168.255.100:38760 192.168.255.1:53    udp               0:02:53
192.168.255.100:52666 217.160.128.61:993  tcp   ESTABLISHED 119:54:01
192.168.255.100:42223 213.30.31.52:80     tcp   TIME_WAIT   0:01:52
192.168.255.125:1069  192.168.255.100:139 tcp   ESTABLISHED 119:56:57
```

Die hohen Werte in der Spalte TTL sind auf die hohen Verweildauern von TCP-Verbindungen in der Zustandstabelle zurückzuführen. Aufgebaute TCP-Verbindungen verbleiben für 5 Tage in der Tabelle, während UDP-Verbindungen lediglich 180 Sekunden lang dort gespeichert werden.

11.4 Webfwlog Firewall Log Analyzer

Der Webfwlog Firewall Log Analyzer bietet Ihnen eine webbasierte Firewall-Protokollanalyse und Berichtserzeugung. Sie benötigen für den Einsatz einen Webserver mit mindestens PHP 4.1, eine MySQL- oder PostgreSQL-Datenbank und einen Webbrowser. Damit Webfwlog die Protokolle analysieren kann, benötigt er Lesezugriff auf die Protokolle der Firewall.

INFO

Webfwlog ist eines der wenigen Werkzeuge, die auch Protokolle aus einer Datenbank lesen können. Wenn Sie das `ULOG`-Target (siehe Abschnitt 24.1) einsetzen, können Sie damit die Protokollmeldungen in eine MySQL-Datenbank schreiben, die Sie mit diesem Werkzeug analysieren können!

Die aktuelle Version 0.94 von Webfwlog wurde am 12. Oktober 2009 veröffentlicht und ist unter `http://www.webfwlog.net/` sowohl als RPM als auch als Quelltext verfügbar. Sie können daher auf einer RPM-basierten Distribution die Installation sehr einfach vornehmen. Falls für Ihre Distribution kein binäres RPM verfügbar ist, empfehle ich, dies selbst zu bauen:

```
rpmbuild --rebuild webfwlog-<version>.src.rpm
```

Falls Sie keine RPM-basierte Distribution einsetzen oder der Bau des RPMs fehlschlägt, können Sie Webfwlog auch leicht aus dem Quelltext übersetzen. Hierbei sollten Sie die folgenden Optionen bei dem Aufruf von `./configure` angeben:

- `--sysconfdir=/etc`. In diesem Verzeichnis wird die Konfigurationsdatei `webfwlog.conf` gesucht.
- `--with-html-doc-root=/var/www/html`. Dies ist das DocumentRoot des Apache Webservers.
- `--enable-syslog`. Hiermit aktivieren Sie den Syslog-Parser. Wenn Sie ULOG verwenden, benötigen Sie diesen nicht.
- `--enable-mysql|--enable-pgsql`. Dies aktiviert die Unterstützung für die entsprechende Datenbank. Die Client-Bibliotheken für die Datenbank müssen auf dem System installiert sein. Sie benötigen die Datenbankunterstützung auch, wenn Sie den Syslog-Parser verwenden.

Dann können Sie die Übersetzung starten:

```
$ ./configure --sysconfdir=/etc --with-html-doc-root=/var/www/webfwlog -- ↵
        with-mysql
$ make
$ sudo make install
```

INFO

Wenn Sie nicht den Syslog-Parser verwenden, wundern Sie sich bitte nicht, dass bei dem `make`-Kommando nichts passiert. Eine Übersetzung ist nur nötig, wenn der Parser aktiviert wird. Dennoch ist der Aufruf des `./configure`-Skripts erforderlich.

INFO

Wenn Sie eine neue Version installieren, überprüft das Installationskript, ob bereits eine Konfigurationsdatei `/etc/webfwlog.conf` *vorhanden ist, und überschreibt diese nicht. Ein Upgrade ist also mit* `make install` *möglich.*

Nun müssen Sie noch die Datenbank für Webfwlog anpassen. Hierfür gibt es in den Verzeichnissen `./mysql/` bzw. `./pgsql/` jeweils ein Setup-Skript. Wechseln Sie einfach in das vorgesehene Verzeichnis, und rufen Sie `./setup` auf. Dieses Skript fragt nach den notwendigen Informationen und erzeugt ein MySQL-Skript, mit dem Sie die notwendige Datenbank erzeugen können.

Anschließend müssen Sie nur noch die Konfigurationsdatei `/etc/webfwlog.conf` und bei Bedarf die Konfigurationsdatei `/etc/ulogd.conf` anpassen.

INFO

Bei mir hat das Installationsskript teilweise bei der Erzeugung der Datenbank einen kleinen Fehler gemacht und die Rechte für die Datenbank nicht richtig gesetzt. Dies lässt sich leicht anschließend mit folgendem Befehl korrigieren:

```
# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 83 to server version: 4.1.12
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> grant all on webfwlog.* to webfwlog@localhost identified by ' ↵
        kennwort';
Query OK, 0 rows affected (0.03 sec)
mysql> Bye
```

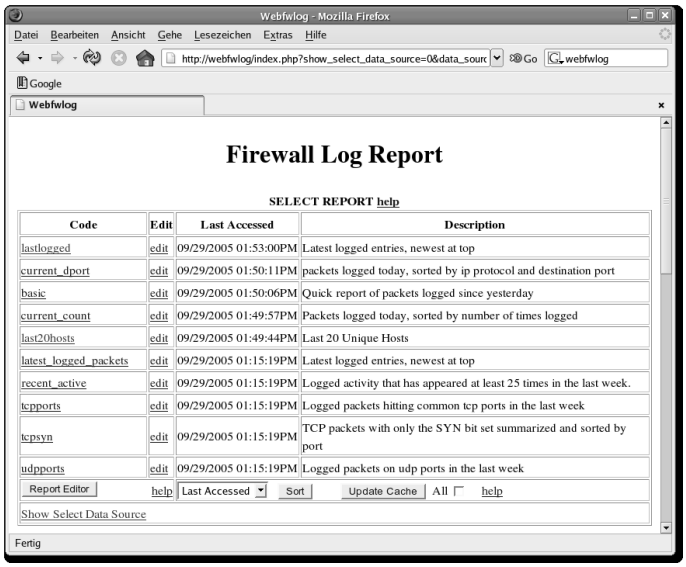


Abbildung 11.6: **Webfwlog besitzt einige vorgefertigte Berichte.**

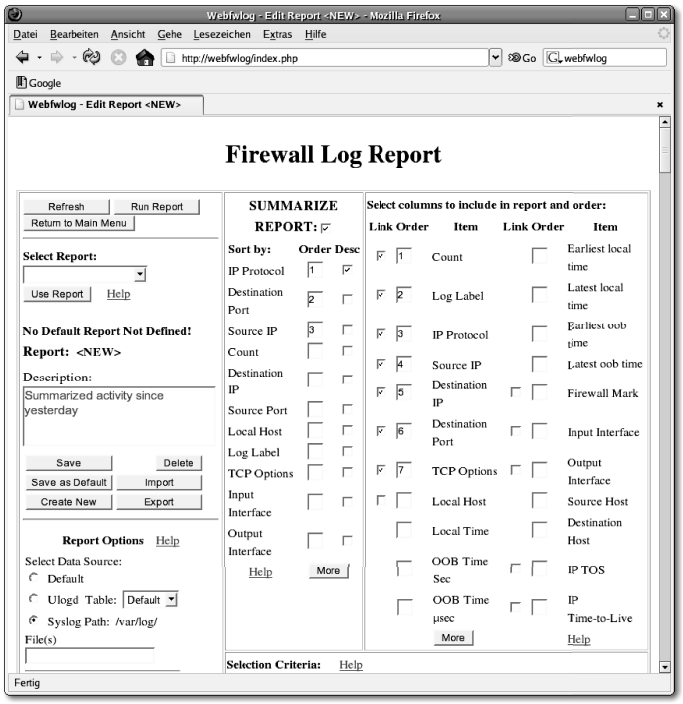


Abbildung 11.7: **Webfwlog bietet Ihnen alle Freiheiten zur Erzeugung des Berichts.**

Wenn Sie bei der Erzeugung des Datenbankskripts die Beispielskripten ausgewählt haben, können Sie direkt Webfwlog verwenden. Bei dem Zugriff auf Webfwlog bietet Ihnen die Weboberfläche direkt einige Beispielberichte an (siehe Abbildung 11.6).

Natürlich können Sie auch Ihre eigenen Berichte erzeugen. Hier stehen Ihnen sehr viele verschiedene Kriterien für die Erzeugung des Berichts zur Verfügung (siehe Abbildung 11.7).

11.5 NuLog2

NuLog2 (`http://software.inl.fr/trac/wiki/EdenWall/NuLog`) ist ein Frontend für das Ulogd-Mysql-Plug-In. Mit Nulog (ehemals ulog-php) können Sie webbasiert die Protokolle analysieren, die von `ULOG` geschrieben werden. Nulog wird für die NuFW geschrieben, die eine authentifizierende Firewall für Linux ist. Hinter diesen Produkten steht die Firma INL, die mit der EdenWall[3] auch ein kommerzielles Firewallprodukt anbietet, das auf diesen Technologien basiert. Allerdings können Sie Nulog auch auf jeder normalen Iptables-Firewall einsetzen, die den Ulogd-Daemon nutzt. Ursprünglich eine Sammlung von PHP-Skripten, die auf die Mysql-Datenbank zugreifen, wurde NuLog2 nun komplett in Python auf dem Twisted Framework

3 `http://www.edenwall.com`

TS Bitte diese Korrektur bestätigen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

aufgesetzt. Dennoch benötigt auch die neue Version eine Datenbank. Diese wird von Ulogd (siehe Abschnitt 24.3) oder Specter (siehe Abschnitt 24.4) mit Werten gefüllt.

Dabei können von Nulog2 auch IPv6-Protokolle ausgewertet werden. Hierzu müssen aber dann getrennte Datenbanken für IPv4 und IPv6 genutzt werden.

Mithilfe von NuLog2 können Sie diese Daten dann mit einem Browser analysieren (siehe Abbildung 11.8).

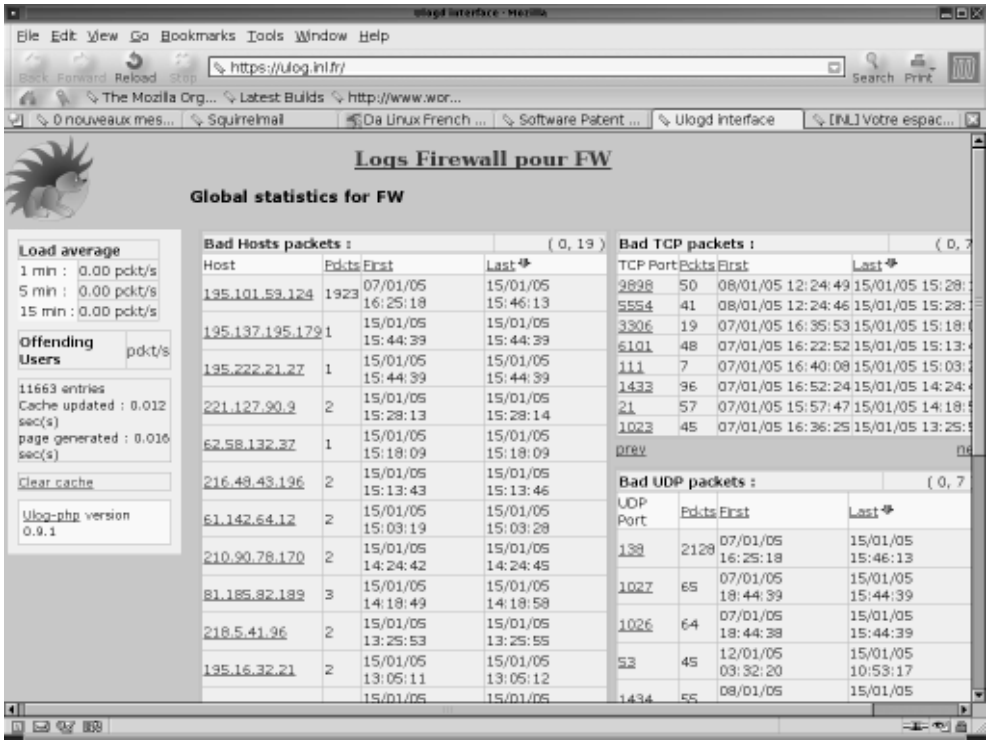


Abbildung 11.8: **NuLog2 bietet die Analyse über einen Webserver.**

11.6 EpyLog Log Analyzer

Der EpyLog Log Analyzer (`https://fedorahosted.org/epylog/`) ist ein Syslog-Parser, der Protokolldateien liest, verarbeitet und einen leicht lesbaren HTML-Bericht erzeugt. Dieser Bericht wird dann per E-Mail versandt. EpyLog wurde für Umgebungen mit vielen Log-Servern geschrieben, die über den Syslog oder den Syslog-ng auf einen zentralen Logserver protokollieren (siehe Kapitel 9). Dort verarbeitet es dann die Protokolle. Es ähnelt in seiner Funktion damit dem Paket Logwatch (`http://www.logwatch.org`).

Da EpyLog über ein spezielles Firewall-Verarbeitungsmodul verfügt, möchte ich es in diesem Kapitel erwähnen.

Bitte bestätigen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

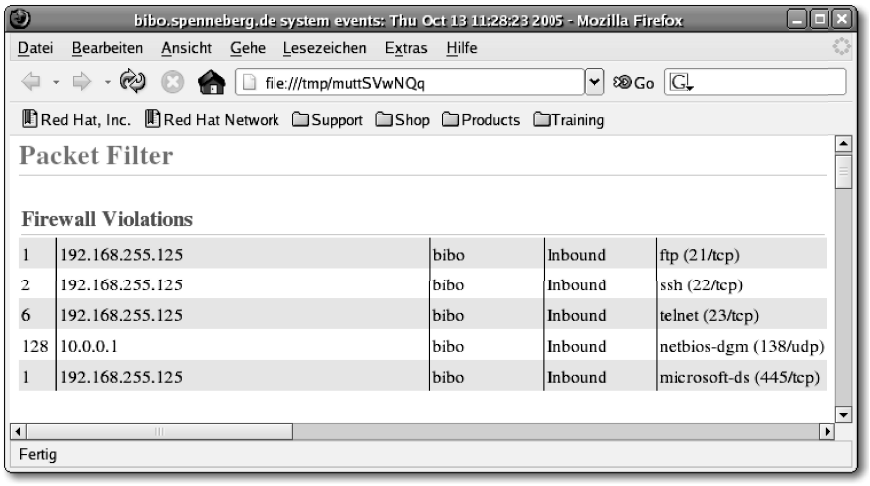


Abbildung 11.9: **EpyLog erzeugt einen HTML-Bericht, in dem die Firewall-Meldungen zusammengefasst werden.**

Die Installation von EpyLog ist sehr einfach. Für viele Distributionen existieren fertige Pakete. EpyLog ist in Python geschrieben und verlangt lediglich als Voraussetzung Python-2.2 oder neuer und die `libxml2-python`.

EpyLog wird üblicherweise täglich aufgerufen und erzeugt einen Bericht über die letzten 24 Stunden. Dabei können Sie das Erscheinungsbild des Berichts und den E-Mail-Versand in der Konfigurationsdatei `/etc/epylog/epylog.conf` anpassen. Meist sind jedoch die Default-Einstellungen bereits ausreichend. Bei seinem Aufruf erzeugt dann EpyLog einen Bericht, der auch die Protokollmeldungen der Firewall beinhaltet. Hier soll nur kurz ein Auszug eines Berichts zur Demonstration gezeigt werden (siehe Abbildung 11.9).

EpyLog kann so die Echtzeit-Protokollanalyse durch die Erzeugung von Berichten unterstützen, die auch andere Meldungen erfassen.

12. Administrationsoberflächen

In diesem Kapitel möchte ich Ihnen einige Administrationsoberflächen vorstellen, die mir besonders gut gefallen haben oder die gewisse Probleme besonders intelligent lösen. Es kann sich dabei nur um eine subjektive Auswahl der allgemein zur Verfügung stehenden Oberflächen handeln. Wenn Ihr favorisiertes Werkzeug nicht dabei ist, verzeihen Sie es mir hoffentlich. Wenn Sie der Meinung sind, dass es auf jeden Fall dazugehört, schreiben Sie mir eine E-Mail. Vielleicht kenne ich es einfach noch nicht.

Da die grafischen Werkzeuge am einfachsten durch die Darstellung der Screenshots beschrieben werden können, befinden sich in diesem Kapitel mehr Bilder als im gesamten restlichen Buch zusammen.

12.1 Firewall Builder

Eine der fortgeschrittensten grafischen Oberflächen für die Konfiguration einer Linux-Firewall mit Iptables ist sicherlich der Firewall Builder (`http://www.fwbuilder.org`). Seine grafische Erscheinung (siehe Abbildung 12.2) ist einem kommerziellen Firewall-Produkt nachempfunden worden und hat einige sehr interessante Eigenschaften:

- Abstraktion von Rechnern und Netzen in Objekten
- echtes Drag & Drop von Objekten
- Speicherung im XML-Format
- Unterstützung von IPv4 und IPv6
- automatische Regelprüfung und Optimierung
- Erzeugung der Konfiguration auf einer Workstation und automatische Übertragung der Firewall-Konfiguration mit SCP auf die Firewall
- Template für die einfache Erzeugung einer neuen Firewall-Konfiguration
- Unterstützung einer Bridge-Firewall
- ein Handbuch mit inzwischen mehr als 400 Seiten

Firewall Builder unterstützt aktuell die Erzeugung von Firewall-Konfigurationen für Iptables, pf, (FreeBSD, OpenBSD und Solaris) und Cisco. Sie können den Firewall Builder unter Linux, FreeBSD, MacOS X und Windows XP$_{CE}^{n}$ einsetzen. Für die letzteren beiden Betriebssysteme stehen binäre Pakete zur Verfügung. Diese sind für 30 Tage ohne Einschränkung nutzbar. Anschließend müssen Sie eine kommerzielle Lizenz erwerben. Diese ist im Moment abhängig

von der Anzahl der verwalteten Firewall-Systeme. Für alle weiteren unterstützten Betriebssysteme und Zielplattformen ist Firewall Builder unter der GPL freigegeben.

Für die einfache Installation finden Sie auf der Homepage RPM-Pakete für die gängigen Distributionen. Viele Distributionen enthalten diese Pakete auch bereits. Die Installation aus den Quellarchiven werde ich daher hier nicht beschreiben. Aktuell ist die Version 4.1.

Für eine funktionsfähige Installation müssen Sie meist mehrere Pakete installieren. Mindestens sind dies:

1. libfwbuilder
2. fwbuilder

Meist genügt es aber, das Paket fwbuilder zu installieren. Sämtliche weiteren Abhängigkeiten werden häufig automatisch aufgelöst. Fwbuilder bietet auch eigene Repositorien für Aptitude- und YUM-basierte Installationen an. Dann genügt ein `yum install fwbuilder`.

Sobald Sie diese Pakete installiert haben, können Sie Firewall Builder starten. Hierzu rufen Sie einfach den Befehl `fwbuilder` auf. Anschließend werden Sie von einem Popup-Fenster gefragt, ob Sie einen „Quick Start Guide“ sehen möchten (siehe Abbildung 12.1). Dieser Guide ist ein Video-Tutorial, das mit dem Browser betrachtet wird. Hierzu benötigen Sie einen Internetzugang, da dieses aktuell 4-minütige englische Youtube-Video aus dem Internet geladen wird.

Bei dem Start des Programms müssen Sie sich entscheiden, ob Sie eine aktuelle Konfiguration importieren oder komplett neu beginnen möchten (siehe Abbildung 12.2).

Sie beginnen die Konfiguration mit dem Erzeugen einer neuen Firewall. Hierzu klicken Sie entweder mit der rechten Maustaste auf den Firewall-Ordner in der linken Spalte und wählen aus dem Kontextmenü New Firewall, oder Sie klicken mit der linken Maustaste auf das Create new Firewall-Icon auf dem Hintergrund. Dann müssen Sie einen Namen für die Firewall vergeben und das Zielbetriebssystem auswählen. Die Verwendung von Templates erleichtert Ihnen die weitere Konfiguration der Firewall.TS°

Abbildung 12.1: **Beim Start von Firewall Builder können Sie wählen, ob Sie einen Quick Start Guide sehen möchten.**

CE? Auch unter neueren Versionen der Microsoft-Betriebssysteme einsetzbar?

TS° Bitte Verweis auf Abbildung 12.3 im Text einfügen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

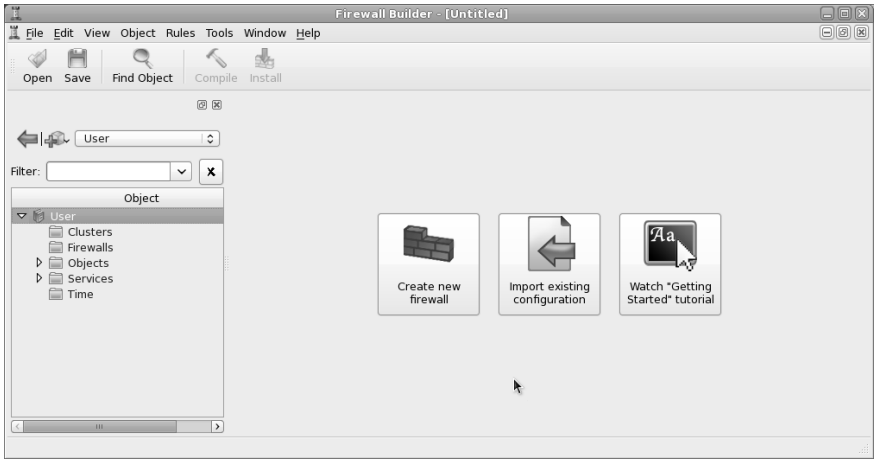


Abbildung 12.2: **Zum Start bietet Ihnen der Firewall Builder den Import vorhandener Regeln an.**

Nun können Sie ein vorkonfiguriertes Template für die weitere Anpassung auswählen. Insgesamt gibt es mehrere verschiedene Templates auch für unterschiedliche Zielplattformen. Die wichtigsten werden im Folgenden aufgeführt:

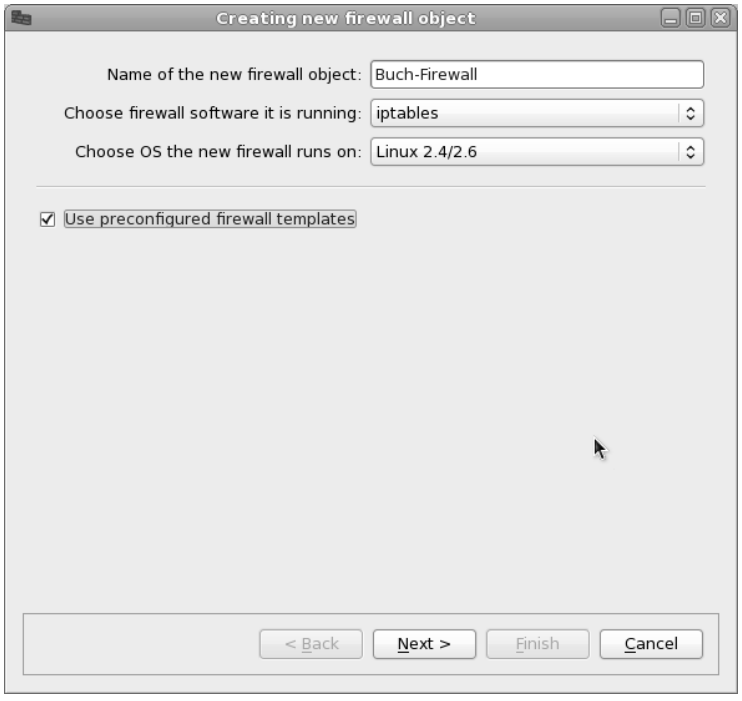


Abbildung 12.3: **Firewall Builder unterstützt auch FreeBSD, Solaris, MacOS X und andere.**

fw template 1: Eine Firewall mit zwei Netzwerkkarten. Die externe Netzwerkkarte hat eine dynamische Adresse. Der Zugriff von innen nach außen ist nicht eingeschränkt.

fw template 2: Dieses Template entspricht dem *fw template 1*. Zusätzlich bietet die Firewall DHCP- und DNS-Dienste für das interne Netz an.

fw template 3: Diese Firewall besitzt drei Netzwerkkarten. An der `eth2` ist die DMZ angeschlossen.

host fw template 1: Dieses Template schützt einen einzelnen Rechner. Lediglich SSH-Zugriff ist erlaubt.

OpenWRT template: Dieses Template ist für die Linux-Firmware von OpenWRT (`http://www.openwrt.org`) vorbereitet, die auf Linksys-DSL-Routern läuft.TSP Die Netzwerkkarte `eth0` ist die innere, und `eth1` ist die äußere Netzwerkkarte.

web server: Dieses Template schützt einen Webserver. Es erlaubt den Zugriff auf den Port 80 und 443.

Wählen Sie das Template, das Ihrer Konstellation am nächsten kommt. Für dieses Beispiel habe ich das *fw template 2* gewählt (siehe Abbildung 12.4).

Im Folgenden werden Sie nun noch nach einigen weiteren Informationen gefragt. Hier geben Sie zum Beispiel an, welche IP-Adressen Ihre Netzwerkkarten (in diesem Fall eth0, eth1 und lo) haben. Dabei ist die äußere Netzwerkkarte mit einer dynamischen IP-Adresse wahrscheinlich in den meisten Fällen richtig konfiguriert. Jedoch sollten Sie die innere Konfiguration an Ihr Netzwerk anpassen (siehe Abbildung 12.5). Die Loopback-Schnittstelle lo ist ebenfalls richtig konfiguriert.

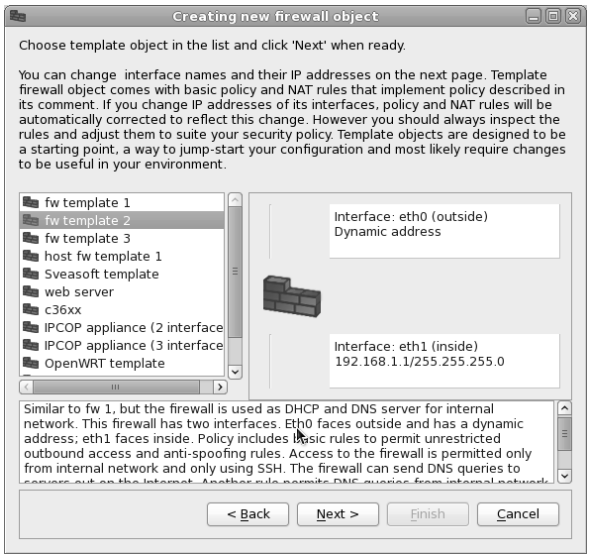


Abbildung 12.4: **Mit den Templates können Sie sich viel Arbeit sparen.**

TSP Bitte Änderung prüfen.

TSQ Bitte bestätigen Sie die Ergänzung dieses Abbildungsverweises.

Editor's or typesetter's annotations (will be removed before the final TeX run)

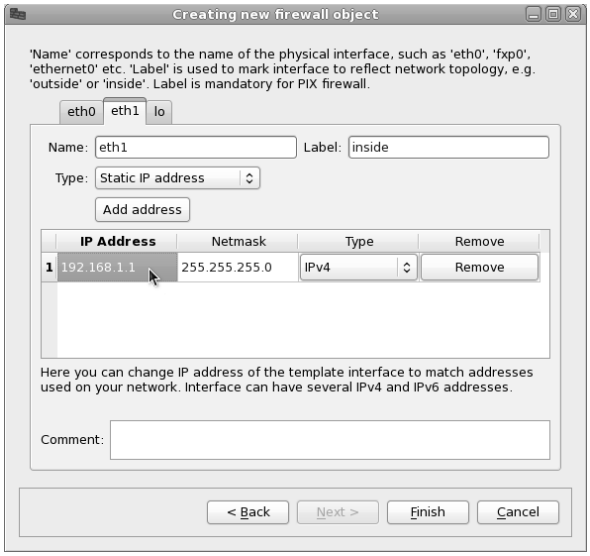


Abbildung 12.5: **Während die externe Netzwerkkarte meist richtig konfiguriert ist, müssen sie die interne Netzwerkkarte anpassen.**

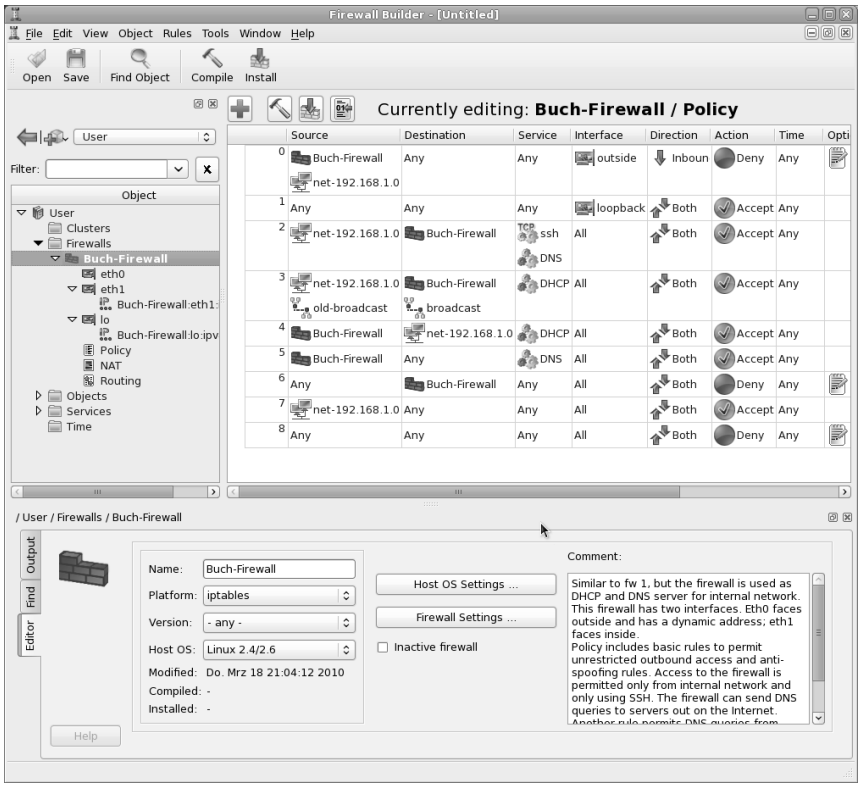


Abbildung 12.6: **Im Template sind bereits eine ganze Reihe Regeln definiert.**

Nun befinden Sie sich wieder in der Hauptansicht des Firewall-Builders. Diese ist dreigeteilt. Während oben rechts bereits eine ganze Reihe von Regeln angezeigt werden, sind im unteren Drittel weitere Anpassungen der Firewall möglich. Interessant sind hier sind die Einstellungen der Iptables-Version und weitere Einstellungen zum Betriebssystem und zur Firewall. Nach Anwahl der HOST OS SETTINGS (siehe Abbildung 12.6TS[9]) können Sie den Kernel über das `/proc`-Interface konfigurieren (siehe Abbildung 12.7). Auf einer reinen IPv4-Firewall sollten Sie die IPv6-Paket-Weiterleitung bewusst abstellen.

Bei den FIREWALL SETTINGS können Sie einstellen, wie die Firewall-Regeln aufgebaut werden sollen. Sie können Einfluss auf die Erzeugung der Regeln, die Installation des Skripts und die Protokollierung nehmen. Dazu können Sie zum Beispiel entscheiden, ob die Firewall auf einer Bridge installiert wird, wie die Pakete abgelehnt werden sollen und ob bei einem Neustart der Firewall alte Verbindungen weiter erlaubt sein sollen (siehe Abbildung 12.8).

Wenn Sie diese Dreiteilung nicht wünschen, können Sie dieses untere Drittel schließen oder in ein eigenes Fenster auslagern. Hierzu dienen die kleinen Icons oben rechts in der Darstellung.

Aktuell erlaubt die Regel 7 sämtlichen Clients aus dem Netz 192.168.1.0/24 den Zugriff auf jeden Dienst im Internet. Um diese Regel zu ändern und den Zugriff auf einen bestimmten Dienst einzuschränken, wählen Sie zunächst eine andere Bibliothek aus. Klicken Sie oben links USER an, und wählen Sie die Standard-Bibliothek aus. Dort wählen Sie den Ordner SERVICES-TCP. Dort wählen Sie dann den Dienst HTTP aus und ziehen ihn in der Regel 7 in die Spalte SERVICE und lassen ihn los (siehe Abbildung 12.9). Um nun eine Regel hinzuzufügen, wählen Sie aus dem Menü RULES die Option ADD NEW RULE BELOW. Ziehen Sie nun aus der Standard-Bibliothek die Gruppe DNS in die Spalte SERVICE der neuen Regel. Nach einem Rechtsklick auf das DENY in der Spalte ACTION wählen Sie als neue Action ACCEPT. Um nun auch

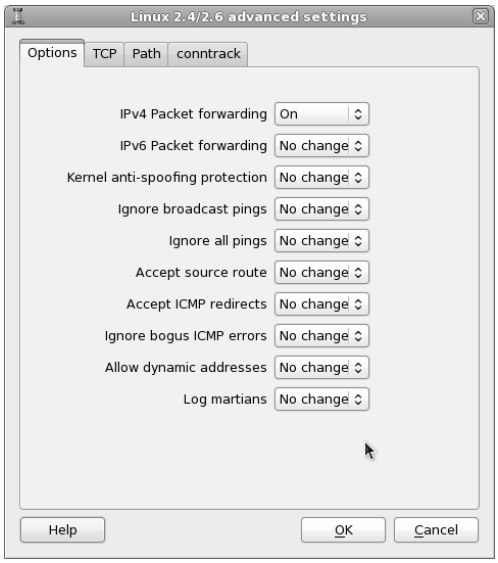


Abbildung 12.7: **Firewall Builder bietet Ihnen auch die Konfiguration der Einstellungen im /proc-Verzeichnis.**

TS[f] Bitte bestätigen.
TS[9] Bitte bestätigen.

Editor's or typesetter's annotations (will be removed before the final TEX run)

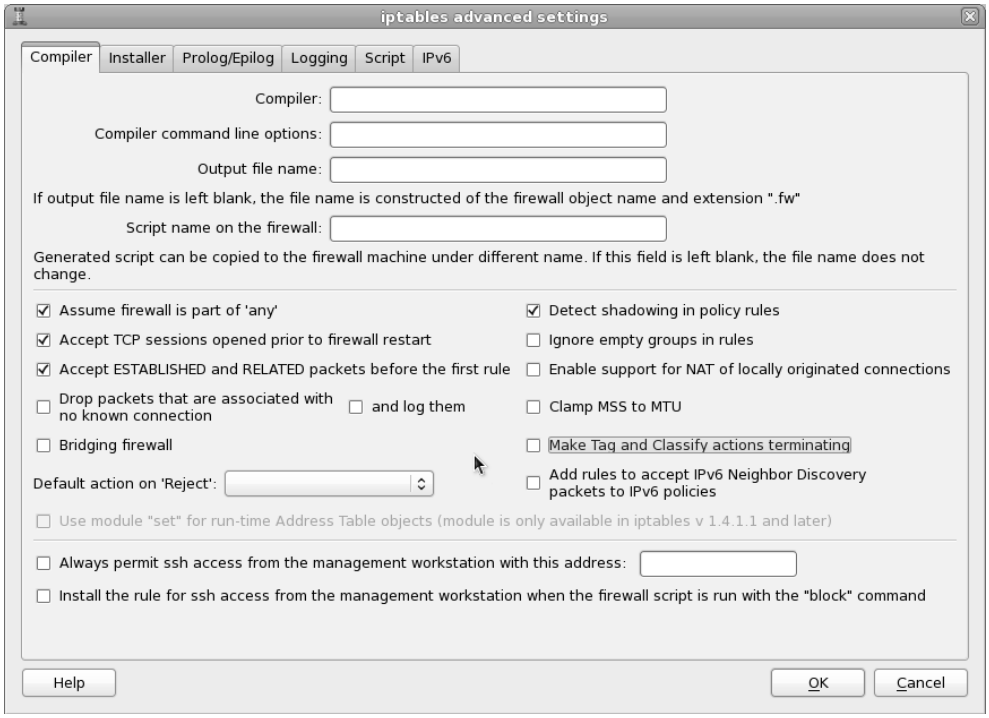


Abbildung 12.8: **Firewall Builder ermöglicht den Zugriff auf jede Einstellung.**

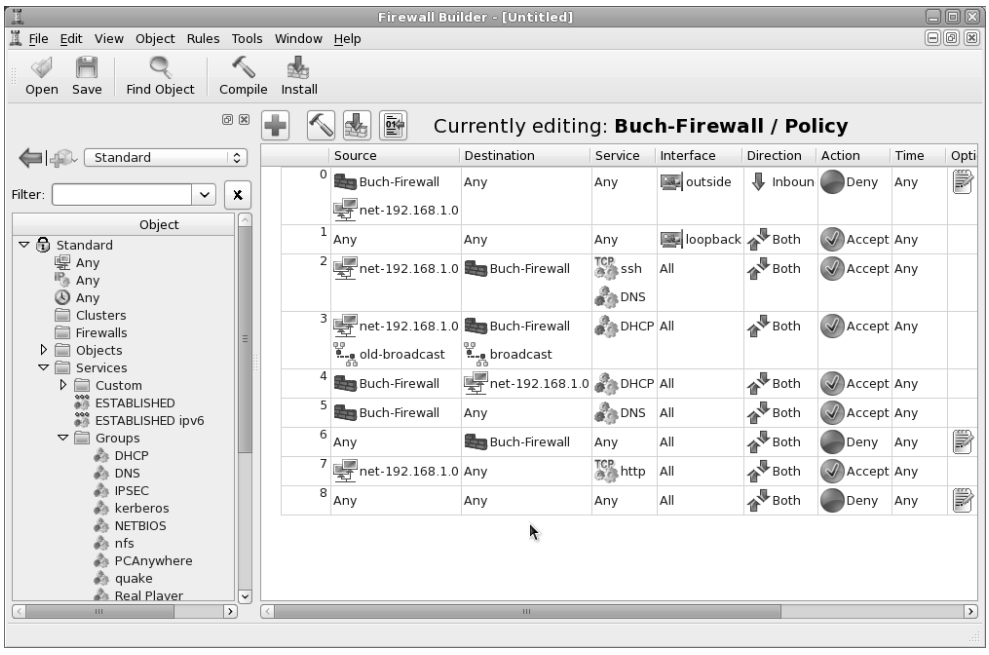


Abbildung 12.9: **Sie können per Drag & Drop einen Dienst einer Regel hinzufügen.**

das Netz als Source zu definieren, wählen Sie das Netz in einer anderen Zeile aus, kopieren es und fügen es mit Paste als Source in dieser Regel wieder ein (siehe Abbildung 12.10).

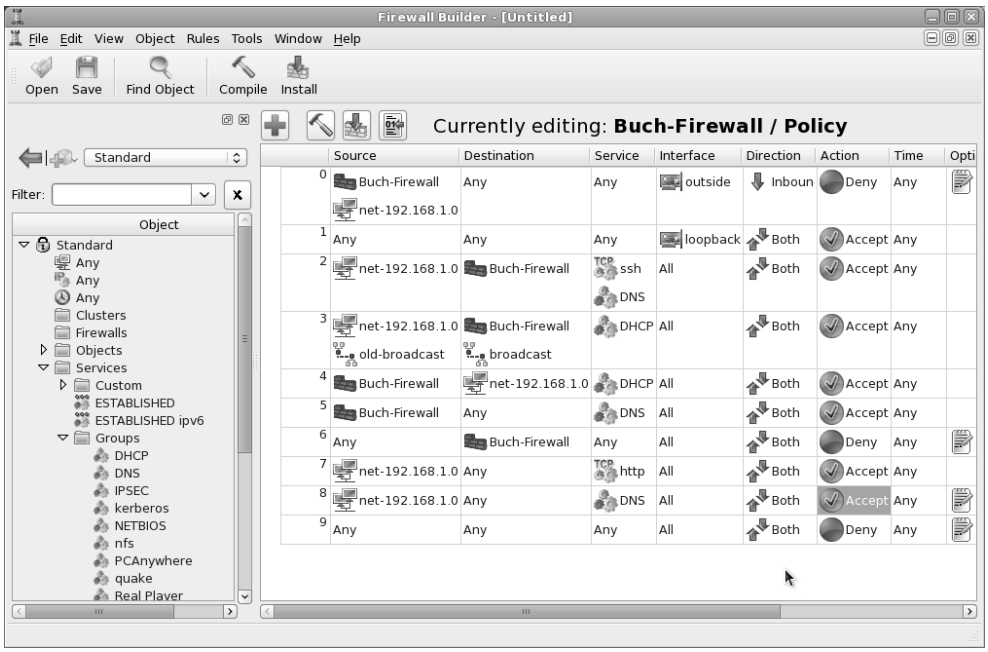


Abbildung 12.10: **Sie können auch sehr einfach eine Regel hinzufügen.**

Abbildung 12.11: **Auch die NAT-Regeln passen Sie mit dem Firewall Builder an.**

INFO

Sie können auch Netzwerkobjekte und Ports in Gruppen zusammenfassen. Diese Administration der Gruppen kann per Drag & Drop erfolgen. So können Sie die beiden Regeln für den Zugriff auf HTTP und DNS auch zusammenfassen.

Auch die Reihenfolge der Regeln können Sie einfach ändern. Dazu aktivieren Sie die Regel mit der rechten Maustaste und wählen Move up oder Move down.

Um die NAT-Regeln zu prüfen und zu bearbeiten, wählen Sie im linken Fenster wieder die User-Bibliothek. Dort finden Sie unter Ihrer Firewall auch die NAT- und Routing-Regeln (siehe Abbildung 12.11). Sobald Sie mit Ihren Regeln zufrieden sind, können Sie die Regeln in die Iptables-Sprache übersetzen lassen. Hierzu wählen Sie im Menü Rules den Unterpunkt Compile (siehe Abbildung 12.12). Nach der Auswahl der zu kompilierenden Firewall[1], werden Ihre Regeln übersetzt.

Wenn Sie die Regeln nicht auf dem Zielsystem erzeugt haben, können Sie diese mit Firewall Builder auch dort installieren. Hierzu wählen Sie im Menü Rules den Unterpunkt Install. In dem nächsten Dialog können Sie den Benutzer und das Kennwort für die Installation angeben. Firewall Builder wird nun mit Secure-Copy die Iptables-Regeln auf dem Zielsystem installieren (siehe Abbildung 12.13). Firewall Builder bietet weitere sehr mächtige Funktionen, die ich hier nicht aufzählen möchte, da der User's Guide sehr ausführlich auf über 400 Seiten die verschiedenen Möglichkeiten darstellt. Ich hoffe, dass ich Ihnen einen einfachen und schnellen Einstieg ermöglicht habe. Firewall Builder ist ein sehr gutes Werkzeug für die Konfiguration einer durchschnittlichen Firewall. Bei komplizierten Setups ziehe ich persönlich immer noch ein Skript vor, da das Regelwerk in Firewall Builder schnell unübersichtlich werden kann. In einem Skript haben Sie die Möglichkeit, selbst Variablen zu definieren und benutzerdefinierte Ketten zu generieren und so Ihre Regeln logisch zu gruppieren. Firewall Builder nutzt

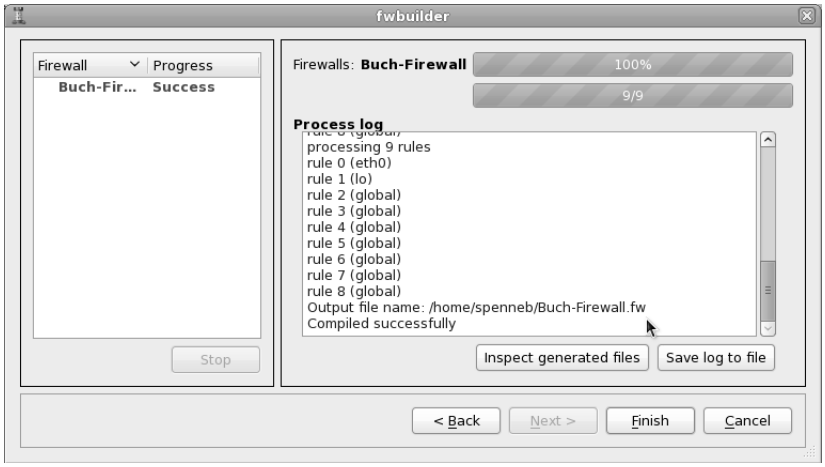


Abbildung 12.12: **Sie können die Übersetzung in die Iptables-Regeln beobachten.**

1 Bedenken Sie, dass Sie mit Firewall-Builder mehrere Firewalls verwalten können!

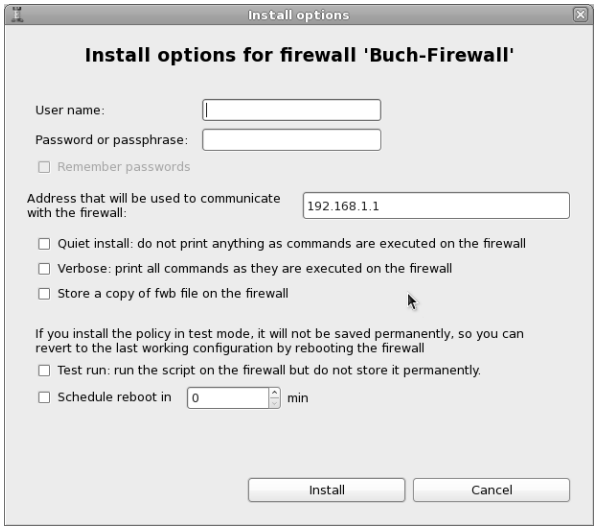


Abbildung 12.13: **Firewall Builder kann auch die Regeln auf dem Zielsystem installieren.**

ebenfalls benutzerdefinierte Ketten, Sie können die Verwendung jedoch nicht steuern. Der Firewall Builder erzeugt für jede kompliziertere grafische Regel eine derartige benutzerdefinierte Kette. Dennoch eignet sich das Werkzeug für mittlere Regelsätze sehr gut, da Sie zusätzliche Kommentare vergeben können und die Verwendung der grafischen Objekte das Fehlen von Variablen mehr als aufwiegt.

12.2 Firestarter

Der Firestarter (`http://www.fs-security.com`) verfolgt eine etwas andere Philosophie als der Firewall Builder. Während der Firewall Builder Ihnen sämtliche Freiheiten bei der Definition Ihrer Firewall lässt und lediglich die Iptables-Details vor Ihnen versteckt, versucht der Firestarter eine möglichst einfache Schnittstelle für die Installation und Konfiguration einer Firewall zu sein. Er ist mit einer Personal-Desktop-Firewall vergleichbar, wie sie auf einigen kommerziellen Betriebssystemen zur Verfügung steht.

STOP

Leider wird der Firestarter nicht mehr aktiv weiterentwickelt. Er ist jedoch in einigen Distributionen enthalten.

Der Firestarter besitzt eine einfache, aber sehr ansprechende grafische Oberfläche (siehe Abbildung 12.14), mit der Sie die Firewall an- oder abschalten, aktuelle Regelverstöße anzeigen und Rechner sperren oder freischalten können.

Auch Firestarter verfügt über ein sehr ausführliches Handbuch, sodass ich mich hier auf einige wenige Hinweise beschränken möchte. Direkt nach dem Start heißt ein Firewall-Assistent Sie herzlich willkommen. Für viele Anwender erfreulich, erkennt Firestarter die konfigurierte Sprache und unterstützt dann auch Deutsch (siehe Abbildung 12.15).

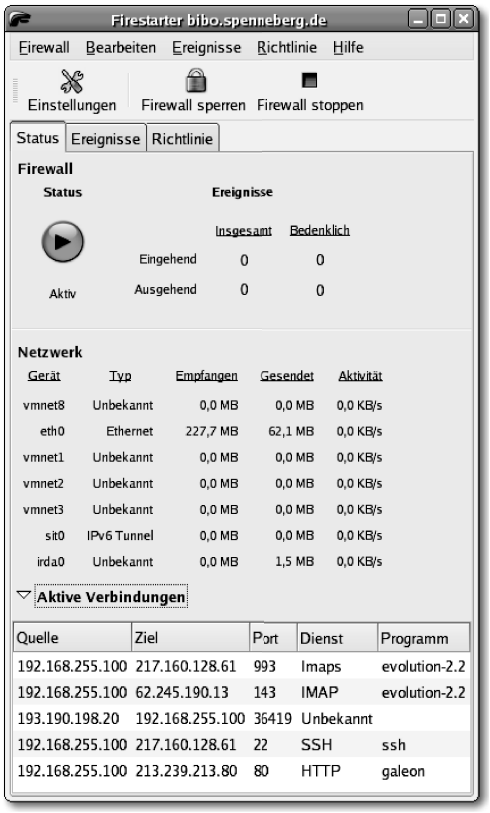


Abbildung 12.14: **Der Firestarter ist eine Personal-Desktop-Firewall.**

Abbildung 12.15: **Der Firestarter unterstützt auch die deutsche Sprache.**

Auf den nächsten Bildschirmen wählen Sie zunächst die Netzwerkkarte aus, die mit dem Internet verbunden ist. Dabei können Sie auch die Unterstützung einer dynamischen IP-Adresse auf dieser Netzwerkkarte aktivieren. Anschließend werden Sie gefragt, ob Sie die

Abbildung 12.16: **Firestarter kann auch als Firewall für ein Netzwerk eingesetzt werden.**

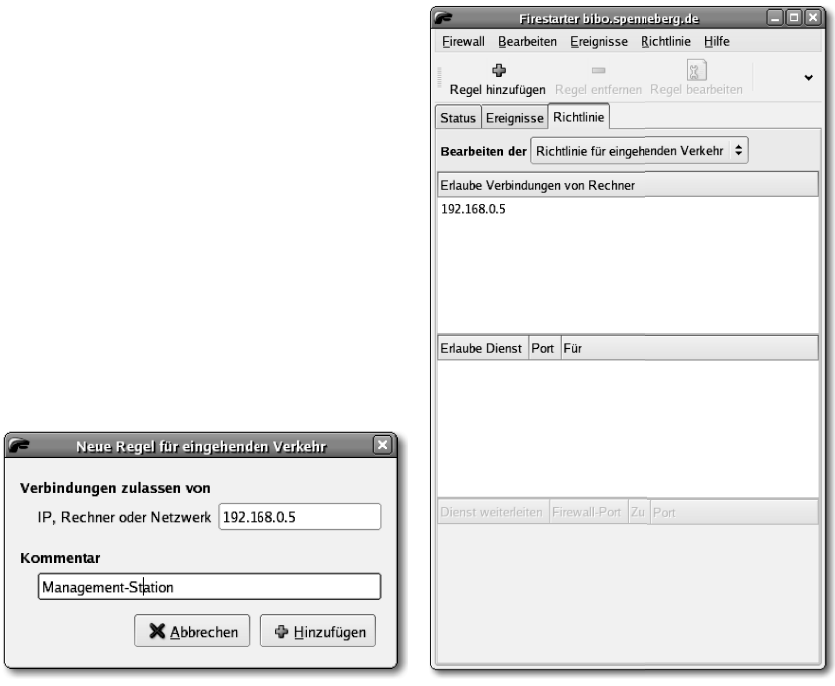


Abbildung 12.17: **Sie können auch Richtlinien hinzufügen.**

Internetverbindung mit anderen Benutzern teilen möchten. Wenn Ihr System über zwei Netzwerkkarten verfügt und über eine Netzwerkkarte mit dem Internet und über die andere Netzwerkkarte mit einem lokalen Netz verbunden ist, können Sie hier für das lokale Netz die Internetverbindung freischalten. Eine besondere Funktion von Firestarter ist die Möglichkeit, hier direkt einen DHCP-Server für das interne Netz zu aktivieren (siehe Abbildung 12.16). Dafür muss jedoch auf Ihrem System ein DHCP-Server installiert sein. Auf dem letzten Bildschirm müssen Sie die Firewall nur noch starten. Achtung: Falls Sie mit dem Firewall-Rechner gerade über das Netzwerk verbunden sind und Firestarter über das Netzwerk gestartet haben, sollten Sie die Firewall noch nicht starten, sondern erst eine Richtlinie erzeugen, die diesen Zugriff erlaubt.

Jetzt befinden Sie sich in dem Hauptfenster von Firestarter. Sie können nun die Firewall starten oder stoppen, die Einstellungen überprüfen, Ereignisse beobachten und Richtlinien hinzufügen (siehe Abbildung 12.17). Sobald Regelverletzungen stattfinden, erkennen Sie dies auf der Statusseite. Dort werden die Ereignisse gezählt (siehe Abbildung 12.18). Wenn Sie dann auf die Registerkarte Ereignisse wechseln, können Sie die Regelverletzungen tatsächlich studieren.

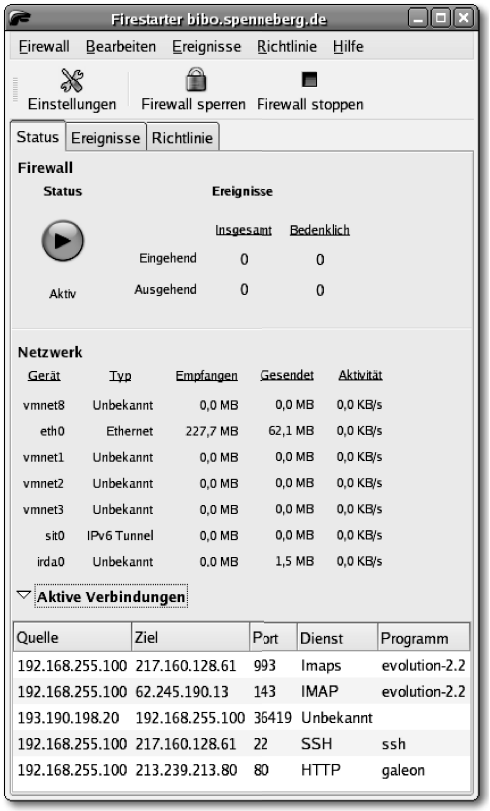


Abbildung 12.18: **Auf der Status-Seite erhalten Sie einen Überblick über eingehende und ausgehende Ereignisse.**

Bitte Verweis auf Abbildung 12.19 im Text einfügen.

Editor's or typesetter's annotations (will be removed before the final TEX run)

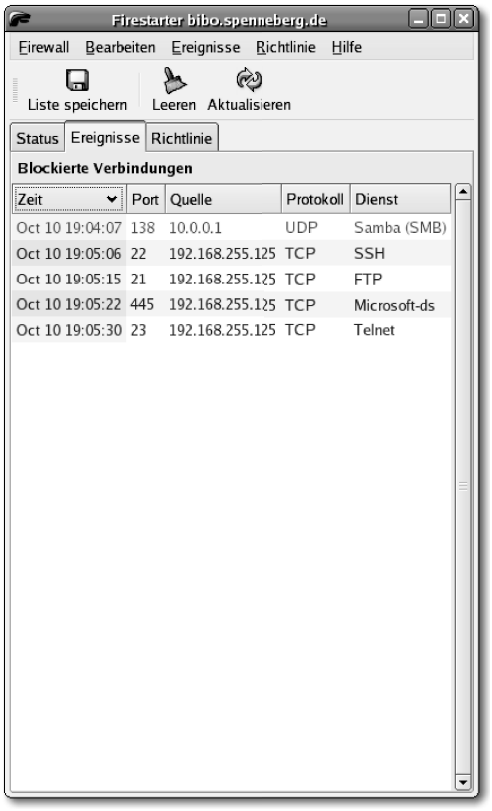


Abbildung 12.19: **Die Ereignisse im Detail zeigt Firestarter nach Auswahl der entsprechenden Registerkarte an.**

Durch einen Rechtsklick auf eines der Ereignisse können Sie wählen, ob Sie alle weiteren Verbindungen von diesem Rechner ablehnen oder akzeptieren möchten, diesen Dienst freischalten möchten oder grundsätzlich diese Ereignisse nicht mehr protokollieren möchten.TS

Wenn Sie das Firestarter-Paket als RPM- oder Debian-Paket installiert haben, läuft die Firewall sogar nach einem Neustart des Systems. Um dies zu erreichen, wurde Firestarter als Systemdienst eingebunden und wird über ein Init-Skript automatisch bei einem Neustart gestartet. Die grafische Oberfläche benötigen Sie dann nur für die Konfiguration und Überwachung. Wenn Ihre Firewall über keinen X-Server verfügt, können Sie das grafische Werkzeug auch über das Netzwerk aufrufen!

Um Firestarter zu installieren, gibt es mehrere Möglichkeiten. Am einfachsten installieren Sie Firestarter als RPM- oder Debian-Paket. Auf der Firestarter-Homepage finden Sie Pakete für alle aktuellen Distributionen inklusive Ubuntu und Gentoo.

Für kleine Installationen sind die Möglichkeiten, die Firestarter Ihnen bietet, häufig vollkommen ausreichend. Die Konfiguration komplizierter Regelsätze ist mit Firestarter nicht möglich, da Sie nicht selbst die Iptables-Regeln modifizieren oder anpassen können.

CEU Bitte im ganzen Abschnitt vereinheitlichen: "*Die* Shorewall" wie "*die* Firewall" oder als Name einer Software – vgl. "Mozilla Firefox" nicht "*ein* Mozilla Firefox".

12.3 Shorewall (Shoreline Firewall)

Die Shoreline Firewall (Shorewall) ist eine sehr komplexe Firewall, die Ihre Konfiguration aus Skripten liest und mit dem Iptables-Befehl in Firewall-Regeln umsetzt. Ihre Anwendung wird nicht durch eine grafische Oberfläche unterstützt. Es gibt jedoch für Webmin (`http://www.webmin.com`) ein leider veraltetes Shorewall-Modul. Dennoch verwenden viele Anwender gern CE U Shorewall, da es sehr flexibel den Aufbau von komplizierten Firewalls auf der Distribution ihrer Wahl ermöglicht.

Shorewall wird im Moment im Wesentlichen von einem einzigen Entwickler gepflegt: Tom Eastep. Daher sollten Sie sich natürlich speziell bei einer Firewall überlegen, ob Sie sich auf die Leistung eines einzigen Entwicklers verlassen möchten. Sollte der Entwickler seine Arbeit einstellen, wissen Sie nicht, ob das Projekt weitergepflegt wird.

Die Installation von Shorewall ist sehr einfach, da Sie auf der Homepage `http://www.shorewall.net` Pakete für sämtliche aktuellen Distributionen finden. Teilweise ist Shorewall bereits in den Distributionen integriert.

12.3.1 Das Shorewall-Konzept

Die gesamte Konfiguration befindet sich in dem Verzeichnis `/etc/shorewall`. Die wichtigste Datei ist die Datei `zones`, in der die Firewall-Zonen definiert werden. Eine typische `zones`-Datei hat den folgenden Inhalt:

```
#ZONE   DISPLAY     COMMENTS
 net    Internet    The big bad Internet
 loc    Local       Local Network
 dmz    DMZ         Demilitarized zone.
```

Diese Firewall verfügt über drei Netzwerkkarten. Per Default gibt es immer noch zusätzlich eine weitere Zone für die Firewall selbst: `fw`. Die zweite Datei, die Sie betrachten sollten, ist die Datei `policy`. Hier definieren Sie die Default-Policies für den Verkehr von einer in die andere Zone. Es gibt sechs verschiedene Policies, die Sie definieren können: `ACCEPT`, `DROP`, `REJECT`, `QUEUE`, `CONTINUE` oder `NONE`. Eine typische `policy`-Datei sieht so aus:

```
#SOURCE DEST POLICY  LOGLEVEL LIMIT:BURST#
 loc    net  ACCEPT
 net    all  DROP    info

# THE FOLLOWING POLICY MUST BE LAST
 all    all  REJECT  info
```

Über den Log-Level definieren Sie, ob und wie die zutreffenden Pakete protokolliert werden sollen. Falls Sie den Ulog-Daemon einsetzen möchten, geben Sie hier `ULOG` an. In der letzten Spalte können Sie die maximale Rate der TCP-Verbindungen und den erlaubten Burst angeben (siehe Abschnitt 16.9.8 TS V).

TS V Bitte Abschnittsverweis überprüfen.

Editor's or typesetter's annotations (will be removed before the final TEX run)

Die Ausnahmen dieser Default-Policies werden dann in der Datei `rules` definiert. Sie müssen hier nur Regeln definieren, die den Verbindungsaufbau erlauben oder ablehnen. Die Shorewall kümmert sich dann selbst um alle weiteren Pakete, die zu der aufgebauten Verbindung gehören (State: ESTABLISHED) oder mit ihr verwandt sind (State: RELATED). Um zum Beispiel aus dem internen Netz mit der Secure Shell auf die Firewall zuzugreifen und die Firewall per HTTPS Updates aus dem Internet (z.B. Red Hat Network) herunterladen zu lassen, müssen Sie die folgende Regel in der Datei `rules` einfügen:

```
#ACTION SOURCE DEST PROTO DEST SOURCE ORIGINAL RATE USER/ PORT PORT(S)  ↵
        DEST LIMIT GROUP
 ACCEPT loc    fw   tcp   22
 ACCEPT fw     net  tcp   53
 ACCEPT fw     net  udp   53
 ACCEPT fw     net  tcp   443
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Wenn es Sie stört, dass Sie zwei Zeilen für den DNS-Verkehr in der Datei `rules` definieren müssen, können Sie auch das mächtige Action-Konzept von Shorewall nutzen. Die Actions können Sie selbst definieren, aber Shorewall verfügt auch bereits über eine Reihe von vorkonfigurierten Actions in `/usr/share/shorewall/`. Die Action `AllowDNS` wurde so definiert:

```
#
# Shorewall version 2.4 - AllowDNS Action
#
# /usr/share/shorewall/action.AllowDNS
#
#           This action accepts DNS traffic.
#
###############################################################################
# TARGET SOURCE DEST PROTO DEST SOURCE ORIGINAL RATE USER/ PORT PORT(S)  ↵
        DEST LIMIT GROUP
#  ACCEPT -      -    udp   53
#  ACCEPT -      -    tcp   53
# LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Das bedeutet, dass Sie auch die folgende Einstellung in der Datei `rules` nutzen können, die das gleiche Ergebnis erzeugt:

```
# ACTION   SOURCE DEST PROTO DEST SOURCE ORIGINAL RATE USER/ PORT PORT(S) ↵
         DEST LIMIT GROUP
#  ACCEPT   loc    fw   tcp   22
#  AllowDNS fw     net
#  ACCEPT   fw     net  tcp   443
# LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Genauso können Sie auch die erste Zeile durch eine `AllowSSH`-Zeile ersetzen, da auch für SSH eine vorkonfigurierte Action vorhanden ist.

Nun müssen Sie noch die Zonen den verschiedenen Netzwerkkarten zuordnen. Dies erfolgt in der Datei `interfaces`.

```
#ZONE INTERFACE BROADCAST OPTIONS
 net  eth0      detect    dhcp,routefilter,norfc1918
 loc  eth1      detect
 dmz  eth2      detect
```

Schließlich fehlt noch die Definition des NAT. Hierzu verwendet Shorewall die Datei `masq`. Um einfach ein Masquerading zu definieren, tragen Sie hier die folgende Zeile ein:

```
#INTERFACE SUBNET ADDRESS PROTO PORT(S) IPSEC
eth0       eth1
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Diese Zeile maskiert jede über `eth0` ausgehende Verbindung aus dem Subnetz, das über die Netzwerkkarte `eth1` angebunden ist. Alternativ können Sie für `eth1` auch das Subnetz in Form von `192.168.0.0/24` angeben.

Nun müssen Sie nur noch Shorewall starten. Seit der Version 1.3.9 wird Shorewall automatisch zum Boot-Zeitpunkt des Rechners über ein Init-Skript gestartet. Ob dies bei Ihnen auch der Fall ist, können Sie mit dem Befehl `chkconfig` überprüfen:

```
# chkconfig shorewall --list
shorewall 0:Aus 1:Aus 2:Ein 3:Ein 4:Ein 5:Ein 6:Aus
```

Damit jedoch nicht eine fehlerhafte oder fehlende Konfiguration den Zugriff auf den Rechner verhindert, ist der tatsächliche Start noch deaktiviert. Bei den aktuellen Distributionen wird der Start über die Datei `/etc/shorewall/shorewall.conf` gesteuert. Diese Datei enthält zu Beginn die Variable `STARTUP_ENABLED=No`. Setzen Sie diese Variable auf `Yes`, und Sie können Shorewall starten. Hierfür verwenden Sie den Befehl `shorewall`. Dieser Befehl hat die folgenden Optionen:

- `help`: Zeigt die Hilfe an.
- `start`: Dies startet Shorewall.
- `stop`: Dies stoppt Shorewall. Dabei werden das Forwarding und das Routing deaktiviert. Wenn Sie diesen Befehl aus der Ferne durchführen, sollte Ihre IP-Adresse in der Datei `routestopped` enthalten sein, damit Shorewall anschließend noch die Kommunikation mit Ihrer IP-Adresse erlaubt.
- `clear`: Entfernt sämtliche Shorewall-Regeln und -Policies. Der Rechner ist anschließend von außen erreichbar.
- `restart`: Führt einen Neustart von Shorewall durch.
- `save`: Speichert die aktuelle Konfiguration in einer Backup-Datei für eine spätere Wiederherstellung ab.

» `restore`: Stellt die Konfiguration wieder her.

» `forget`: Entfernt die Backup-Datei.

» `check`: Prüft die Konfiguration.

» `try <Conf> [<timeout>]`: Führt einen Neustart von Shorewall mit dem angegebenen Konfigurationsverzeichnis durch. Allerdings wird, sobald ein Fehler auftritt, erneut ein Neustart mit der Default-Konfiguration in `/etc/shorewall` durchgeführt. Wenn Sie einen Timeout angeben, dann wird der Neustart immer nach Ablauf des Timeouts durchgeführt. Damit können Sie auch aus der Ferne eine neue Konfiguration testen, ohne befürchten zu müssen, sich dauerhaft auszuschließen.

Shorewall unterstützt die Verwendung von alternativen Konfigurationsverzeichnissen. Um diese mächtige Funktion zu nutzen, erzeugen Sie einfach ein neues Verzeichnis `/etc/test` und kopieren die Dateien, die Sie ändern möchten, in dieses Verzeichnis. Nach ihrer Modifikation können Sie zunächst die Konfiguration auf ihre syntaktische Korrektheit prüfen: `shorewall check /etc/test`. In dem Verzeichnis `/etc/test` fehlende Dateien werden automatisch aus dem Verzeichnis `/etc/shorewall` geladen. Nachdem Sie alle Fehler korrigiert haben, können Sie mit `shorewall try /etc/test 30` die neue Konfiguration prüfen. Sollten Sie sich bei diesem Vorgang selbst ausgeschlossen haben, müssen Sie nur 30 Sekunden warten. Dann wird die Shorewall sich selbst wieder zurücksetzen! Ist alles zu Ihrer Zufriedenheit, so kopieren Sie die modifizierten Dateien in das Verzeichnis `/etc/shorewall`, und die Firewall wird nun nach einem Neustart die aktuelle Konfiguration verwenden.

12.3.2 Die Shorewall-Dateien

Dieser Abschnitt soll Ihnen kurz einen Eindruck von den von Shorewall verwendeten Dateien und ihren Funktionen vermitteln. Diese Information wird sicherlich nicht ausreichen, damit Sie die Dateien in Ihrer ganzen Fülle verstehen und administrieren können (dafür ist das Shorewall-Handbuch da), aber Sie erhalten so einen Überblick über die Möglichkeiten.

accounting

In dieser Datei können Sie Accounting-Regeln angeben. Dies sind Zählregeln, die lediglich bestimmte Pakete zählen, um zum Beispiel den Anteil des HTTP-Verkehrs oder die Anzahl der SMTP-Verbindungen zu zählen.

actions

Hier können Sie Ihre eigenen Actions definieren. Eine Action ist eine Art Makro, mit dem Sie komplizierte Regelsätze vereinfachen können. Beispiele für vordefinierte Shorewall-Actions finden Sie in `/usr/share/shorewall/`.

blacklist

Sie können in der Datei `interfaces` bei den verschiedenen Netzwerkkarten die Option `blacklist` mit angeben. Dann wird diese Datei für die entsprechende Netzwerkkarte ausgewertet.

Jede IP-, Netz- oder MAC-Adresse in dieser Datei wird dann automatisch entsprechend der Variablen `BLACKLIST_DISPOSITION` und `BLACKLIST_LOGLEVEL` in der Datei `shorewall.conf` abgewiesen.

continue

In dieser Datei können Sie Befehle eintragen, die nach dem Aufruf des Befehls `shorewall clear` aufgerufen werden sollen. Damit können Sie zum Beispiel nach dem Löschen sämtlicher Regeln doch wieder bestimmte Grundregeln für die Grundsicherheit des Systems aktivieren.

ecn

Die Explicit Congestion Notification (ECN) ist eine sehr nützliche Methode, die eine Verstopfung der Netzwerkverbindung frühzeitig erkennen und verhindern kann. Da ECN jedoch bisher reservierte Bits in den IP- und TCP-Headern benutzt, blockieren viele ältere und kommerzielle Firewalls die Pakete, die ECN nutzen. Wenn Sie dennoch ECN nutzen wollen, können Sie in dieser Datei die Rechner angeben, für die Shorewall ECN deaktivieren soll.

hosts

In einigen Konstellationen ist es sinnvoll, die Rechner, die über ein und dieselbe Netzwerkkarte mit Shorewall verbunden sind, in unterschiedliche Zonen einzuteilen. Damit können Sie die Regeln für die einzelnen Rechner noch feiner konfigurieren und zum Beispiel eine Zone `dmzdns` und eine Zone `dmzproxy` definieren. Die Zonen müssen Sie zunächst in der Datei `zones` angeben. Dann ordnen Sie nicht den Zonen in der Datei `interfaces` Netzwerkkarten zu, sondern geben in dieser Datei Subnetze und Rechner an.

init, initdone, start, started, stop und stopped

Hier können Sie Befehle definieren, die vor dem Start von `shorewall start` (`init`) oder nach der Initialisierung des Befehls und vor dem Hinzufügen der Regeln (`initdone`) ausgeführt werden sollen.

Die Dateien `start` und `started` werden nach dem Start der Shorewall ausgeführt. Dabei wird die Datei `started` erst nach dem vollständigen Start der Shorewall aufgerufen (siehe Dokumentation).

Die Datei `stop` wird zu Beginn des Befehls `shorewall stop` ausgeführt, und die Datei `stopped` wird anschließend ausgeführt.

Über diese Dateien können Sie die Shorewall mit eigenen Skripten beliebig erweitern.

interfaces

In dieser Datei ordnen Sie die Zonen verschiedenen Netzwerkkarten zu.

CE W Bitte prüfen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

ipsec

In dieser Datei können Sie Regeln für IPsec-Verbindungen hinterlegen. Grundsätzlich können Sie hier zum Beispiel Regeln für die Maximum Segment Size (MSS) setzen. Wenn Sie zusätzliche Regeln setzen möchten, die den IPsec-Verkehr erlauben oder verbieten, müssen Ihr Kernel und Ihr Iptables-Befehl über den Policy-Match (siehe Abschnitt 16.10.4 ) verfügen.

maclist

Mit dieser Datei können Sie Regeln definieren, die nur bestimmten MAC-Adressen die Kommunikation erlauben. Damit können Sie einen MAC-Filter erzeugen, wie er häufig auch für WLAN-Anwendungen verwendet wird.

masq

In dieser Datei definieren Sie das Masquerading und Source-NAT. Dabei haben Sie alle Freiheiten, die der Iptables-Befehl Ihnen auch gibt.

modules

Diese Datei lädt alle zum Start von Shorewall benötigten Module.

nat

Mit dieser Datei können Sie 1:1-NAT-Regeln definieren. Achten Sie darauf, dass Sie bei Shorewall auch ProxyARP verwenden können. In einigen Fällen ist ProxyARP vorzuziehen. Auch ein einfaches Port-Forwarding wird nicht in dieser Datei, sondern in der Datei `rules` definiert.

netmap

Mit dieser Datei können Sie das Iptables-`NETMAP`-Target (siehe Abschnitt 20.5) nutzen. Dieses ermöglicht es Ihnen, sehr einfach jede IP-Adresse aus einem Netz einer IP-Adresse aus einem anderen Netz zuzuordnen.

params

Hier können Sie Variablen definieren, die Sie an anderen Stellen und in anderen Dateien wiederverwenden. Dazu definieren und nutzen Sie die Variablen in einfacher Bourne-Shell-Syntax.

policy

Diese Datei definiert die Default-Policies für die Verbindungen zwischen den verschiedenen Zonen. Alle Ausnahmen dieser Default-Policies werden dann in der Datei `rules` definiert.

providers

Hier können Sie zusätzliche Routing-Tabellen definieren. Dazu verwendet Shorewall die Advanced-Routing-Funktionen des Kernels. Damit können Sie zum Beispiel Szenarien abdecken, bei denen Sie mehrere Verbindungen zum Internet besitzen und diese gleichzeitig nutzen möchten.

proxyarp

In dieser Datei können Sie Proxy-ARP definieren (siehe auch Kapitel 29).

routes

In dieser Datei können Sie Firewall-Regeln definieren, die dazu führen, dass einzelne Verbindungen anders geroutet werden. Ihr Kernel und Ihr Iptables-Befehl müssen dafür das `ROUTE`-Target unterstützen. Dies ist jedoch nur selten der Fall.

routestopped

Wenn Sie Shorewall stoppen, ist keine Kommunikation mehr möglich. Wenn Sie dennoch, zum Beispiel auf einer Bridge, die Kommunikation erlauben möchten, können Sie in dieser Datei Regeln angeben, die bei einer gestoppten Firewall dennoch Verbindungen akzeptieren.

rules

Diese Datei enthält den eigentlichen Kern des Shorewall-Regelwerkes. Hier definieren Sie Ihre Firewall-Regeln in der abstrakten Shorewall-Syntax. Dabei können Sie auch selbst definierte Actions aus der Datei `actions` verwenden.

shorewall.conf

Diese Datei steuert zentral das Verhalten von Shorewall. Hier können Sie zum Beispiel konfigurieren, ob Shorewall überhaupt startet!

tcrules

Häufig soll eine Firewall auch direkt die zur Verfügung stehende Bandbreite der Internetverbindung regulieren. Dazu unterstützt Shorewall die Quality-of-Service-Funktionen des Linux-Kernels. Hier können Sie Regeln definieren, die Pakete oder Verbindungen markieren und klassifizieren.

tos

Wenn Ihre Infrastruktur noch die Type-of-Service-Bits (TOS) in dem IP-Header unterstützt und Sie diese nutzen möchten, um bestimmte Verbindungen zu priorisieren, können Sie in dieser Datei Regeln definieren, die die TOS-Bits entsprechend setzen (siehe auch Abschnitt 21.8).

tunnels

Eine Firewall ist häufig auch der Endpunkt von VPN-Tunneln. Diese Tunnel können Sie in dieser Datei definieren. Die Shorewall wird dann diese Tunnel erlauben. Natürlich müssen Sie zusätzlich zum Beispiel Racoon oder OpenVPN konfigurieren, damit die Tunnel auch aufgebaut werden.

zones

In dieser Datei definieren Sie die Zonen, die von der Firewall überwacht werden sollen. Für die Anzeige können Sie hier jeder Zone einen aussagekräftigen Namen zuweisen.

12.3.3 Weitere Shorewall-Eigenschaften

Shorewall unterstützt im Weiteren folgende Funktionen, auf die ich im Rahmen dieses Buches nicht eingehen kann. Diese Funktionen werden aber in der mitgelieferten Dokumentation ausführlich beschrieben und mit Beispielen dokumentiert:

» Tunnel
  - IPv6- in IPv4-Tunnel
  - PPTP-Tunnel
  - IPv4- in IPv4-Tunnel
  - IPsec-Tunnel mit dem Linux-Kernel 2.4 und 2.6
  - OpenVPN-Tunnel

» Bandbreitenbegrenzung

» Firewall im Bridge-Modus

» mehrere Internetzugänge

» Blacklisting von IP-Adressen

Wenn Sie eine mächtige und flexible Oberfläche suchen, die aber ohne Maus zu bedienen ist, einfach aus der Ferne gewartet werden kann und ein einfaches Backup ermöglicht, ist Shorewall ein interessanter Kandidat.

13. Distributionswerkzeuge

Viele Distributionen enthalten auch Werkzeuge für die Verwaltung von Firewalls. In diesem Kapitel werde ich diese Werkzeuge und einige spezielle Firewall-Distributionen vorstellen und deren Besonderheiten erläutern.

13.1 Fedora

Fedora und Red Hat Enterprise Linux bieten nur eine rudimentäre Firewall-Funktion. Nachdem lange Zeit mit `lokkit` nur ein Curses-basiertes Werkzeug (siehe Abbildung 13.1) zur Verfügung stand, gibt es in den aktuellen Distributionen mit `system-config-firewall`[1] ein einfaches grafisches Werkzeug. Dieses Werkzeug bietet aber immer noch keine besondere Intelligenz oder Komfort (siehe Abbildung 13.2).

Für die Konfiguration einer lokalen Firewall ist das aktuelle Werkzeug durchaus einsetzbar. Für eine komplexe Firewall mit mehreren Netzwerkkarten ist es jedoch unbrauchbar. Hier sollten Sie auf den Firewall-Builder oder Shorewall zurückgreifen, wenn Sie nicht Ihre eigenen Skripte schreiben wollen.

13.2 OpenSUSE

SUSE/Novell liefert seit der Version SuSE 8.0 die SuSEfirewall2 aus. Dabei handelt es sich im Wesentlichen um die Konfigurationsdatei `/etc/sysconfig/SuSEfirewall2` und um eine Reihe von Bash-Skripten, die diese Konfigurationsdatei lesen und in Regeln umsetzen. Die SuSEfirewall2 unterstützt eine Firewall mit drei Zonen: innen (INT), außen (EXT) und entmilitarisiert (DMZ). Die Konfiguration kann sowohl über die Konfigurationsdatei direkt als auch über Yast erfolgen. Ich werde hier die Konfiguration mit dem grafischen Werkzeug Yast der Version OpenSUSE 10 erläutern. Dabei werde ich aber auch jeweils die entsprechenden Variablen in der Konfigurationsdatei ansprechen, die von Yast bei dem entsprechenden Dialog geändert werden.

Da die gesamte Konfigurationsdatei inklusive der Kommentare fast 1000 Zeilen lang ist, werde ich mich hier auf die wesentlichen Punkte beschränken. Auf `http://susefaq.sourceforge.net/articles/firewall/fw_manual.html` finden Sie den Versuch einer Dokumentation der SuSEfirewall2 in einem 110 Seiten starken PDF-Dokument.

1 Bei älteren Distributionen heißt dieses Werkzeug `system-config-securitylevel`.

Abbildung 13.1: **Das Werkzeug lokkit ist weder schön noch komfortabel. Es ist heute ein textbasiertes Interface des Befehls system-config-securitylevel.**

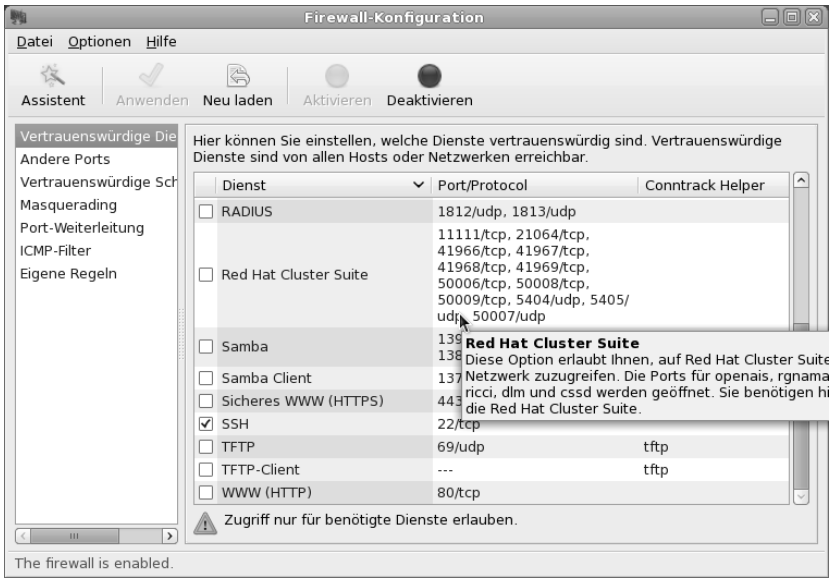


Abbildung 13.2: **Der grafische Befehl system-config-firewall erlaubt die grafische Konfiguration der Firewall.**

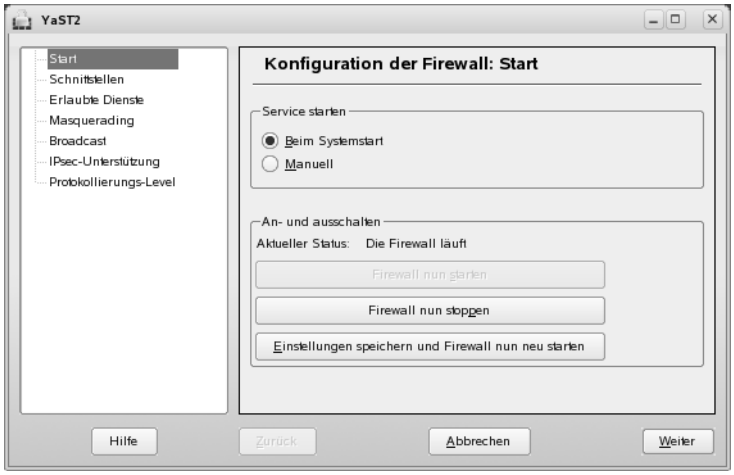


Abbildung 13.3: **Sie können mit diesem Dialog die Firewall stoppen oder neu starten.**

Sie starten zunächst `yast2` und wählen dort das Menü SICHERHEIT aus. Dort finden Sie ein Modul FIREWALL. Nach dem Anklicken öffnet sich das Yast2-Firewall-Modul (siehe Abbildung 13.3). Dort können Sie auswählen, ob die SuSEfirewall2 automatisch bei jedem Systemstart oder nur auf manuelle Aufforderung gestartet werden soll. Außerdem erkennen Sie, ob aktuell die Firewall aktiv ist oder nicht.

Ein Neustart der Firewall kann übrigens auch auf der Kommandozeile erfolgen. Hierfür rufen Sie einfach `rcSuSEfirewall2 restart` auf. Bevor Sie die Firewall konfigurieren können, müssen Sie die Netzwerkkarten festlegen. Hierfür wählen Sie im linken Baum den Punkt SCHNITTSTELLEN aus (siehe Abbildung 13.4). Hier legen Sie fest, welche Karte mit welcher Zone verbunden ist. Die SuSEfirewall2 unterscheidet drei Zonen:

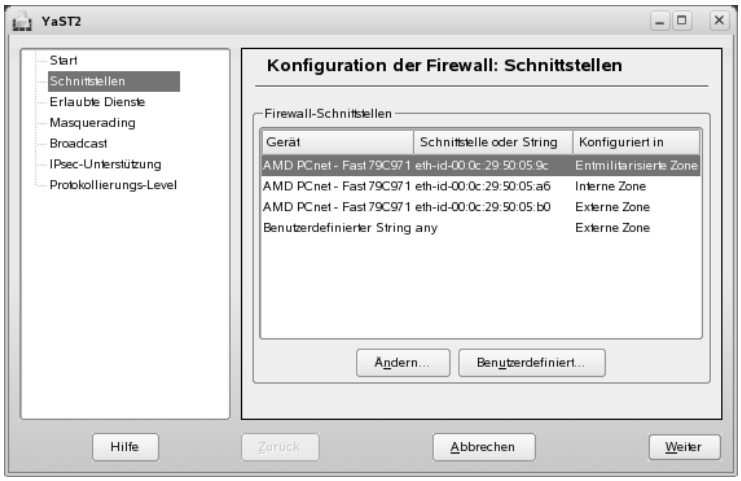


Abbildung 13.4: **Weisen Sie jeder Netzwerkkarte die richtige Zone zu.**

interne Zone. Diese darf ungehindert auf die externe Zone zugreifen.

externe Zone. Diese Zone hat keinerlei Rechte, außer wenn ein Forwarding definiert ist (siehe weiter unten).

entmilitarisierte Zone. Der Zugriff auf die externe Zone ist erlaubt. Ein Zugriff aus der externen Zone in die entmilitarisierte Zone ist nach Konfiguration auch erlaubt.

Wenn Sie die Einstellungen in der Konfigurationsdatei nachvollziehen möchten oder manuell durchführen wollen, müssen Sie die folgenden Variablen editieren:

```
FW_DEV_EXT="any eth-id-00:0c:29:50:05:b0"
FW_DEV_INT="eth-id-00:0c:29:50:05:a6"
FW_DEV_DMZ="eth-id-00:0c:29:50:05:9c"
```

Lassen Sie sich nicht von den ungewöhnlichen Bezeichnungen der Netzwerkkarten irritieren. SuSE verwendet statt `eth0` die Bezeichnung `eth-id-<MAC-Adresse>`.

Sie sollten jede Netzwerkkarte einer Zone zuordnen. Wenn Sie dies für eine Netzwerkkarte versäumen, darf diese keine Pakete versenden oder empfangen.

Nun müssen Sie im nächsten Dialog (siehe Abbildung 13.5) die erlaubten Dienste definieren. Es handelt sich hierbei um Dienste auf der Firewall, auf die Sie aus den verschiedenen Zonen zugreifen. Es handelt sich nicht um einen Zugriff auf einen Dienst hinter der Firewall!

Wählen Sie nun jede Zone aus, und konfigurieren Sie die Dienste, auf die ein Zugriff erlaubt sein soll. Bei der internen Zone handelt es sich um einen Sonderfall. Da dies eine vertraute (*Trusted*) Zone ist, darf diese zunächst auf alle Dienste der Firewall zugreifen. Es ist jedoch sinnvoll, die Firewall auch vor der internen Zone zu schützen und nur bestimmte Dienste freizuschalten.

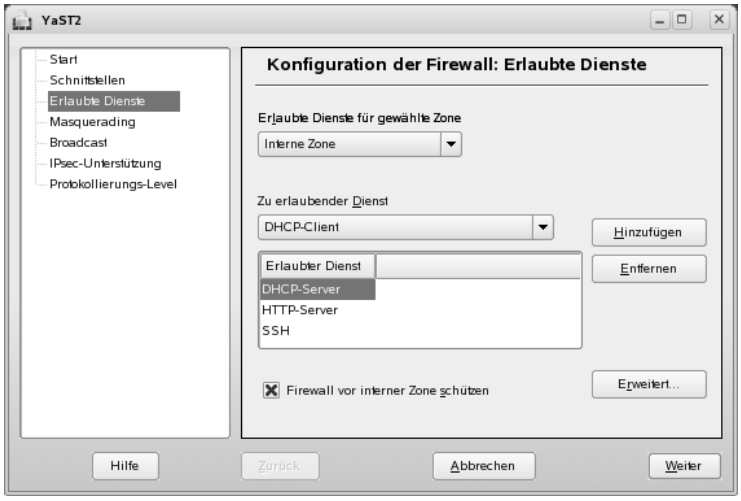


Abbildung 13.5: **Um auf einen Dienst der Firewall zuzugreifen, müssen Sie diese freischalten.**

Auch diese Einstellungen können Sie in der Konfigurationsdatei wiederfinden oder direkt dort vornehmen. Dazu müssen Sie die folgenden Variablen editieren:

```
FW_PROTECT_FROM_INT="yes"
FW_SERVICES_EXT_TCP=""
FW_SERVICES_EXT_UDP="ipsec-nat-t isakmp"
FW_SERVICES_EXT_IP="esp"
FW_SERVICES_EXT_RPC=""
FW_SERVICES_DMZ_TCP=""
FW_SERVICES_DMZ_UDP=""
FW_SERVICES_DMZ_IP=""
FW_SERVICES_DMZ_RPC=""
FW_SERVICES_INT_TCP="http ssh"
FW_SERVICES_INT_UDP="bootps"
FW_SERVICES_INT_IP=""
FW_SERVICES_INT_RPC=""
```

Die hier aufgeführte Variable `FW_SERVICES_EXT_UDP="ipsec-nat-t isakmp"` wird an mehreren Stellen von Yast verändert. Die hier eingefügten Werte `ipsec-nat-t isakmp"` werden von dem IPsec-Dialog gesetzt, wenn die IPsec-Unterstützung aktiv ist.

Der nächste Dialog beschreibt das Masquerading oder auch NAT. Zunächst aktivieren Sie hier grundsätzlich das Masquerading auf der externen Netzwerkkarte (siehe Abbildung 13.6).

Die folgenden Variablen in der Konfigurationsdatei reflektieren Ihre Einstellungen:

```
FW_ROUTE="yes"
FW_MASQUERADE="yes"
```

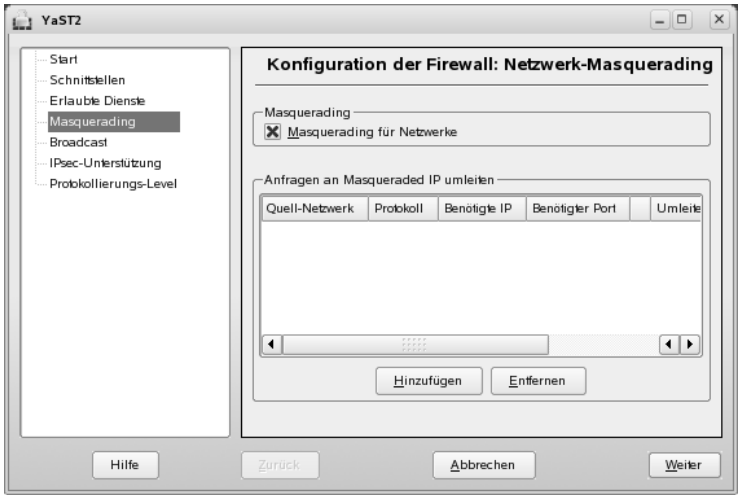


Abbildung 13.6: **Yast erlaubt Ihnen sehr einfach, das Masquerading einzuschalten.**

```
FW_MASQ_DEV="$FW_DEV_EXT"
FW_MASQ_NETS="0/0"
```

Über denselben Dialog können Sie auch Weiterleitungen von Verbindungen nach innen definieren. Wählen Sie dazu in diesem Dialog lediglich HINZUFÜGEN, und Sie erhalten einen neuen Dialog (siehe Abbildung 13.7), in dem Sie die weiterzuleitende Verbindung eingeben.

Wenn Sie die Einstellung manuell vornehmen möchten, verwenden Sie die folgenden Variablen:

```
FW_FORWARD_MASQ="0/0,10.0.0.5,tcp,80"
```

Die Weiterleitung von Broadcast-IP-Paketen wird in dem nächsten Dialog (siehe Abbildung 13.8) konfiguriert. Hier können Sie einstellen, welche IP-Broadcast-Pakete die Firewall erlaubt. Diese Broadcast-Pakete werden zum Beispiel von DHCP-Clients oder dem NETBIOS-Protokoll verwendet.

Diese Einstellungen können Sie auch manuell in den folgenden Variablen durchführen:

```
FW_ALLOW_FW_BROADCAST_EXT="no"
FW_ALLOW_FW_BROADCAST_INT="bootps"
FW_ALLOW_FW_BROADCAST_DMZ="no"
FW_IGNORE_FW_BROADCAST_EXT="no"
FW_IGNORE_FW_BROADCAST_INT="no"
FW_IGNORE_FW_BROADCAST_DMZ="no"
```

Wenn Sie bei einem der Dialoge eine Hilfe benötigen, können Sie jederzeit unten links in dem Dialog HILFE anklicken. Sie kehren anschließend durch Anklicken von BAUM zur Baumansicht zurück.

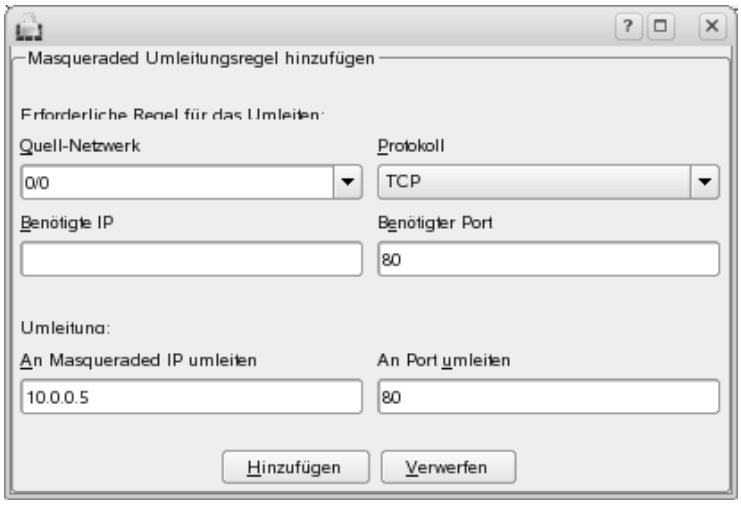


Abbildung 13.7: **Auch ein Port- oder IP-Forwarding kann leicht eingerichtet werden.**

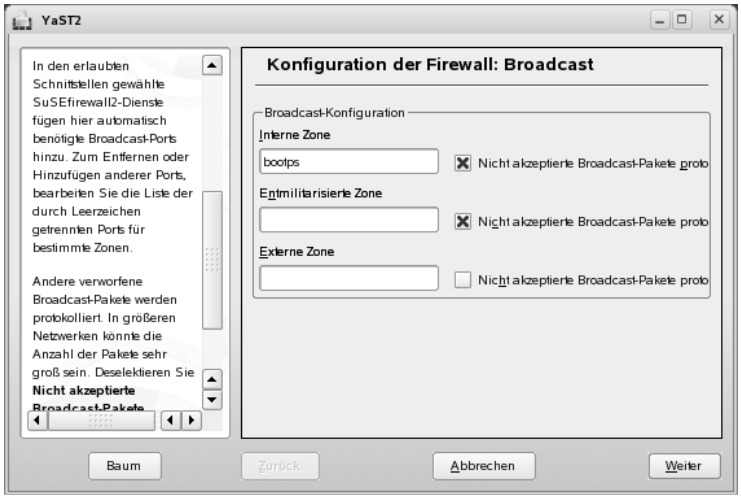


Abbildung 13.8: **Normalerweise möchten Sie auf Ihrer Firewall sämtliche Broadcast-Pakete verwerfen. Eine Ausnahme ist vielleicht ein DHCP-Server.**

Die SuSEfirewall2 unterstützt auch die Filterung von IPsec-Verkehr. Sie können einfach durch Auswahl der IPsec-Option Ihre Firewall mit entsprechenden Regeln ausstatten, die den Zugriff per IPsec erlauben (siehe Abbildung 13.9).

In der Konfigurationsdatei macht sich das durch die folgenden Variablen bemerkbar:

```
FW_SERVICES_EXT_UDP="ipsec-nat-t isakmp"
FW_SERVICES_EXT_IP="esp"
```

Als letzter Dialog wird von Yast der Protokollierungs-Level angeboten (siehe Abbildung 13.10). Hier können Sie auswählen, welche Pakete eine Protokollierung erzeugen sollen. Um die Menge der Protokolle in einem erträglichen Rahmen zu halten, ist es sinnvoll, nur kritische Pakete zu protokollieren.

Die SuSEfirewall2 unterscheidet akzeptierte und verworfene kritische Pakete. Kritische verworfene Pakete sind gespoofte Pakete, TCP-Verbindungsanfragen und bestimmte ICMP-Pakete. Kritische akzeptierte Pakete sind TCP-Verbindungsaufbauten, RPC-Verbindungs-

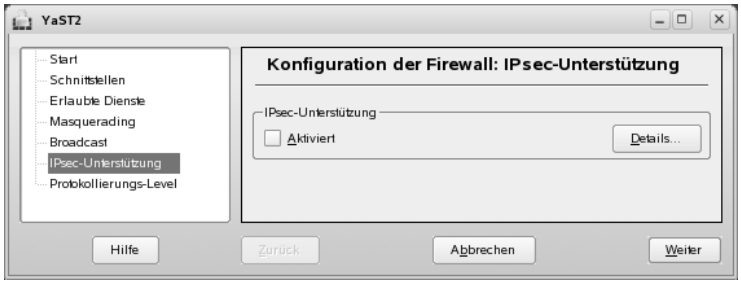


Abbildung 13.9: **Die Aktivierung der IPsec-Unterstützung erfolgt nur durch das Setzen eines Hakens.**

Abbildung 13.10: **Wählen Sie nur kritische Pakete für die Protokollierung aus, sonst erkennen Sie anschließend den Wald vor lauter Bäumen nicht mehr.**

anfragen und Zugriff auf hohe Ports. Die entsprechenden Variablen in der Konfigurationsdatei sind:

```
FW_LOG_DROP_CRIT="yes"
FW_LOG_DROP_ALL="no"
FW_LOG_ACCEPT_CRIT="no"
FW_LOG_ACCEPT_ALL="no"
```

Weitere Einstellungen erlaubt Yast Ihnen nicht. Wenn Sie nun auf WEITER klicken, fasst Yast Ihre Firewall-Konfiguration noch einmal zusammen (siehe Abbildung 13.11). Sie sollten hier noch einmal alle Einstellungen auf ihre Korrektheit kontrollieren. Durch Anwählen von ÜBERNEHMEN werden diese Einstellungen jetzt in die Konfigurationsdatei geschrieben und wird die Firewall mit diesen Einstellungen neu gestartet (siehe Abbildung 13.12).

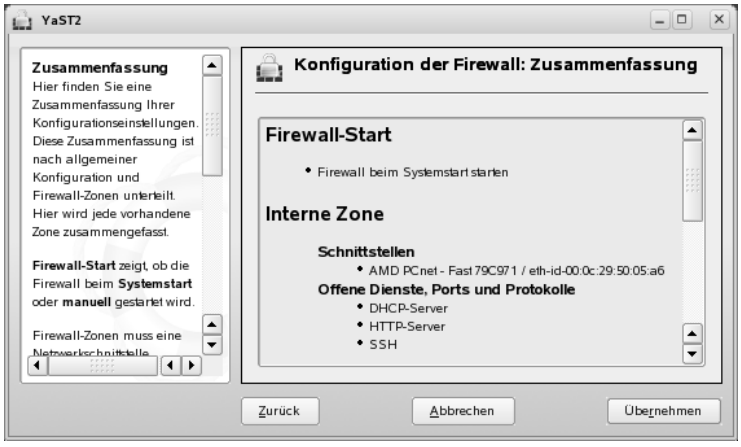


Abbildung 13.11: **Yast zeigt die Firewall-Konfiguration noch einmal in der Übersicht an.**

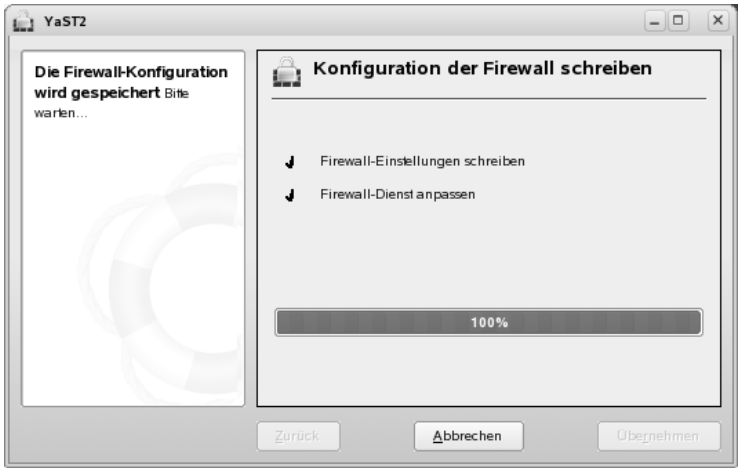


Abbildung 13.12: **Nun wird die Konfiguration aktiviert. Hoffentlich haben Sie sämtliche Fehler korrigiert.**

Die SuSEfirewall2 bietet wesentlich mehr Funktionen, als Yast Ihnen zur Konfiguration anbietet. Viele Optionen, die Yast setzt, bietet es Ihnen gar nicht zur Auswahl an, sondern diese werden grundsätzlich vorausgesetzt. Natürlich können Sie aber auch auf Yast verzichten und die Datei manuell editieren. Insgesamt enthält die Datei bei OpenSUSE-10.0 57 verschiedene Variablen, deren Namen relativ selbsterklärend sind und die durch zusätzliche Kommentare in der Datei erläutert werden. Hilfreich bei der manuellen Anpassung ist auch die erwähnte SuSEfirewall2-Dokumentation. Natürlich können Sie auch die SuSEfirewall2 abschalten. Eine Deinstallation ist nicht möglich. Aber setzen Sie einfach den Start-Modus auf MANUELL. Dann wird das System die Firewall nicht mehr starten, und Sie können eine alternative Oberfläche, z. B. Shorewall, verwenden.

Zum Abschluss finden Sie hier alle Variablen der SuSEfirewall2 als Referenz:

```
FW_DEV_EXT="any eth-id-00:0c:29:50:05:b0"
FW_DEV_INT="eth-id-00:0c:29:50:05:a6"
FW_DEV_DMZ="eth-id-00:0c:29:50:05:9c"
FW_ROUTE="yes"
FW_MASQUERADE="yes"
FW_MASQ_DEV="$FW_DEV_EXT"
FW_MASQ_NETS="0/0"
FW_PROTECT_FROM_INT="yes"
FW_SERVICES_EXT_TCP=""
FW_SERVICES_EXT_UDP="ipsec-nat-t isakmp"
FW_SERVICES_EXT_IP="esp"
FW_SERVICES_EXT_RPC=""
FW_SERVICES_DMZ_TCP=""
FW_SERVICES_DMZ_UDP=""
FW_SERVICES_DMZ_IP=""
```

```
FW_SERVICES_DMZ_RPC=""
FW_SERVICES_INT_TCP="http ssh"
FW_SERVICES_INT_UDP="bootps"
FW_SERVICES_INT_IP=""
FW_SERVICES_INT_RPC=""
FW_SERVICES_DROP_EXT=""
FW_SERVICES_REJECT_EXT="0/0,tcp,113"
FW_SERVICES_ACCEPT_EXT=""
FW_TRUSTED_NETS=""
FW_ALLOW_INCOMING_HIGHPORTS_TCP=""
FW_ALLOW_INCOMING_HIGHPORTS_UDP=""
FW_FORWARD=""
FW_FORWARD_MASQ="0/0,10.0.0.5,tcp,80"
FW_REDIRECT=""
FW_LOG_DROP_CRIT="yes"
FW_LOG_DROP_ALL="no"
FW_LOG_ACCEPT_CRIT="no"
FW_LOG_ACCEPT_ALL="no"
FW_LOG_LIMIT=""
FW_LOG=""
FW_KERNEL_SECURITY="yes"
FW_STOP_KEEP_ROUTING_STATE="no"
FW_ALLOW_PING_FW="yes"
FW_ALLOW_PING_DMZ="no"
FW_ALLOW_PING_EXT="no"
FW_ALLOW_FW_SOURCEQUENCH=""
FW_ALLOW_FW_BROADCAST_EXT="no"
FW_ALLOW_FW_BROADCAST_INT="bootps"
FW_ALLOW_FW_BROADCAST_DMZ="no"
FW_IGNORE_FW_BROADCAST_EXT="no"
FW_IGNORE_FW_BROADCAST_INT="no"
FW_IGNORE_FW_BROADCAST_DMZ="no"
FW_ALLOW_CLASS_ROUTING=""
FW_CUSTOMRULES=""
FW_REJECT=""
FW_HTB_TUNE_DEV=""
FW_IPv6=""
FW_IPv6_REJECT_OUTGOING=""
FW_IPSEC_TRUST="no"
FW_ZONES=""
FW_USE_IPTABLES_BATCH=""
FW_LOAD_MODULES=""
```

13.3 Debian

Die Debian-Distribution stellt sicherlich eine Ausnahme dar. Es gibt kein vom Debian-Projekt entwickeltes Firewall-Konfigurationswerkzeug, das diese Distribution auszeichnet. Jedoch enthält sie eine Vielzahl solcher Werkzeuge von anderen Autoren. So finden Sie Shorewall, Firewall Builder und Firestarter in der Distribution. Sie können also hier frei wählen.

14. Firewall-Distributionen

Es gibt eine Reihe von reinen Firewall-Distributionen. Während Fedora, Debian und OpenSUSE allgemeinen Zwecken genügen wollen, zielen diese Distributionen auf den reinen Einsatzzweck einer Firewall ab. Die bekanntesten aktuellen Versionen sind IPCop, IPFire und Endian. Ich beschränke mich hier auf die IPCop-Distribution. Sowohl IPFire als auch Endian sind Weiterentwicklungen dieser Version.

14.1 IPCop

IPCop ist eine reine GPL-Firewall-Distribution (`http://www.ipcop.org`). Die aktuelle Version ist die Version 1.4.21. Gleichzeitig wird aber auch eine Version 1.9.x als Beta entwickelt. Diese bringt im Wesentlichen die folgenden Neuerungen mit:

- aktueller Linux-Kernel
- Unterstützung für weitere Hardware
- Entfernung der Kernkomponente Snort. Dies ist nun ein Add-On.
- Integration von OpenVPN

Sie können diese Distribution als ISO-Image von der Homepage herunterladen, auf eine CD brennen und anschließend zur Installation nutzen. Für die Installation von IPCop benötigen Sie einen beliebigen Intel-Rechner mit mindestens einem Intel 386-Prozessor, 32 Mbyte RAM und einer 300-Mbyte-Festplatte. In den meisten Fällen werden Sie sicherlich mehr zur Verfügung haben. Ein Diskettenlaufwerk und ein CD-ROM-Laufwerk sind nicht erforderlich, aber nützlich. Nach der Installation benötigen Sie auch keinen Monitor und keine Tastatur mehr.

14.1.1 Das IPCop-Konzept

Die IPCop-Firewall dient, wie jede andere Firewall auch, als Wächter zwischen mehreren Netzwerken. Um diese Funktion leicht verständlich zu visualisieren, benennt IPCop die verschiedenen Netzwerkkarten der Firewall mit unterschiedlichen Farben:

- **Rot**: Die rote Netzwerkkarte verbindet die IPCop-Firewall mit dem Internet oder einem ähnlich unsicheren Netz.
- **Grün**: An der grünen Netzwerkkarte wird das von der Firewall geschützte Netz angeschlossen.

» **Blau**: Das blaue Netz ist für drahtlose Netze reserviert. Der Zugriff von dem blauen Netz auf das grüne Netz wird streng kontrolliert. Die Verwendung der blauen Netzwerkkarte ist optional.

» **Orange**: Das orange Netz ist die DMZ. Der Zugriff aus diesem Netz in das blaue oder grüne Netz wird streng kontrolliert.

Diese Farben sind bei der Konfiguration und der späteren Administration der IPCop-Firewall sehr wichtig. Die Farben vereinfachen aber auch die Konfiguration und die Kommunikation.

Sie sollten sich bereits vor der Installation der IPCop-Firewall Gedanken über die in dem grünen Netz verwendeten IP-Adressen und speziell über die IP-Adresse der grünen Netzwerkkarte der IPCop-Firewall machen.

Die IPCop-Firewall unterstützt noch folgende bemerkenswerte Eigenschaften:

» den eingebauten Update-Mechanismus mit einem Protokoll der installierten Updates. Nur signierte Updates werden von IPCop entgegengenommen.

» grafische Status-Seiten mit Anzeige des Netzwerkdurchsatzes und der Prozessorauslastung

» zusätzliche Netzwerkdienste: Proxy, DHCP-Server und NTP-Zeitserver

» Bandbreitenkontrolle. Die Bandbreitenkontrolle kann sehr einfach konfiguriert werden, ohne dass man die Einzelheiten des TCP-Protokolls kennen muss.

» Intrusion Detection System (IDS) mit Snort

» grafische Administration der Firewall. Jedoch können Sie leider nicht den Zugriff von innen auf das Internet einschränken. Sie haben mit diesem Interface lediglich die Möglichkeit, den Zugriff von außen nach innen zu erlauben.

» komplette VPN-Unterstützung für IPsec-Verbindungen mit Preshared-Keys und Zertifikaten

» ausführliche Protokolle

Wenn Sie eine einfache Lösung für die Verwaltung einer Firewall suchen, die auch von Nicht-Linux-Administratoren verwaltet werden kann, ist IPCop durchaus ein vielversprechender Kandidat.

14.1.2 IPCop-Installation

Die Installation von IPCop ist recht einfach, wenn Sie bereits andere Linux-Distributionen installiert haben. Besonders, wenn Sie schon Debian oder Fedora Linux textbasiert installiert haben, werden die Fragen bei der Installation Sie vor keine größeren Probleme stellen.

STOP

Eine Dual-Boot-Installation ist bei IPCop nicht möglich. IPCop verwendet die gesamte Festplatte! Der erste Bildschirm nach dem Boot von der CD-ROM weist Sie darauf hin.

Wenn Sie IPCop testweise in VMWare installieren möchten, wählen Sie bitte eine virtuelle IDE-Festplatte in VMWare aus. Die virtuelle SCSI-Festplatte wird teilweise nicht erkannt.

Nun können Sie zunächst die Sprache auswählen. IPCop unterstützt auch die deutsche Sprache. Nach einigen weiteren Bildschirmen werden die Pakete installiert, und Sie werden gefragt, ob Sie eine Diskette mit dem Backup einer alten IPCop-Installation besitzen. Diese können Sie hier wieder einspielen (siehe Abbildung 14.1).

Auf dem nächsten Bildschirm werden Sie aufgefordert, das Netzwerk zu konfigurieren. IPCop unterscheidet das grüne, das blaue, das orange und das rote Netzwerk. Hier müssen Sie die grüne Netzwerkkarte konfigurieren (siehe Abbildung 14.2). Wählen Sie die Automatische Erkennung. Anschließend müssen Sie die IP-Adresse für die grüne Netzwerkkarte eingeben. Wenn Ihr grünes Netzwerk die IP-Adressen 192.168.15.0/24 verwendet, werden Sie der IPCop-Firewall wahrscheinlich entweder die IP-Adresse 192.168.15.1 oder 192.168.15.254 zuweisen.

Sobald Sie die IP-Adresse für die grüne Netzwerkkarte bestätigt haben, wird der Boot-Manager Grub installiert, und die Installation ist abgeschlossen. Entfernen Sie nun alle Disketten und CD-ROMs aus den Laufwerken, und bestätigen Sie mit Ok.

Abbildung 14.1: **IPCop unterstützt auch die Wiederherstellung mit einer Backup-Diskette.**

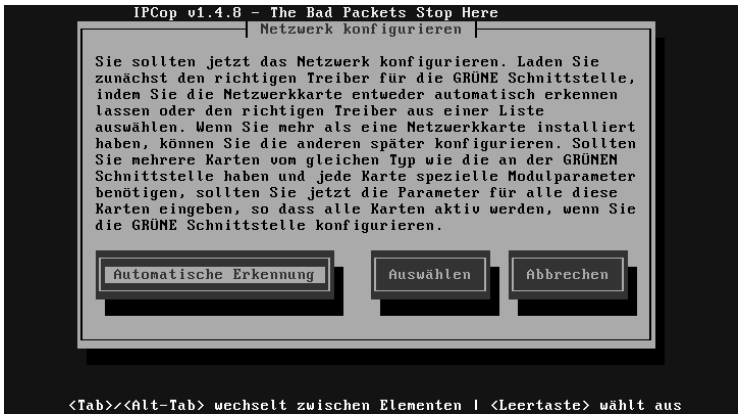


Abbildung 14.2: **Achten Sie bei der Wahl der grünen Netzwerkkarte darauf, dass Sie die richtige auswählen.**

IPCop wird nun das Setup-Programm für die weitere Konfiguration aufrufen. Als Tastatur-Layout wählen Sie für eine deutsche Tastatur bitte `de-latin1-nodeadkeys`. Die Zeitzone für Deutschland ist `Europe/Berlin`. Als Rechnernamen können Sie die Voreinstellung `ipcop` beibehalten oder auch einen eigenen Namen nach Ihren Wünschen wählen. Anschließend wählen Sie einen geeigneten Domänennamen. Wenn Sie sich mit einer ISDN-Karte einwählen, müssen Sie das Protokoll (DSS1), die ISDN-Karte und mögliche zusätzliche Modulparameter für den Treiber und Ihre lokale Telefonnummer eingeben. Anschließend aktivieren Sie das ISDN. Wenn Sie keine ISDN-Karte verwenden, können Sie einfach den Bildschirm mit ISDN Deaktivieren bestätigen.

Wenn Sie keine ISDN-Karte verwenden oder zusätzlich eine DMZ oder ein drahtloses Netz verwenden möchten, müssen Sie auf dem nächsten Bildschirm die Netzwerkkonfiguration ändern. Ist Ihre IPCop-Firewall zum Beispiel mit einer Netzwerkkarte über DSL mit dem Internet verbunden, wählen Sie GREEN + RED. Verfügen Sie zusätzlich über eine DMZ, wählen Sie GREEN + ORANGE + RED und so weiter (siehe Abbildung 14.3). Anschließend müssen

Abbildung 14.3: **IPCop weist den Firewall-Zonen Farben zu. Wählen Sie die für Sie richtige Farbkombination aus.**

Abbildung 14.4: **IPCop unterstützt die Internetanbindung mit PPTP, PPPOE, DHCP und statischer IP-Adresse.**

Sie den verschiedenen Farben Netzwerkkarten und IP-Adressen zuweisen. Die grüne Netzwerkkarte besitzt bereits eine IP-Adresse. Die orange und die blaue Netzwerkkarte erhält jeweils statische IP-Adressen. Bei der roten Netzwerkkarte können Sie zwischen einer statischen und einer dynamischen IP-Adresse wählen (siehe Abbildung 14.4). Wenn Sie für die rote Netzwerkkarte eine statische IP-Adresse ausgewählt haben, können Sie auch noch zwei DNS-Server und das Gateway angeben.

STOP

Die angegebenen DNS-Server und das Gateway haben Vorrang vor den Angaben, die Ihr Provider Ihnen bei z. B. PPPOE mitteilt. Stellen Sie sicher, dass Sie bei einer dynamischen IP-Adresse hier keine Daten eingeben!

Nun sind Sie fast fertig. Auf dem nächsten Bildschirm gibt Ihnen IPCop die Möglichkeit, einen DHCP-Server auf der IPCop-Firewall für Ihre Clients im grünen Netz zu betreiben. Wenn Sie diese Funktion wünschen, aktivieren Sie den DHCP-Server und geben einen sinnvollen DHCP-Bereich vor (siehe Abbildung 14.5).

Abbildung 14.5: **IPCop kann selbst als DHCP-Server für das grüne Netz arbeiten.**

Abbildung 14.6: **Ein Aufruf der Konfigurationswerkzeuge ist auf der Konsole mit dem Befehl setup möglich.**

Nun müssen Sie nur noch das Kennwort für den Benutzer root und für den IPCop-Administrator eingeben, und die Installation ist abgeschlossen. Nach einem Reboot können Sie sich mit dem Web-Interface verbinden.

Wenn Sie zu einem späteren Zeitpunkt feststellen, dass Sie die Konfiguration der letzten Schritte ändern müssen, dann können Sie das nicht über das Web-Interface erledigen. Sie müssen sich auf dem IPCop-System lokal anmelden und können dort das Kommando `setup` aufrufen. Dieses Kommando präsentiert Ihnen ein Menü (siehe Abbildung 14.6), in dem Sie die Werkzeuge auswählen können, die Sie für die Änderung benötigen.

14.1.3 IPCop-Web-Interface

Das Web-Interface von IPCop ist über die IP-Adresse der grünen Netzwerkkarte erreichbar. Geben Sie einfach eine der beiden folgenden URLs in Ihrem Browser ein:

» `http://192.168.15.254:81/` oder

» `https://192.168.15.254:445/`

Ab der Version 1.4.0 ist die Verbindung nur noch verschlüsselt erlaubt. Daher werden Sie bei der Eingabe der ersten URL automatisch auf die zweite URL umgeleitet. Achten Sie bei den URLs auf die ungewöhnlichen Port-Nummern. Wenn Sie in Ihrer `/etc/hosts`-Datei den folgenden Eintrag vornehmen, können Sie auch anstelle der IP-Adresse den Rechnernamen `ipcop` verwenden:

```
192.168.15.254  ipcop.spenneberg.net ipcop
```

Bei der ersten Verbindung mit IPCop wird sich Ihr Browser über das Zertifikat des IPCop-Webservers beschweren. Die Meldung wird ähnlich aussehen wie in Abbildung 14.7. Wenn Sie

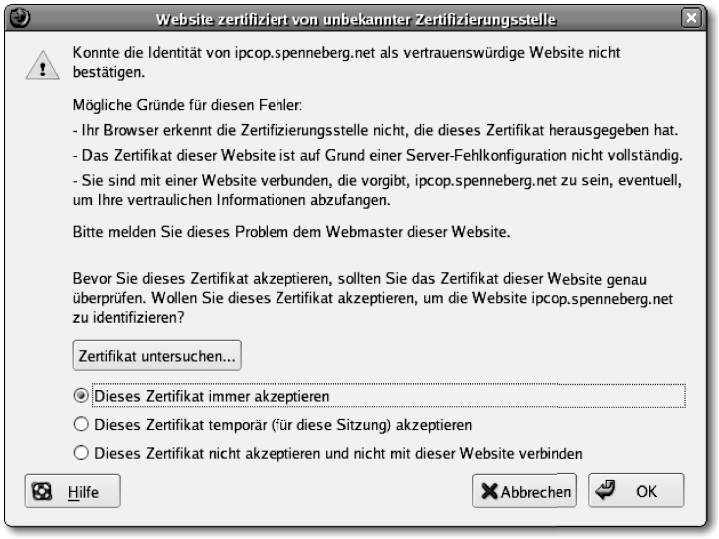


Abbildung 14.7: **Bei der ersten Verbindung müssen Sie das Zertifikat akzeptieren.**

sicher sind, dass sich in diesem Moment kein Bösewicht in Ihre Verbindung eingeklinkt hat, sollten Sie das Zertifikat dauerhaft akzeptieren.

Falls Sie sich nicht sicher sind, dass das Zertifikat tatsächlich von Ihrer IPCop-Installation stammt, können Sie das überprüfen, wenn Sie sich auf Ihrem IPCop-System anmelden können. Geben Sie auf der Kommandozeile folgenden Befehl ein:

```
# openssl x509 -noout -fingerprint -in /etc/httpd/server.crt
MD5 Fingerprint=7D:94:26:27:D1:A3:59:59:A4:64:33:43:D7:FD:7E:59
```

Vergleichen Sie diesen Fingerprint mit dem von dem Browser angezeigten Fingerprint (bei Firefox wählen Sie ZERTIFIKAT UNTERSUCHEN; siehe Abbildung 14.8).

Wenn Sie sich über die IP-Adresse mit Ihrer IPCop-Installation verbinden, wird Ihnen anschließend noch eine zweite Warnung (siehe Abbildung 14.9) angezeigt. Die können Sie ignorieren. Um in Zukunft diese Warnung nicht mehr angezeigt zu bekommen, sollten Sie Ihre Datei `/etc/hosts` entsprechend anpassen. Diese Datei existiert auch auf einem Windows-System in `system32/drivers/etc/` und wird auch dort benutzt.

Nun sind Sie mit dem Web-Interface verbunden und können die IPCop-Firewall administrieren. Hoffentlich haben Sie bei der Konfiguration alles richtig gemacht. Wenn die IPCop-

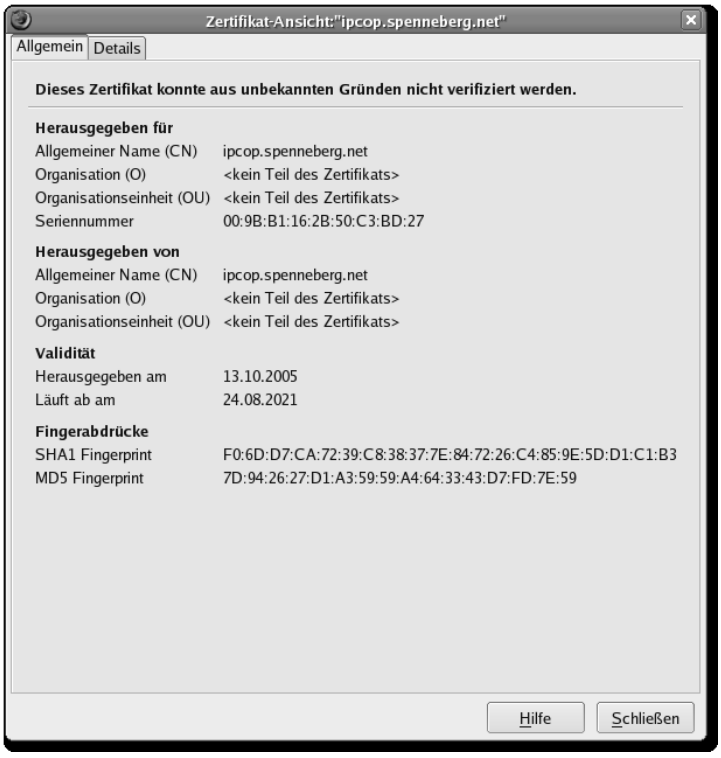


Abbildung 14.8: **Überprüfen Sie im Zweifelsfall die Fingerabdrücke des Zertifikats.**

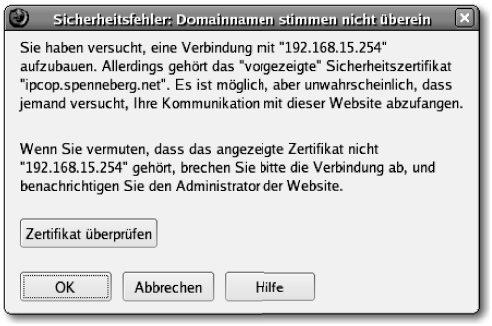


Abbildung 14.9: **Firefox beschwert sich, da das Zertifikat für ipcop.spenneberg.net ausgestellt ist, die Verbindung jedoch mit 192.168.15.254 aufgebaut wird.**

Firewall bereits an das Internet angeschlossen ist, begrüßt die Firewall Sie mit der Meldung, dass die Firewall verbunden sei, und zeigt an, ob Updates verfügbar sind. Wenn dies der Fall ist, sollten Sie zunächst diese Updates installieren. Wechseln Sie hierzu auf die Seite UPDATES

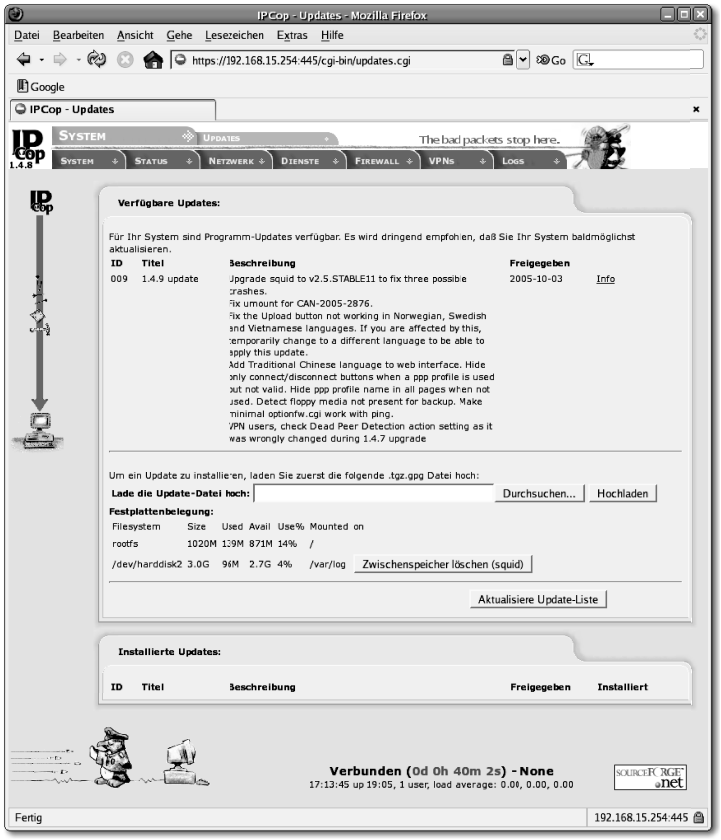


Abbildung 14.10: **IPCop hat einen eingebauten Update-Mechanismus. So sind Sie vor bekannten Sicherheitslücken geschützt.**

in der Registerkarte System. Dazu werden Sie aufgefordert, sich als Benutzer admin anzumelden und Ihr Kennwort einzugeben. Das Update wird Ihnen dann beschrieben, und Sie finden dort einen Download-Link (Info), mit dem Sie zunächst das Update herunterladen und dann auf die IPCop-Firewall hochladen können (siehe Abbildung 14.10).

Die Updates sind zum Schutz vor Trojanern mit GnuPG signiert. Nur korrekt signierte Updates werden von Ihrer IPCop-Firewall akzeptiert. Damit sind Sie sicher, dass nicht ein Angreifer Ihnen ein fehlerhaftes Update unterschiebt.

Auf der Registerkarte Status finden Sie die folgenden Informationen:

» **System-Status**. Hier finden Sie Informationen über die laufenden Dienste, den freien Speicherplatz und die Prozesse.

» **Netzwerk-Status**. Diese Seite zeigt Ihnen an, wie Ihre Netzwerkkarten, Routen und Internetverbindungen konfiguriert sind.

Abbildung 14.11: **IPCop unterstützt bereits die VRT-Snort-Regelsätze.**

» **System-Diagramme**. Hier finden Sie grafische Darstellungen der CPU-, Speicher- und Swap-Nutzung Ihrer Firewall.

» **Netzwerk-Diagramme**. Diese Seite zeigt die Auslastung des Netzwerks mithilfe von Diagrammen grafisch an.

» **Verbindungen**. Hier werden Ihnen die aktuellen Netzwerkverbindungen angezeigt. Im Grunde handelt es sich um eine Darstellung der Datei `nf_conntrack`.

Auf der Registerkarte Netzwerk können Sie besondere Einstellungen für spezielle Modems und DSL-Geräte vornehmen. Einige dieser Geräte benötigen eine spezielle Firmware für ihre Funktion, die von IPCop in dem Gerät installiert werden muss. Hier konfigurieren Sie diese Funktionen.

Die Registerkarte Dienste erlaubt Ihnen die Konfiguration eines Proxys, eines DHCP-Servers, eines dynamischen DNS-Clients, die Bearbeitung der lokalen Hosts-Datei, den Einsatz eines

Abbildung 14.12: **IPCop übernimmt die komplette Verwaltung der VPNs.**

NTP-Zeitservers, die Konfiguration der Bandbreitenregelung und die Aktivierung des Snort-Intrusion-Detection-Systems. Dabei unterstützt IPCop sogar bereits die neuen VRT-Regelsätze von Snort und erlaubt Ihnen die Eingabe des Lizenzschlüssels (siehe Abbildung 14.11).

Auf der Registerkarte Firewall können Sie lediglich Verbindungen von außen nach innen konfigurieren. Normalerweise sind diese Verbindungen nicht erlaubt. Sie können aber von außen Port-Weiterleitungen nach innen konfigurieren. Sie können den externen Zugang zur IPCop-Administrationsoberfläche von außen freigeben und DMZ-Schlupflöcher (aus der DMZ in das grüne Netz) konfigurieren. Die Verbindungen aus dem grünen und dem blauen Netz in das Internet sind immer erlaubt und können nicht eingeschränkt werden!

Auf der Registerkarte VPN können Sie sehr einfach ein VPN konfigurieren. Hierzu können Sie zunächst eine Zertifikatsautorität erzeugen oder die Zertifikate einer externen CA hochladen. Anschließend erzeugen Sie Verbindungen und können den Status der Verbindung beobachten (siehe Abbildung 14.12). Bevor ich das Buch mit Screenshots zu sehr aufblähe, beende ich hier die Beschreibung von IPCop, da es auf der IPCop-Homepage auch eine sehr gute 60 Seiten starke Administrationsanleitung gibt. Ich denke, Sie haben die wesentlichen Möglichkeiten von IPCop kennengelernt und finden sich auf dem Web-Interface von IPCop auch selbst recht schnell zurecht.

Ein großer Vorteil von IPCop ist die große Anzahl an Add-Ons, die für diese Firewall verfügbar sind. So kann fast jeder Wunsch befriedigt werden.

15. Testmethoden und -werkzeuge

Wer eine Firewall aufsetzt, sollte diese auch testen. Syntaktische Fehler in Ihrem Firewall-Skript erkennen Sie direkt bei dem Aufruf. Logische Fehler sind wesentlich schwerer zu erkennen. Nur der Test zeigt Ihnen, ob Sie einen logischen Fehler in Ihrem Regelwerk übersehen haben. Die Werkzeuge, die ich in diesem Kapitel vorstelle, eignen sich auch zum Test und Audit fremder Firewall-Systeme und Sicherheitsstrukturen. Daher gehe ich auch ein wenig ausführlicher auf Funktionen ein, die hier sinnvoll sind. Denken Sie jedoch bitte daran, dass niemand gern gescannt wird. Sie sollten sich vorher immer die schriftliche Erlaubnis einholen oder von der betroffenen Person dazu aufgefordert worden sein. Die Anwendung von `nessus` kann im Gegensatz zu der von `nmap` meiner Einschätzung nach sogar strafrechtliche Konsequenzen nach sich ziehen.

INFO

Jedoch bietet auch `nmap` in seiner aktuellen Version die Möglichkeit, automatisch Sicherheitslücken zu prüfen und Angriffe durchzuführen. Es kommt wie immer auf die tatsächliche Anwendung an.

15.1 Test der Regeln

Sobald Sie Ihr Firewall-Skript fertiggestellt haben, sollten Sie es auf syntaktische Fehler testen. Die beste Möglichkeit hierzu ist der Aufruf des Shell-Skripts. Die Shell wird Syntaxfehler auf der Konsole melden. Leider schleichen sich auch immer logische Fehler in einem Firewall-Skript ein. Obwohl Sie glauben, alle Regeln richtig definiert zu haben, entspricht das Ergebnis nicht Ihren Vorstellungen. Dann müssen Sie nach dem Fehler suchen und diesen beheben.

Am einfachsten beginnen Sie mit den Diensten, deren Nutzung das Firewall-Skript erlauben soll. Prüfen Sie, ob die gewünschten Funktionen zur Verfügung stehen. Falls dies nicht der Fall ist, analysieren Sie den Netzwerkverkehr, und vergleichen Sie diesen mit Ihren Regeln. Für die Analyse des Netzwerkverkehrs eignen sich unter Linux die Befehle `tcpdump` und `wireshark`. `Tcpdump` ist ein einfacher Kommandozeilenbefehl, der die Netzwerkpakete anzeigen kann. Dabei kann `Tcpdump` auch die Pakete sehen, die später von den Regeln verworfen werden. Die Ausgabe von `Tcpdump` ist in Abbildung 15.1 zu sehen.

Wireshark ist ein grafisches Werkzeug, das im Gegensatz zum spartanischen Tcpdump die Daten ausführlicher aufbereitet (siehe Abbildung 15.2).

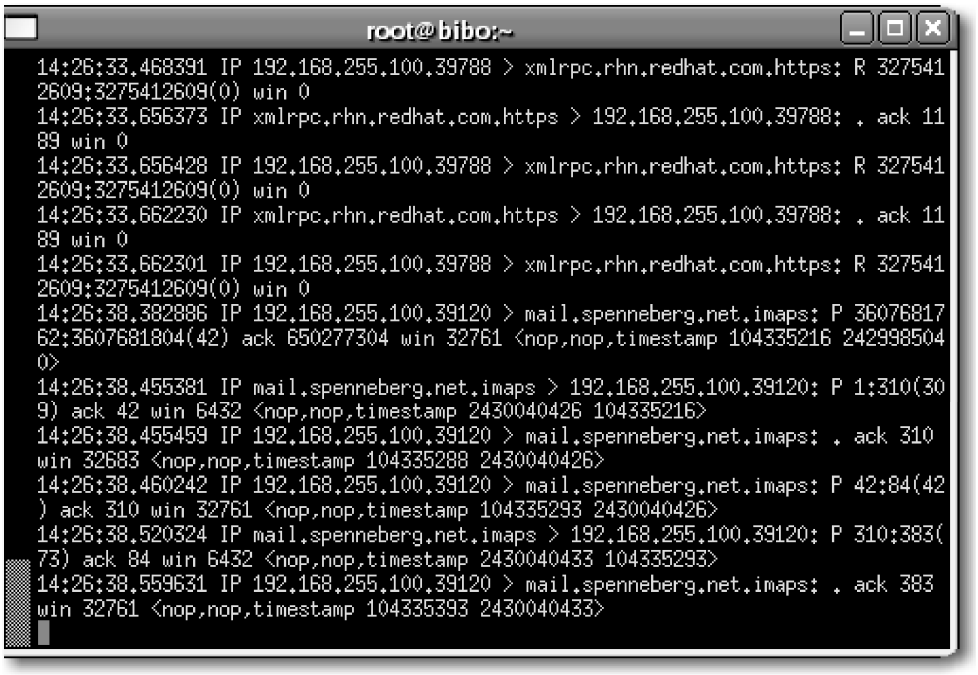


Abbildung 15.1: **Tcpdump zeigt die Netzwerkpakete auf der Konsole an.**

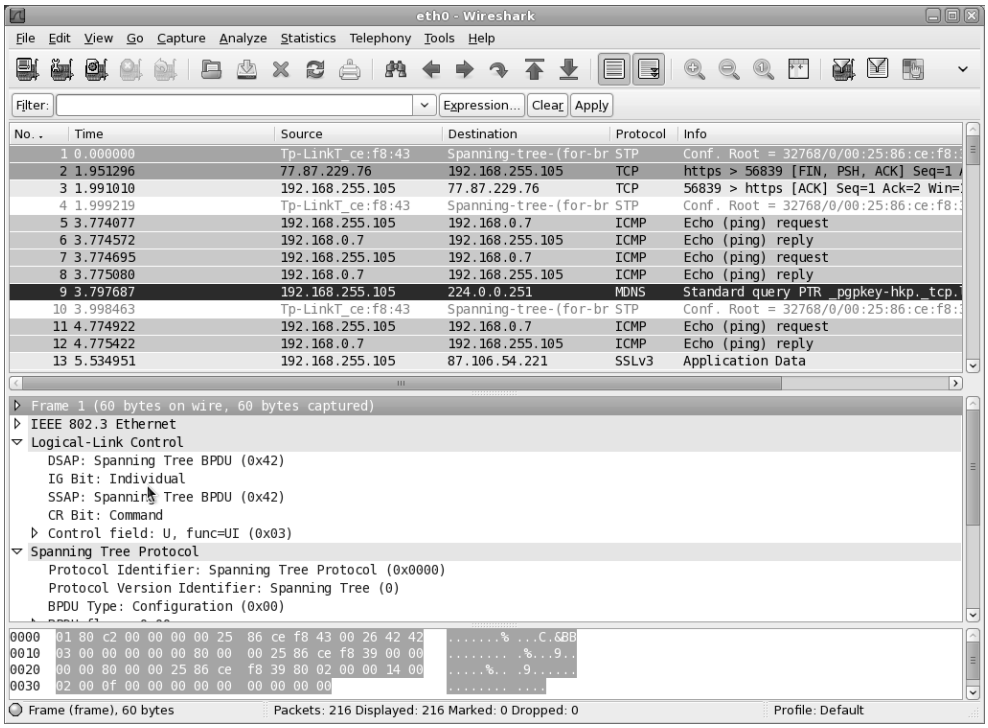


Abbildung 15.2: **Wireshark browst die Netzwerkpakete grafisch.**

15.2 Fehlersuche am Beispiel

Falls Sie so immer noch nicht erkennen können, warum Ihr Firewall-Skript den speziellen Dienst unterbindet, fügen Sie zusätzliche Log-Regeln ein. Sorgen Sie dafür, dass in jeder Firewall-Filterkette alle Pakete protokolliert werden, bevor die Pakete verworfen werden.

Im Folgenden möchte ich Ihnen anhand eines Beispiels die Vorgehensweise zeigen. Schauen Sie sich das folgende Skript an, das von unserem hypothetischen Firewall-Administrator geschrieben wurde. Die Anforderung an die Firewall war:

» Erlaube keinen Zugriff von außen in das geschützte Netz.

» Erlaube aus dem geschützten Netz den HTTP- und HTTPS-Zugriff auf beliebige Webserver.

Der Firewall-Administrator erzeugte aus dieser Anforderung das folgende Skript:

```
#!/bin/sh

# Firewall-Skript
# Lehnt jeden Zugriff von außen ab
# Erlaubt HTTP und HTTPS

# Netzwerkkarten
INTDEV=eth0
EXTDEV=eth1

IPTABLES=/sbin/iptables

SERVICE="80,443"

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

$IPTABLES -A FORWARD -p tcp -i $INDEV -o $EXTDEV -m multiport --dport ↵
        $SERVICE -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Dieses Skript hat drei Schönheitsfehler, die wir nun suchen wollen. Zunächst startet der Administrator dieses Skript. Bei dem Start bekommt er direkt eine Fehlermeldung angezeigt:

```
# sh fw_skript
Warning: wierd character in interface '-o' (No aliases, :, ! or *).
Bad argument 'eth1'
Try 'iptables -h' or 'iptables --help' for more information.
```

Leider zeigt die Fehlermeldung nicht, wo in dem Skript der Fehler aufgetreten ist. Bei diesem kurzen Skript ist es sicherlich leicht für Sie, den Fehler zu finden. Bei einem Skript von 200 oder 300 Regeln ist es ungemein schwerer, die Zeile mit dem Fehler zu entdecken.

Die Bash-Shell bietet hierfür die Option `-x`. Dann zeigt die Shell jede Zeile inklusive der ersetzten Variablen vor der Ausführung:

```
# sh -x fw_skript
+ INTDEV=eth0
+ EXTDEV=eth1
+ IPTABLES=/sbin/iptables
+ SERVICE=80,443
+ /sbin/iptables -P INPUT DROP
+ /sbin/iptables -P OUTPUT DROP
+ /sbin/iptables -P FORWARD DROP
+ /sbin/iptables -A FORWARD -p tcp -i -o eth1 -m multiport --dport 80,443 ↵
         -m state --state NEW -j ACCEPT
Warning: wierd character in interface '-o' (No aliases, :, ! or *).
Bad argument 'eth1'
Try 'iptables -h' or 'iptables --help' for more information.
+ /sbin/iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ↵
        ACCEPT
```

Nun fällt auf, dass in der fett hervorgehobenen Zeile die Angabe der Netzwerkkarte für die Option `-i` fehlt. Scheinbar wurde die Variable nicht initialisiert, oder es gibt einen Tippfehler. Sie werden den Tippfehler sicherlich schon gefunden haben. Die Variable wurde mit dem Namen `INTDEV` initialisiert, und in dieser Zeile wurde die nicht definierte Variable `INDEV` verwendet. Nach der Korrektur läuft das Skript fehlerfrei. Allerdings stellt der Administrator bei mehrmaligen Aufrufen des Skripts fest, dass die neuen Regeln immer an die Ketten angehängt werden. Die Ketten werden nie gelöscht. Er fügt zusätzlich zu Beginn eine Zeile ein, die die Ketten zunächst löscht:

```
$IPTABLES -F
```

Dennoch ist es nicht möglich, die Firewall wie gewünscht zu nutzen. Zur Fehlersuche zeigt der Administrator nach dem Versuch, auf die Seite `http://www.os-t.de` zuzugreifen, folgende Regeln an:

```
# iptables -vnL FORWARD
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in   out  source    destination
0    0     ACCEPT all  --  *    *    0.0.0.0/0 0.0.0.0/0   state RELATED, ↵
        ESTABLISHED
0    0     ACCEPT tcp  --  eth1 eth0 0.0.0.0/0 0.0.0.0/0   multiport  ↵
        dports 80,443 state NEW
0    0     ACCEPT all  --  *    *    0.0.0.0/0 0.0.0.0/0   state RELATED, ↵
        ESTABLISHED
```

Erstaunlicherweise haben alle Zähler der Regeln den Wert 0. Der Administrator startet nun auf der Firewall `tcpdump` auf der internen Netzwerkkarte, um den Verkehr zu beobachten:

```
# tcpdump -ni eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol ↵
        decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
10:53:43.766500 IP 10.0.1.128.32769 > 8.8.8.8.domain: 35667+ A? www.os-t. ↵
        de. (36)
10:53:48.270266 arp who-has 10.0.1.1 tell 10.0.1.128
10:53:48.270286 arp reply 10.0.1.1 is-at 00:50:56:c0:00:02
10:53:48.270823 IP 10.0.1.128.32769 > 8.8.8.8.domain: 35667+ A? www.os-t. ↵
        de. (36)
```

Eigentlich sollte diese Information dem Administrator für die Fehlersuche bereits ausreichen. Der Client versucht, einen DNS-Server zu erreichen, um den Namen `www.os-t.de` aufzulösen, und erhält keine Antwort. Anscheinend erlaubt der Regelsatz keinen DNS-Verkehr.

Der Administrator möchte es aber genau wissen und fügt manuell eine `LOG`-Regel an die Kette an:

```
# iptables -A FORWARD -j LOG --log-prefix "FORWARD: "
# iptables -vnL FORWARD
 Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in   out  source     destination
0    0     ACCEPT tcp  --  eth1 eth0 0.0.0.0/0  0.0.0.0/0  multiport ↵
        dports 80,443 state NEW
0    0     ACCEPT all  --  *    *    0.0.0.0/0  0.0.0.0/0  state RELATED, ↵
        ESTABLISHED
0    0     LOG    all  --  *    *    0.0.0.0/0  0.0.0.0/0  LOG flags 0 ↵
        level 4 prefix ‘FORWARD: ’
```

Nun versucht er erneut, auf die Webseite zuzugreifen, und beobachtet die Protokolle. Dort findet er folgende Einträge:

```
Jul 23 11:00:24 bibo kernel: FORWARD: IN=eth1 OUT=eth0 SRC=10.0.1.128 DST ↵
        =8.8.8.8 LEN=64 TOS=0x00 PREC=0x00 TTL=63 ID=0 DF PROTO=UDP SPT ↵
        =32769 DPT=53 LEN=44
Jul 23 11:00:29 bibo kernel: FORWARD: IN=eth1 OUT=eth0 SRC=10.0.1.128 DST ↵
        =8.8.8.8 LEN=64 TOS=0x00 PREC=0x00 TTL=63 ID=1 DF PROTO=UDP SPT ↵
        =32769 DPT=53 LEN=44
```

Diese Meldungen zeigen ihm nun eindeutig, dass die DNS-Anfragen an den UDP-Port 53 nicht von der Firewall erlaubt werden. Nach der Kontrolle seines Skripts stellt er fest, dass er folgende Regel vergessen hat:

```
$IPTABLES -A FORWARD -p udp -i $INTDEV -o $EXTDEV --dport 53  -m state -- ↵
        state NEW -j ACCEPT
```

Nach einem Neustart des Skripts funktioniert auch der Zugriff auf das Internet. Das Skript erlaubt den gewünschten Zugriff. Nun muss der Administrator noch prüfen, ob auch der unerwünschte Zugriff unterbunden wird. Anstatt jeden einzelnen Dienst manuell zu testen, verwendet er `nmap`, das ich im nächsten Abschnitt genauer erläutere.

15.3 Nmap

Nmap, der „Network Mapper", ist ein Open-Source-Werkzeug, das Sie bei der Analyse von Netzwerken und bei Sicherheitsaudits unterstützen kann. Es scannt große Netzwerke, wie auch einzelne Rechner, sehr schnell und kann sogar in vielen Fällen das Betriebssystem und die Laufzeit des Rechners ermitteln. In den Händen eines kundigen Anwenders, der Sie nach diesem Kapitel beginnen zu sein, erlaubt Nmap mächtige Prüfungen. Sie können feststellen, ob die Firewall *stateful* ist, welchen Patchlevel das Betriebssystem hat, welcher Dienst mit welcher Versionsnummer eingesetzt wird und ob ein Spoofen von TCP-Verbindungen möglich ist. Dabei sind die verschiedenen Funktionen von Nmap so gut dokumentiert, dass Sie nach dieser Einführung weitere Funktionen selbst sehr leicht erlernen können.

15.3.1 Nmap-Installation

Der Befehl `nmap` ist bei den meisten Linux-Distributionen fester Bestandteil. Daher ist eine zusätzliche Installation aus den Quellen nicht zwingend erforderlich. Dennoch möchte ich im Weiteren kurz die Installation von Nmap beschreiben, da Sie nur so die aktuellste Version erhalten.

INFO

Nmap wird regelmäßig aktualisiert. Nmap erhält neue Funktionen, und die Fingerprint-Datenbank wird regelmäßig erweitert. Es ist daher sinnvoll, immer die neueste Version von Nmap einzusetzen. Hierfür ist nicht unbedingt die Installation aus den Quellen erforderlich, da Sie auf der Nmap-Homepage auch RPM-Pakete der neuesten stabilen Version finden (`http://www.insecure.org/nmap/nmap_download.html`). Dort finden Sie auch Nmap in einer Version für Microsoft Windows! Allerdings gibt es bei einigen Windows-Versionen Probleme zwischen Nmap und der Windows-Firewall. Hier sollten Sie die Nmap-Dokumentation lesen.

Um Nmap aus den Quellen zu installieren, laden Sie zunächst die aktuelle Version von der Homepage und entpacken diese in einem geeigneten Verzeichnis.

```
# cd /tmp wget http://download.insecure.org/nmap/dist/nmap-<version>.tar. ↵
        bz2
# cd /usr/local/src tar
# xjf /tmp/nmap-<version>.tar.bz2
# cd nmap-<version>
```

Die Übersetzung erfolgt mit dem üblichen Dreisatz:

```
# ./configure
# make
# sudo make install
```

Wenn Sie Nmap in einem anderen Verzeichnis als `/usr/local` installieren möchten, können Sie das Verzeichnis bei dem `./configure --prefix <dir>`-Aufruf mit angeben.

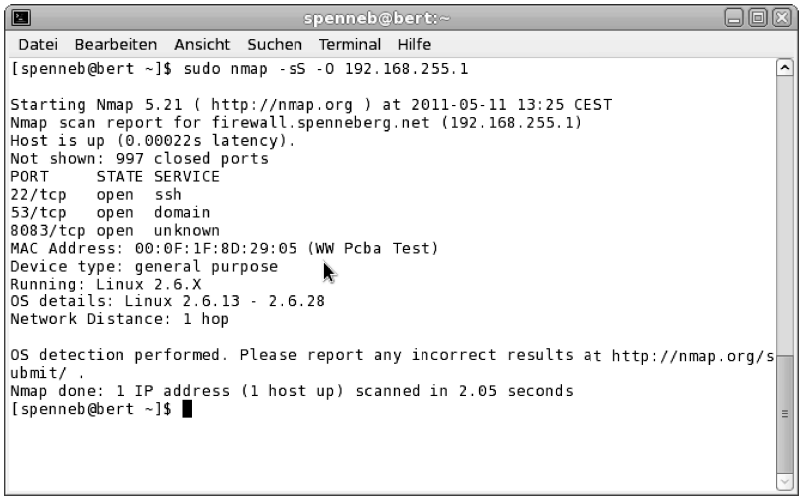


Abbildung 15.3: **Alle Funktionen von Nmap stehen nur auf der Kommandozeile zur Verfügung.**

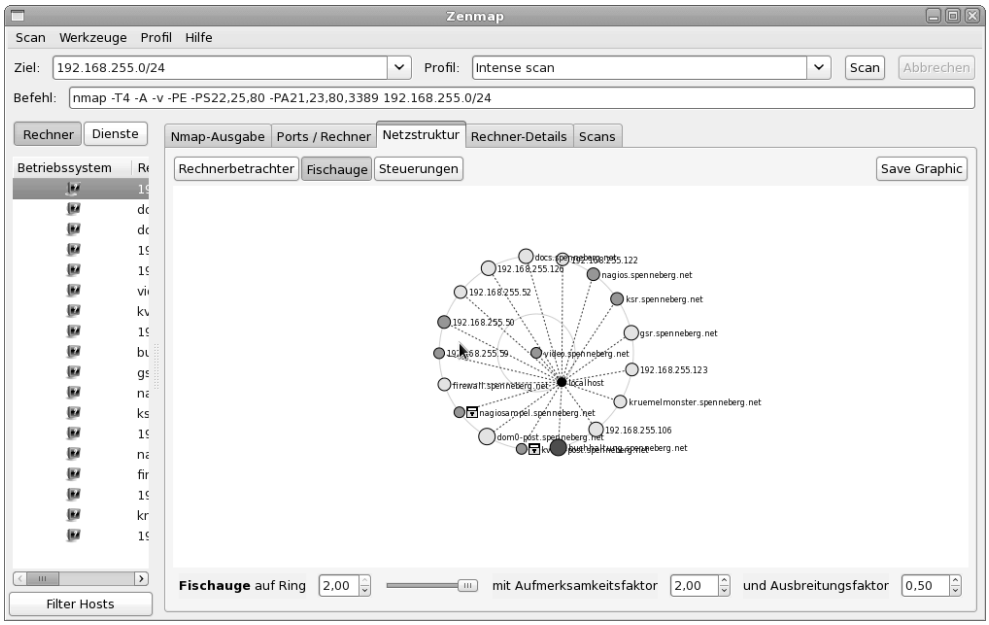


Abbildung 15.4: **Die grafische Oberfläche erleichtert den Einstieg.**

Bei der Installation werden zwei Befehle installiert:

» `nmap`: Hiermit können Sie auf der Kommandozeile einen Scan starten (siehe Abbildung 15.3).

» `Zenmap` und `nmapfe`: Dieses grafische Werkzeug erlaubt Ihnen die Konfiguration und den Start des Scans mit der Maus (siehe Abbildung 15.4).

Das grafische Nmap-Frontend `zenmap` ist sehr gut geeignet, um einen ersten Einstieg in die Verwendung von Nmap zu erhalten. Auch wenn ich eigentlich kein großer Freund von grafischen Oberflächen bin und der Kommandozeilenbefehl mehr Funktionen aufweist, besitzt die grafische Oberfläche die nette Eigenschaft, immer das entsprechende Nmap-Kommando anzuzeigen, das ausgeführt wird. Außerdem kann es eine Karte der gescannten Rechner anzeigen.

15.3.2 Einfache Scans

Nmap unterstützt eine Vielzahl von Netzwerk- und Rechnerscans. Am häufigsten wird Nmap eingesetzt, um festzustellen, welche Ports auf einem anderen Rechner geöffnet sind und ob diese Ports von einer Firewall geschützt werden. In diesem Abschnitt wollen wir uns mit dieser Form der Scans beschäftigen. Die fortgeschrittenen Scans werden Sie in dem nächsten Abschnitt kennenlernen.

Nmap kann sowohl einen TCP-Port-Scan als auch einen UDP-Port-Scan durchführen. Selbst bei dem TCP-Port-Scan gibt es verschiedene Varianten, wie dieser Port-Scan durchgeführt werden kann. Am einfachsten ist der TCP-Connect-Scan.

TCP-Connect-Scan

Bei dem TCP-Connect-Scan (`nmap -sT`) versucht Nmap, zu jedem Port des Zielrechners eine vollständige TCP-Verbindung aufzubauen. Dabei wird ein kompletter TCP-Handshake durchlaufen. Dieser Scan liefert Ihnen das sicherste Ergebnis, da hier die Verbindung tatsächlich so aufgebaut wird, wie auch ein Client sie aufbauen würde. Dies ist auch der einzige TCP-Scan, den Nmap ohne root-Privilegien durchführen kann, da dieser Scan ohne Umgehung des TCP/IP-Stapels durchgeführt werden kann.

```
# nmap -sT www.spenneberg.com
Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-23 15:04 ↵
        CEST
Interesting ports on mail.spenneberg.net (217.160.128.61):
(The 1661 ports scanned but not shown below are in state: closed)
PORT    STATE    SERVICE
21/tcp  open     ftp
22/tcp  open     ssh
23/tcp  filtered telnet
25/tcp  open     smtp
53/tcp  open     domain
```

```
80/tcp  open     http
443/tcp open     https
993/tcp open     imaps

Nmap finished: 1 IP address (1 host up) scanned in 23.218 seconds
```

Nmap erkennt mit diesem Scan offene, geschlossene und von einer Firewall gefilterte Ports. Bei einem offenen Port kann Nmap die TCP-Verbindung aufbauen. Bei einem geschlossenen Port erhält Nmap bei dem Aufbau ein TCP-RST-Paket als Antwort, und bei einem von einer Firewall gefilterten Port erhält Nmap keine Antwort. Leider führt dieser Port-Scan zu Protokolleinträgen auf dem Zielsystem. Im Folgenden ist das Protokoll des Postfix-Mailservers während des Scans zu sehen:

```
Sep 23 15:07:17 mail postfix/smtpd[9259]: connect from p508F484E.dip.t- ↵
        dialin.net[80.143.72.78]
Sep 23 15:07:17 mail postfix/smtpd[9259]: lost connection after CONNECT  ↵
        from p508F484E.dip.t-dialin.net[80.143.72.78]
Sep 23 15:07:17 mail postfix/smtpd[9259]: disconnect from p508F484E.dip.t ↵
        -dialin.net[80.143.72.78]
```

Um nun mit Nmap Ihre Firewall-Regeln zu prüfen, sollten Sie mit Nmap Ihre Firewall scannen und das Ergebnis mit Ihren Regeln vergleichen. Falls sich hinter der Firewall weitere nicht genattete Systeme befinden, sollten Sie auch versuchen, diese IP-Adressen zu scannen.

INFO

Sobald eine Firewall im Spiel ist, können mindestens zwei verschiedene Phänomene auftreten:

Nmap bricht die Abarbeitung direkt ab. Dazu kommt es, weil Nmap vor Beginn des Scans versucht, den Rechner zu pingen. Wenn dies nicht erfolgreich ist, vermutet Nmap, dass der Rechner nicht online ist, und bricht des Scan ab. Durch die Angabe von `-P0` überspringen Sie den Ping. Zusätzlich kann es zu erhöhten Laufzeiten bei Nmap kommen. Eine Firewall verwirft möglicherweise die Pakete, die Nmap verschickt, und antwortet nicht darauf. Zur Sicherheit muss Nmap aber eine gewisse Zeit (Default 10 Sekunden) auf eine mögliche Antwort warten! Sie können die Nmap-Wartezeit mit der Option `--max_rtt_timeout` anpassen. Die Zeit geben Sie dabei in Millisekunden an. Um eine Sekunde als Timeout zu verwenden, nutzen Sie: `nmap -sT --max_rtt_timeout 1000 www.spenneberg.com`.

Außerdem beantworten einige Firewalls den TCP-Scan nicht mit TCP-RST-Paketen, sondern mit ICMP-Fehlermeldungen. So reagiert die Fedora-Core-3-Standard-Firewall auf einen SYN-Scan mit ICMP-Host-Prohibited-Fehlermeldungen. Da diese Meldungen zusätzlich in ihrer Geschwindigkeit beschränkt werden, dauert der Scan mehrere Stunden. Drücken Sie während des Aufrufs von Nmap einfach die Enter-Taste. Nmap zeigt ihnen dann an, wie weit der Scan fortgeschritten ist.

TCP-SYN-Scan

Eine Alternative zu dem Connect-Scan ist der SYN-Scan. Dieser Scan wird häufig auch als *Half-Open-Scan* bezeichnet, da er den TCP-Handshake nur zur Hälfte durchführt. Bei diesem

Scan sendet Nmap ein SYN-Paket an den Port auf dem Server. Wenn Nmap ein TCP-ACK-Paket als Antwort eines offenen Ports erhält, sendet Nmap direkt im Anschluss ein TCP-RST-Paket, um die Verbindung bereits wieder zu beenden (siehe Abbildung 15.5).

Einen geschlossenen Port erkennt Nmap an einer Fehlermeldung. Jedoch erkennt Nmap in diesem Modus, ob es sich um ein TCP-RST-Paket oder eine ICMP-Port-Unreachable-Meldung handelt. Während Nmap im Connect-Scan diesen Unterschied nicht erkennen kann, interpretiert es ein TCP-RST-Paket als geschlossenen Port, während es eine ICMP-Port-Unreachable-Meldung richtig als Ablehnung durch eine Firewall erkennt. Sobald Nmap eine ICMP-Port-Unreachable-Meldung oder keine Antwort erhält, zeigt es den Port als gefiltert an. Der folgende Scan ist auf derselben Maschine durchgeführt worden wie oben:

```
# nmap -sS www.spenneberg.com

Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-23 15:24 ↵
        CEST
Interesting ports on mail.spenneberg.net (217.160.128.61):
(The 1657 ports scanned but not shown below are in state: closed)
PORT     STATE    SERVICE
21/tcp   open     ftp
22/tcp   open     ssh
23/tcp   filtered telnet
25/tcp   open     smtp
53/tcp   open     domain
80/tcp   open     http
110/tcp  filtered pop3
143/tcp  filtered imap
443/tcp  open     https
993/tcp  open     imaps
3306/tcp filtered mysql

Nmap finished: 1 IP address (1 host up) scanned in 10.177 seconds
```

Sie können erkennen, dass Nmap nun die Ports 110, 143 und 3306 zusätzlich als gefiltert anzeigt. Bei dem Connect-Scan wurden diese Ports als geschlossen interpretiert. Nmap benötigt für diesen Scan jedoch *root*-Rechte.

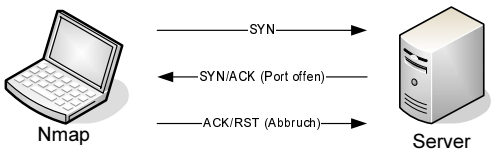


Abbildung 15.5: **Nmap bricht bei dem SYN-Scan die Verbindung ab, bevor es zur Protokollierung auf dem Zielsystem kommt.**

Wie Sie sehen, ist der SYN-Scan genauer als der Connect-Scan. Sie sollten daher Ihre Firewall und alle weiteren Systeme erneut mit dem SYN-Scan unter die Lupe nehmen. Falls Sie einen Port entdecken, der von der Firewall nicht geschützt wird, müssen Sie Ihr Firewall-Skript entsprechend anpassen.

Betriebssystem-Bestimmung

Sie möchten wissen, welches Betriebssystem auf einem entfernten System installiert ist? Nun, fragen Sie Nmap. Nmap kann mithilfe des TCP/IP-Stack-Fingerprinting sehr gut entscheiden, welches Betriebssystem sich auf der fraglichen Maschine befindet. Heute ist dies häufig eine der wenigen sicheren Methoden, das Betriebssystem zu ermitteln. Sicherlich sollten Sie zuvor prüfen, ob ein Banner-Grabbing Ihnen nicht einfacher und leichter diese Informationen zur Verfügung stellt:

```
# telnet www.spenneberg.com
Trying 217.160.128.61...
Connected to www.spenneberg.com (217.160.128.61).
Escape character is '^]'.
Fedora Core release 4 (Stentz)
Kernel 2.6.12-1.1398_FC4 on an i686
login:
```

Viele Administratoren ändern heute aber die Ausgabe der verschiedenen Dienste, sodass eine so ausführliche Ausgabe nur noch selten zu sehen ist. Dann hilft `nmap -O`:

```
# nmap -O www.spenneberg.com

Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-24 06:03 ↵
        CEST
Interesting ports on mail.spenneberg.net (217.160.128.61):
(The 1657 ports scanned but not shown below are in state: closed)
PORT     STATE    SERVICE
21/tcp   open     ftp
....
993/tcp  open     imaps
3306/tcp filtered mysql
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux 2.4.0 - 2.5.20
Uptime 282.906 days (since Wed Dec 15 07:19:50 2004)

Nmap finished: 1 IP address (1 host up) scanned in 17.989 seconds
```

Wie macht Nmap das? Nun, es verschickt verschiedene manuell erzeugte Pakete und beobachtet, wie das Zielsystem reagiert. Hierbei kommt es zu ganz charakteristischen Unterschieden. Zwar existieren Standards, die die TCP/IP-Protokolle beschreiben. Allerdings wer-

den diese Standards von unterschiedlichen Programmierern unterschiedlich verstanden und interpretiert. Einige TCP/IP-Stacks enthalten auch einfach Fehler in der Behandlung von Paketen und verletzen den Standard. Anschließend vergleicht Nmap die gewonnenen Daten mit den Informationen in einer Datenbank und kann so das Betriebssystem bestimmen.

INFO

Nmap muss nicht immer das Betriebssystem richtig bestimmen können. Teilweise, besonders bei neuen Betriebssystemen, kann Nmap in seiner Datenbank kein passendes Betriebssystem finden. Dann können Sie, falls Sie wissen, um welches Betriebssystem es sich handelt, diese Informationen an den Programmierer von Nmap weitergeben!

Der Operating-System-Scan (`-O`) kann nur von root verwendet werden, da zur Erzeugung der Pakete besondere Privilegien erforderlich sind. Außerdem muss das Zielsystem über mindestens einen offenen und einen geschlossenen Port verfügen. Nmap scannt zu Beginn alle Ports und wählt dann jeweils einen aus. Wenn Sie bereits einen offenen und einen geschlossenen Port kennen, können Sie diese auch bei dem Aufruf angeben und Nmap so einen kompletten Portscan ersparen:

```
# nmap -O -p25,26 www.spenneberg.com
Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-24 06:24 ↵
        CEST
Interesting ports on mail.spenneberg.net (217.160.128.61):
PORT   STATE  SERVICE
25/tcp open   smtp
26/tcp closed unknown
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux 2.4.0 - 2.5.20
Uptime 282.920 days (since Wed Dec 15 07:19:50 2004)

Nmap finished: 1 IP address (1 host up) scanned in 2.716 seconds
```

INFO

Sie können Nmap auch ein anderes Betriebssystem vorgaukeln. Für den Linux-Kernel 2.4 existiert der IP-Personality Patch (`http://ippersonality.sf.net`). Dieser Patch erlaubt es Ihnen, die Funktionen Ihres TCP/IP-Stacks so zu modifizieren, dass er einem anderen Stack entspricht. Für den Kernel 2.6 kenne ich keine entsprechende Funktion.

Wenn Sie die Bestimmung des Betriebssystems betrachten, werden Sie feststellen, dass Nmap sogar die Uptime des gescannten Rechners ermitteln kann. Dies kann für einen potenziellen Angreifer eine sehr interessante Information sein. Ein Microsoft Windows-System, das seit über einem Jahr nicht neu gebootet wurde, kann kaum in der Zeit wesentlich gepatcht worden sein. Nach dem Einspielen vieler Patches oder Servicepacks ist ein Reboot erforderlich. Nmap verwendet für diese Technik die TCP-Timestamps aus dem RFC1323. Diese Technik wurde von den Nmap-Entwicklern von der Webpage `http://uptime.netcraft.com` übernommen.

INFO

Wenn Sie den Timestamp-Scan Ihrer Systeme unterbinden möchten, können Sie unter Linux die Timestamps (`tcp_timestamps`) sehr leicht abschalten:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

Wenn Sie möchten, dass diese Einstellung nach jedem Reboot aktiv ist, tragen Sie die Zeile `net.ipv4.tcp_timestamps=0` in der Datei `/etc/sysctl.conf` Ihrer Distribution ein. Dies kann sich jedoch nachteilig auf die Performance auswirken. Diese Werte werden für genauere Berechnung der Laufzeit (Round-Trip) genutzt. Hieraus leiten sich die TCP-Retransmission-Zeiten ab.

UDP-Scan

Viele Administratoren verwenden lediglich die TCP-Scans von Nmap. Dabei kann Nmap noch viele weitere Scans durchführen. Der interessanteste Scan ist der UDP-Scan. Es gibt viele Dienste, die über einen UDP-Port erreichbar sind. Im Gegensatz zu TCP besteht bei UDP sehr leicht die Möglichkeit, gespoofte Pakete zu senden.

INFO

Natürlich ist es auch ein Leichtes, ein gespooftes TCP-Paket zu senden. Jedoch müssen Sie erst einen kompletten TCP-Handshake durchlaufen, bevor der Dienst Daten entgegennimmt. Das ist bei UDP anders. Sie versenden ein gespooftes UDP-Paket, und die Daten werden direkt an den Dienst weitergereicht. Handelt es sich um einen verwundbaren UDP-Dienst, so können Sie möglicherweise mit einem Paket bereits den kompletten Angriff durchführen. Eine sehr eindrucksvolle Demonstration war der SQL-Slammer-Wurm, der die Microsoft-SQL-Engine auf dem Port 1434/udp mit einem 376 Byte großen Paket angriff und die Kontrolle übernahm (`http://www.cert.org/advisories/CA-2003-04.html`).

Leider steht der UDP-Scan (`-sU`) nur dem Benutzer root zur Verfügung. Außerdem kann es bei dem UDP-Scan zu sehr langen Nmap-Laufzeiten kommen, wenn eine Firewall den Zugriff auf die Ports unterdrückt. Hier hilft auch die Option `--max_rtt_timeout`. Um nun einen UDP-Scan durchzuführen, verwenden Sie den folgenden Befehl:

```
nmap -sU www.spenneberg.com
```

Hier kommt es im Zusammenhang mit Firewalls immer wieder zu Verwirrungen. Wenn eine Firewall mit einer DROP-Regel den Zugriff auf einen UDP-Port verwehrt, wird Nmap diesen Port als offen kennzeichnen. Um dies zu verstehen, müssen Sie wissen, wie Nmap UDP-Ports scannt. Hierzu sendet es ein UDP-Paket an den fraglichen Port des Zielrechners. Erhält Nmap ein ICMP-Port-Unreachable zurück, ist der Port geschlossen. Sobald Nmap aber ein UDP-Paket oder keine Antwort zurückerhält, muss Nmap den Port als offen deklarieren. Die Tatsache, dass Nmap keine Antwort erhält, kann auch in einer fehlerhaften Anfrage an einen laufenden Dienst begründet sein. Wundern Sie sich also nicht, wenn Nmap offene UDP-Ports auf Ihrem System nennt.

15.3.3 Fortgeschrittene Anwendung

(TCP/Ping-)Sweeps

Bevor Sie mit der Überprüfung Ihrer Firewall-Regeln oder einer Analyse eines Netzwerks beginnen, sollten Sie ermitteln, welche Rechner überhaupt sichtbar sind. Vielleicht haben Sie von Ihrem Provider ein Klasse-C-Netzwerk zugeordnet bekommen. Diese IP-Adressen nutzen Sie nun in Ihrer Firewall und Ihrer DMZ. Natürlich kennen Sie diese IP-Adressen, aber Sie möchten prüfen, welche Rechner von außen ermittelt werden können. Hierfür eignen sich die Sweep-Scans. Diese fegen (*sweep*) durch ein Netzwerk und probieren dabei jede IP-Adresse aus.

Der einfachste Sweep-Scan ist der Ping-Sweep. Hier sendet Nmap an jede IP-Adresse eines Netzwerks ein Echo-Request-Paket und wartet auf die Echo-Reply-Antwort. Da einige Systeme Ping-Pakete blockieren, sendet Nmap zusätzlich auch ein TCP-ACK-Paket an den Port 80/tcp des Zielsystems. Wenn das Zielsystem erreichbar ist, antwortet es mit einem TCP-RST.

```
# nmap -sP 192.168.0.0/24

Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-24 07:08 ↵
        CEST
Host 192.168.0.1 appears to be up.
MAC Address: 00:00:B4:B0:12:10 (Edimax Computer Company)
Host 192.168.0.108 appears to be up.
Host 192.168.0.254 appears to be up.
MAC Address: 00:50:18:1B:BC:A0 (AMIT)
Nmap finished: 256 IP addresses (3 hosts up) scanned in 6.364 seconds
```

Sobald sich jedoch eine Firewall zwischen Ihnen und dem Zielnetzwerk befindet, besteht die Gefahr, dass dieser Scan nicht funktioniert, da die Firewall Ping-Pakete unterbindet und ein TCP-ACK-Paket von einer zustandsorientierten (*stateful*) Firewall nur akzeptiert wird, wenn vorher eine TCP-Verbindung aufgebaut wurde. Für diese Zwecke können Sie mit der Option `-P` die für den „Ping“ zu verwendenen Pakete angeben!

- `-P0`: Kein Ping. Dies ist sinnvoll, wenn Sie wissen, dass ein Rechner online ist und Sie direkt einen Portscan starten möchten.
- `-PA25,80`: Sendet TCP-ACK-Pakete an die Ports 25 und 80. Wenn das System erreichbar ist, sendet es ein TCP-RST-Paket als Antwort. Eine zustandsorientierte Firewall erlaubt das TCP-ACK-Paket nicht.
- `-PS25,80`: Diese Option sendet ein SYN-Paket an den angegebenen Port. Damit besteht häufig die Möglichkeit, durch eine zustandsorientierte Firewall hindurch einen Rechner zu erreichen. Die Firewall öffnet häufig die Ports 25 und 80, um SMTP- und HTTP-Verkehr zu erlauben. Weitere gute Ports zum Test sind 21 (FTP), 22 (SSH) und 443 (HTTPS).
- `-PU54,222`: Hiermit können Sie UDP-Pakete für den Ping nutzen. Wenn der Port geschlossen ist und der Rechner erreichbar ist, erwartet der Scan ein ICMP-Port-Unreachable. Bei einem offenen Port erhält man häufig keine Antwort, da der Dienst die Anfrage nicht ver-

steht. Daher sollten Sie Ports verwenden, die üblicherweise geschlossen sind (z. B. 54 und 222).

» `-PE`: Wenn Sie diese Option verwenden, führt Nmap einen echten Ping-Scan durch. Damit können auch Broadcast-Adressen in dem Zielnetzwerk gefunden werden. Viele Systeme (vor allem UNIX, kein Windows) reagieren auf einen Broadcast-Ping mit einer Antwort. Diese Broadcast-Adressen können dann für Denial-of-Service-Angriffe à la SMURF (`http://www.cert.org/advisories/CA-1998-01.html`) missbraucht werden.

» `-PP`: Diese Option verwendet ICMP-Timestamp-Request-Pakete für den Scan. Einige Systeme antworten auf diesen Scan mit einem ICMP-Timestamp-Reply. Daher kann dieser Scan anstelle eines Ping genutzt werden. Linux-Systeme ignorieren diese Anfrage.

» `-PM`: Diese Option verwendet einen ICMP-Netmask-Request für den Scan.

» `-PB`: Diese Option verwendet die Default-Nmap-Ping-Kombination aus einem Echo-Request- und einem TCP-ACK-Paket (siehe oben).

Wenn Sie durch eine zustandsorientierte Firewall die sich dahinter befindlichen Systeme auskundschaften möchten, ist es am einfachsten, die Option `-PS` zu verwenden. Allerdings sollten Sie sich Gedanken über die zu testenden Ports machen. Sinnvoll sind 22 (SSH), 21 (FTP), 25 (SMTP), 80 (HTTP), 110 (POP3), 143 (IMAP) und 443 (HTTPS). Damit erreichen Sie auch Systeme durch eine Firewall, die den Port 80 nur bei einer Verbindung zum Webserver erlaubt, aber nicht bei einer Verbindung zu einem Mailserver (siehe Abbildung 15.6).

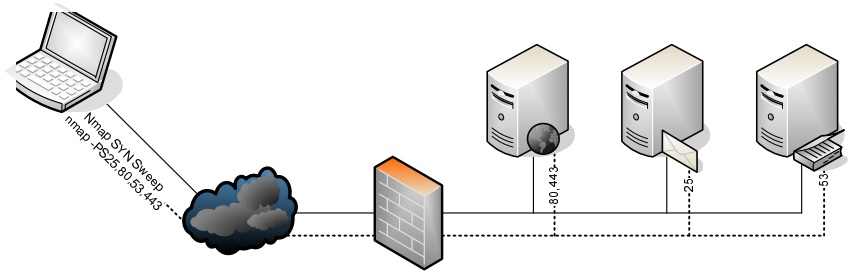


Abbildung 15.6: **Nmap kann durch eine zustandsorientierte Firewall die in der DMZ stehenden Rechner ermitteln.**

Protokoll-Scan

Sehr interessant ist häufig auch der Protokoll-Scan (`-sO`). Er kann sehr viel über ein System aussagen. Hier führt Nmap einen Scan der möglichen IP-Protokolle (`/etc/protocols`) durch. Bei einem normalen System sieht man üblicherweise nur die Protokolle ICMP, TCP und UDP:

```
# nmap -sO www.spenneberg.com

Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-24 16:01 ↵
        CEST
Interesting protocols on mail.spenneberg.net (217.160.128.61):
(The 253 protocols scanned but not shown below are in state: open| ↵
        filtered)
```

```
PROTOCOL STATE    SERVICE
1        open     icmp
6        open     tcp
17       filtered udp

Nmap finished: 1 IP address (1 host up) scanned in 6.099 seconds
```

Sobald das System aber auch über IPv6-Funktionalität verfügt, taucht mindestens auch dieses Protokoll in der Liste auf:

```
# nmap -sO localhost

Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-24 16:04 ↵
        CEST
Interesting protocols on bibo.spenneberg.de (127.0.0.1):
(The 250 protocols scanned but not shown below are in state: closed)
PROTOCOL STATE         SERVICE
1        open          icmp
2        open|filtered igmp
6        open          tcp
17       open          udp
41       open|filtered ipv6
```

Sehr interessant ist ein Protokoll-Scan eines VPN-Gateways. Dann tauchen zusätzlich in der Liste die Protokolle ESP und AH auf. Diese IPsec-Protokolle tauchen auch auf einem Linux-System erst auf, wenn die VPN-Funktionalität tatsächlich genutzt wird. So können Sie also mögliche VPN-Zugangssysteme erkennen.

```
# nmap -sO schulung.spenneberg.net

Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-24 16:08 ↵
        CEST
Interesting protocols on schulung.spenneberg.net (x.y.z.a):
(The 248 protocols scanned but not shown below are in state: closed)
PROTOCOL STATE         SERVICE
1        open          icmp
2        open|filtered igmp
6        open          tcp
17       open          udp
41       open|filtered ipv6
50       open|filtered esp
51       open|filtered ah
```

So können Sie auch prüfen, ob Ihre Firewall die entsprechenden IP-Protokolle richtig filtert.

Weitere Stealth-Scans

Wenn der Port-Scan möglichst unentdeckt bleiben soll oder Sie die Protokollierung Ihrer Firewall, die Auswertung der Protokolle oder Ihr Intrusion-Detection-System testen wollen, dann sollten Sie auch die Stealth-Scans von Nmap kennen. Diese Scans sollen unbemerkt von einem Administrator durchgeführt werden können. Allerdings erkennen viele IDS mindestes den Xmas- und den Null-Scan. Nmap bezeichnet drei Scans als Stealth-Scan:

» `-sF`: Der TCP-FIN-Scan sendet ein Paket, in dem nur das TCP-FIN-Flag gesetzt ist.

» `-sX`: Der Xmas-Scan sendet ein Paket, in dem das TCP-FIN, -URG und -PSH-Flag gesetzt sind. Im Gegensatz zur weitläufigen Meinung sind hier nicht alle Flags gesetzt, sondern nur sehr viele. Daher kommt auch der Name: Alle Flags brennen wie die Kerzen an einem Christbaum zu Weihnachten (*Xmas*).

» `-sN`: Der Null-Scan sendet ein Paket, in dem kein TCP-Flag gesetzt ist.

Bei allen diesen Scans muss das Zielsystem laut RFC793 bei einem offenen Port das Paket ignorieren. Ein geschlossener Port muss ein TCP-RST-Paket als Antwort generieren. Sie können mit diesen Scans also erkennen, ob ein Port geschlossen ist. Leider können Sie nicht sicher sagen, ob ein Port tatsächlich offen ist, da auch eine Firewall das Paket verworfen haben kann, sodass es nicht zur Antwort kam! Nmap listet daher die Ports als `open|filtered` auf:

```
# nmap -sF www.spenneberg.com

Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-24 16:35 ↵
        CEST
Interesting ports on mail.spenneberg.net (217.160.128.61):
(The 1657 ports scanned but not shown below are in state: closed)
PORT     STATE         SERVICE
21/tcp   open|filtered ftp
22/tcp   open|filtered ssh
23/tcp   open|filtered telnet
25/tcp   open|filtered smtp
53/tcp   open|filtered domain
80/tcp   open|filtered http
110/tcp  filtered      pop3
143/tcp  filtered      imap
443/tcp  open|filtered https
993/tcp  open|filtered imaps
3306/tcp filtered      mysql
```

Dass in dem obigen Fall doch einige Ports als definitiv gefiltert angezeigt wurden, hängt mit den Firewall-Regeln zusammen. Diese Ports werden von der Firewall mit einem ICMP-Port-Unreachable abgelehnt. Da das echte Zielsystem ein TCP-RST-Paket erzeugen würde, muss die ICMP-Nachricht von einer Firewall stammen. Der Port wird also gefiltert. Für mich zählt der TCP-ACK-Scan (`-sA`) auch zu den Stealth-Scans, denn ein IDS protokolliert üblicherweise

den Scan nicht, und Sie können mit ihm feststellen, ob ein Port tatsächlich offen ist oder von einer Firewall gefiltert wurde:

```
# nmap -sA www.spenneberg.com

Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-24 16:38 ↵
        CEST
Interesting ports on mail.spenneberg.net (217.160.128.61):
(The 1664 ports scanned but not shown below are in state: UNfiltered)
PORT     STATE    SERVICE
23/tcp   filtered telnet
110/tcp  filtered pop3
143/tcp  filtered imap
3306/tcp filtered mysql
```

Dieser Scan sendet ein TCP-ACK-Paket an den entsprechenden Port. Wird der entsprechende Port gefiltert, erwarten wir keine Antwort. Wird der entsprechende Port aber nicht gefiltert, erwarten wir in jedem Fall (Port offen oder geschlossen) immer ein TCP-RST-Paket als Antwort. Nmap zeigt nun, dass die Ports 23, 110, 143 und 3306 tatsächlich gefiltert werden. Das bedeutet in Kombination mit dem TCP-FIN-Scan, dass die Ports 21, 25, 53, 80, 443 und 993 offen sein müssen. Sie können also, ohne einen TCP-SYN-Scan einzusetzen, ermitteln, ob ein Port offen oder geschlossen ist oder durch eine Firewall gefiltert wird.

INFO

Dies funktioniert nur richtig bei einer zustandslosen Firewall wie `ipchains`. Eine richtig konfigurierte zustandsorientierte Firewall wird die TCP-ACK- und TCP-FIN-Pakete immer filtern und nicht passieren lassen. Nur eine zustandslose Firewall muss diese Pakete für offene Ports passieren lassen, da sie nicht erkennen kann, ob die Pakete zu aufgebauten Verbindungen gehören.

15.4 Decoy-Scan

Der Decoy-Scan ist sicherlich auch eher ein Scan für den Test eines Intrusion-Detection-Systems als für den Test einer Firewall. Aber er kann eingesetzt werden, um die Analyse der Firewall-Protokolle zu testen. Hierbei handelt es sich um eine Ergänzung eines normalen Scans. Der Decoy-Scan kann mit allen Scans, außer dem RPC-Scan, eingesetzt werden. Dabei definieren Sie mit der Option `-D` mehrere IP-Adressen, die Nmap als Decoy (Tarnung) verwenden soll. An beliebiger Stelle in der kommaseparierten Liste fügen Sie `ME` als Platzhalter für die eigene IP-Adresse ein. Nmap führt dann entsprechend viele Scans durch und fälscht dabei die Absenderadressen in den verschiedenen Scans. Wenn das gescannte System diese Scans protokolliert, hat es den Anschein, als ob das System gleichzeitig von verschiedensten Orten gescannt wird. Das Opfer kann kaum ermitteln, welche Ihre IP-Adresse ist, und den Scan daher nicht zurückverfolgen. Sie müssen Ihre Adresse natürlich angeben, denn sonst erhält Nmap keine Rückmeldung und kann Ihnen nicht das Ergebnis anzeigen. Versäumen Sie die Angabe der Position mit `ME`, wählt Nmap zufällig eine Position in der IP-Adressliste.

15.5 Banner-Grabbing

Sicherlich ist es ganz interessant, welche Ports auf einem System geöffnet sind. Die Tatsache, dass der Port 25/tcp offen ist, sagt bereits eine ganze Menge über das System aus: Es empfängt E-Mail, wahrscheinlich versendet es sie auch. Möglicherweise ist es ein Relay, das die E-Mail weiter nach innen versendet. Wahrscheinlich kann das System auch eine DNS-Auflösung durchführen. Um aber zu erkennen, ob sich dahinter eine echte Sicherheitslücke befindet, benötigen Sie mehr Informationen, und auch der potenzielle Angreifer, der durch Ihre Firewall auf Ihren Mailserver zugreift, wird versuchen, mehr Informationen zu erhalten. Am einfachsten erhält er die Informationen, indem er sich mit dem Port verbindet und versucht, den Dienst zur Ausgabe seines Banners zu überreden. Bei einem Mailserver erhalten Sie die Informationen automatisch, bei einem Webserver müssen Sie tatsächlich zunächst nach Informationen fragen, bevor Sie sein „Banner" erhalten. Nmap kann diese Aufgabe für Sie übernehmen. Nmap nennt diesen Scan den Version-Detection-Scan (`-sV`). Nmap ist häufig in der Lage, das Applikationsprotokoll (ftp, ssh, telnet etc.), den Namen der Applikation (Apache, Solaris, telnetd etc.), dessen Version und sogar weitere Details zu ermitteln. Verlangt der Dienst eine Verbindung über Secure Socket Layer (SSL), so nutzt Nmap auch OpenSSL, falls die Unterstützung bei der Übersetzung aktiviert wurde.

```
# nmap -sV www.spenneberg.com

Starting nmap 3.93 ( http://www.insecure.org/nmap/ ) at 2005-09-24 17:12 ↵
        CEST
Interesting ports on mail.spenneberg.net (217.160.128.61):
(The 1657 ports scanned but not shown below are in state: closed)
PORT     STATE     SERVICE  VERSION
21/tcp   open      ftp      vsFTPd 1.2.1
22/tcp   open      ssh      OpenSSH 3.6.1p2 (protocol 1.99)
23/tcp   filtered  telnet
25/tcp   open      smtp     Postfix smtpd
53/tcp   open      domain   ISC Bind 9.2.4
80/tcp   open      http     Apache httpd 2.0.46 ((Red Hat))
110/tcp  filtered  pop3
143/tcp  filtered  imap
443/tcp  open      ssl/http Apache httpd 2.0.46 ((Red Hat))
993/tcp  open      ssl/imap Dovecot imapd
3306/tcp filtered  mysql    Service
Info: Host: mail.spenneberg.net; OS: Unix

Nmap finished: 1 IP address (1 host up) scanned in 40.872 seconds
```

Wenn Sie Ihre Systeme prüfen, sollten Sie bei einer Ausgabe wie der obigen überlegen, ob Sie die Banner der Dienste nicht anpassen wollen, sodass die Ausgabe weniger Informationen enthält. Mindestens die Angabe der Versionsnummern sollte bei den Diensten, bei denen es

möglich ist, unterbunden werden. Ein Beispiel hierfür ist der Dovecot-`imapd` oder der Postfix-`smtpd` in dem obigen Beispiel.

15.6 Idle-Scan

Der Idle-Scan wurde zuerst im Dezember 1998 von dem Hacker Antirez für sein Werkzeug `hping2` (`http://www.hping.org`) entwickelt. Hierbei führen Sie einen Portscan mit gespoofter Absenderadresse durch. Eigentlich ist das nicht möglich, denn für einen Portscan benötigen Sie ja eine Rückmeldung von dem Zielsystem, um zu unterscheiden, ob der Port offen oder geschlossen ist. Bei diesem Scan fälschen Sie jedoch Ihre Absenderadresse, sodass diese Antwort an ein anderes System geschickt wird. Bei dem Decoy-Scan konnte das funktionieren, da auch noch ein Scan mit der eigenen Adresse zusätzlich zu den gefälschten Adressen erfolgte. Das ist hier nicht der Fall.

Wie funktioniert das? Betrachten Sie die Abbildung 15.7. Dort sehen Sie ein Opfer (rechts), den Rechner des Angreifers (unten) und einen weiteren Rechner (links). Dieser weitere Rechner ist der Zombie oder Silent Host. In dem Scan verwendet der Angreifer dessen IP-Adresse als Absender-IP-Adresse.

Der Angreifer sendet nun mehrere TCP-SYN-Pakete an einen Port (1). CE[X] Ist der Port offen, so sendet das Opfer ein TCP-SYN-ACK-Paket an den Zombie (2a). Ist der Port geschlossen, so sendet das Opfer ein TCP-RST-Paket an den Zombie (2b). Wie reagiert nun der Zombie? Auf das TCP-RST-Paket (2b) erfolgt keine Reaktion (3b). Auf das TCP-SYN-ACK-Paket (2a) reagiert der Zombie aber mit einem TCP-RST-Paket, da er die Verbindung nicht kennt (3a). Also, halten wir fest: Ist der Port auf dem Opfer offen, versendet der Zombie ein Paket. Ist der Port geschlossen, ist das nicht der Fall. Der Angreifer muss nun also herausfinden, ob der Zombie ein Paket versendet. Daraus kann er schließen, ob der Port offen oder geschlossen ist.

Hierzu nutzt der Angreifer das IP-Identifikationsfeld im IP-Header der Pakete. Dieses Feld wird von fast allen Betriebssystemen gesetzt, aber nur ausgewertet, wenn Pakete fragmentiert werden. Dann erkennt der Empfänger der Fragmente zusammengehörige Fragmente an der identischen Nummer in diesem Feld. Es gibt keinen weiteren regulären Nutzen für dieses Feld. Die meisten TCP/IP-Stacks zählen daher diese Nummer einfach hoch. Jedes Paket, das

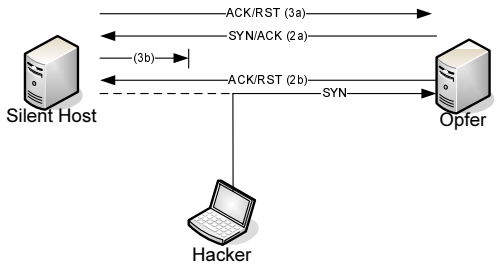


Abbildung 15.7: **Bei dem Idlescan nutzt der Angreifer einen dritten Rechner in dem Scan.**

CE[X] Bitte prüfen: Im Bild ist keine (1) markiert.

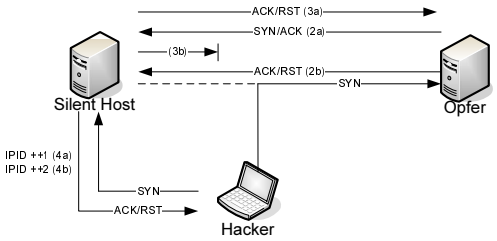


Abbildung 15.8: **Bei dem Idlescan hängt die Identifikationsnummer der Pakete des Zombies von dem Zustand des Ports auf dem Opfer ab.**

sie versenden, erhält eine um eins hochgezählte Nummer. Sobald sie bei 65535 angekommen sind, fangen sie wieder bei null an.

INFO

Die Linux-Kernel 2.4 und 2.6 setzen dieses Feld auf null, sobald die Path Maximum Transmission Unit Discovery (PMTU-Discovery) eingeschaltet ist. Da nun kein Paket mehr fragmentiert werden darf, wird dieses Feld nicht benötigt.

OpenBSD erzeugt zufällige Nummern in diesem Feld. Fragmente können dann immer noch an der gleichen Nummer erkannt werden.

Der Angreifer sucht nun als Zombie einen Rechner, der so die Identifikationsnummer hochzählt. Eine weitere Bedingung für dieses System ist seine Leblosigkeit. Das System sollte keinen weiteren Netzwerkverkehr erzeugen. Es sollte online sein, aber tot (Zombie, Silent Host).

Nun beginnt der Angreifer, diesem Rechner einmal pro Sekunde ein TCP-SYN-Paket auf einem geschlossenen Port zu senden. Der Zombie reagiert einmal pro Sekunde mit einem TCP-RST-Paket. Die Identifikationsnummer in den Paketen wird immer um 1 hochgezählt (4a, siehe Abbildung 15.8). Gleichzeitig führt der Angreifer den gespooften Portscan einmal pro Sekunde auf sein Opfer aus. Ist der Port offen, so muss der Zombie einmal pro Sekunde ein weiteres Paket versenden. Die Identifikationsnummer in dem TCP-SYN-Scan (4b) wird nun immer um zwei hochgezählt. War der Port geschlossen, so ändert sich an der Differenz zweier Identifikationsnummern nichts.

Während dieser Scan mit dem originalen Werkzeug `hping2` durchaus umständlich war, ist dies mit Nmap sehr einfach. Sie geben lediglich mit der Option `-sI <zombie host[:probeport]>` den Zombie-Rechner und optional einen Port an. Den Rest erledigt Nmap automatisch.

STOP

Denken Sie daran, mit der Option `-P0` den Ping vor dem Scan abzuschalten. Dieser Ping wird sonst mit Ihrer echten IP-Adresse durchgeführt!

15.6.1 Modifikation des Client-Ports, Time-Tuning und Fragmentierung

Häufig werden Sie einen Port-Scan einer Firewall durchführen, bei dem Sie anschließend das Gefühl haben, dass alles in Ordnung sei. Kein Port ist offen. Dennoch finden Sie anschließend

Hinweise, dass ein Angreifer doch durch die Firewall gekommen ist, oder bei einem Penetrationstest findet eine andere Person doch Lücken in Ihrer Firewall. Wie kann das sein?

Die Probleme, die ich hier beschreibe, betreffen Sie nicht, wenn Sie Ihre Regeln so entwickelt und aufgesetzt haben, wie in diesem Buch beschrieben wird. Sie können aber davon betroffen sein, wenn Sie noch `ipchains` als Firewall einsetzen oder Ihre `Iptables`-Regeln nicht das Connection Tracking nutzen.

Bei `ipchains`-Regelwerken finden Sie häufig die folgenden Regeln:

```
# DNS
ipchains -A forward -s $INTNETZ -p udp --destination-port 53 -j ACCEPT
ipchains -A forward -d $INTNETZ -p udp --source-port 53 -j ACCEPT

# Aktives FTP
ipchains -A forward -d $INTNETZ -p tcp --source-port 20 -j ACCEPT
ipchains -A forward -s $INTNETZ -p tcp --destination-port 20 -j ACCEPT
```

Die ersten beiden Regeln erlauben UDP-Pakete an den Port 53 im Internet und den Rückweg. Damit sollen DNS-Anfragen ermöglicht werden. Die nächsten beiden Regeln erlauben Verbindungen von dem Port 20/tcp in das interne Netz. Hiermit werden aktive FTP-Verbindungen erlaubt.

STOP

Leider gibt es bei dem Einsatz von `ipchains` keine wesentlich bessere Alternative für die DNS-Regeln. Die Regeln für das aktive FTP sind aber grob fahrlässig. Bei Verwendung von `ipchains` müssen Sie aktives FTP verbieten. Dies öffnet zu viele Sicherheitslücken. Nutzen Sie stattdessen passives FTP. Bei richtiger Anwendung von `iptables` besteht hier aber kein Sicherheitsproblem.

Diese Regeln erlauben Pakete von außen in das interne Netz `$INTNETZ`. Sie müssen nur als Absender den richtigen Port verwenden! Bei Nmap können Sie den Source-Port mit der Option `--source_port` angeben. Sinnvoll sind hier die Ports 53/udp, 20/tcp und 53/tcp. Bei dem oben angegebenen Regelwerk könnten Sie damit jeden Rechner hinter der Firewall scannen!

Auch ein Spoofing des Scans wie bei dem Idle-Scan kann Ihnen mehr Informationen liefern, als Sie normalerweise erhalten würden. Viele Regelsätze erlauben nur bestimmten Rechnern, durch die Firewall auf interne Rechner zuzugreifen. Wenn Sie derartige Vertrauensverhältnisse vermuten oder in Ihrer eigenen Firewall testen möchten, können Sie den Idle-Scan hierfür verwenden! Geben Sie als Zombie einfach den „vertrauten" Rechner an. Der Scan muss natürlich durchgeführt werden, wenn der Zombie keinen weiteren Netzwerkverkehr erzeugt (nachts).

Zwei weitere Optionen in Nmap versuchen, den Scan vor möglichen Erkennungssystemen zu verstecken. Die Option `-f` fragmentiert die Pakete vor dem Versand in kleine Fragmente. Viele Firewalls defragmentieren diese zwar vor ihrer Analyse, aber häufig übersehen gerade einfache Firewalls und IDS die Fragmente und beachten sie nicht. Mithilfe der Fragmentierung können daher diese Systeme umgangen werden.

STOP

Wenn Sie Linux als scannendes Betriebssystem einsetzen, müssen Sie darauf achten, dass auf Ihrem System keine `iptables`-Firewall mit Connection Tracking aktiv ist. Diese Firewall defragmentiert auch die ausgehenden Pakete! Sie können dann keine fragmentierten Pakete versenden.

Einen versteckten Port-Scan können Sie häufig auch durchführen, indem Sie das Timing des Port-Scans verändern. Hierzu bietet Nmap die Option `-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane>`, mit der Sie sehr einfach das Timing einstellen können. Viele IDS erkennen einen Port-Scan nur an einer bestimmten Häufigkeit und Geschwindigkeit. Natürlich können Sie bei Nmap auch die Zeiten einzeln einstellen. Die Manpage gibt Ihnen hier weitere Informationen.

15.6.2 Ausgabeformate

Wenn Sie häufiger derartige Scans durchführen, werden Sie auch unterschiedliche Ausgabeformate testen wollen. Nmap kann zunächst die Ausgabe mit `-oN` (Normal), `-oG` (Grepable), `-oX` (XML), `-oA` (Alle: NGX) und `-oS` (Skript-Kiddie-Sprache) in einer Datei protokollieren. Sie können sogar einen abgebrochenen Scan, der mit `-oN` oder `-oG` protokolliert wurde, mit `--resume` wieder aufnehmen.

Um die XML-Ausgabe später darstellen zu können, liefert Nmap auch ein XSL-Stylesheet mit (siehe Abbildung 15.9).

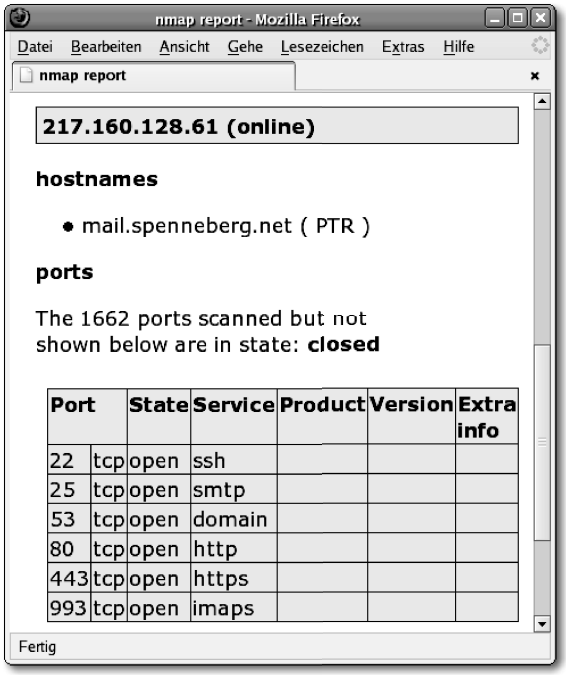


Abbildung 15.9: **Nmap kann seine Ergebnisse in XML formatieren. Dieses können Sie zum Beispiel in Firefox anzeigen.**

15.6.3 Nmap-Hilfswerkzeuge

Nmap ist ein sehr gutes Werkzeug, um Audits durchzuführen. Sie erkennen neue Systeme in Ihrem Netzwerk. Sobald ein neuer Port und damit ein neuer Dienst auftaucht, zeigt Ihnen Nmap diese Information an. Wenn Sie aber schon Nmap für 20 oder sogar 200 Rechner gestartet haben, werden Sie wissen, dass die Ausgabe von Nmap nicht gerade die manuelle Verarbeitung erleichtert. Speziell bei einem wiederholenden Einsatz von Nmap ist das maschinelle Verarbeiten und Vergleichen eine Arbeitsersparnis. Ich werde Ihnen im Weiteren einige Werkzeuge vorstellen, die mir dabei das Leben vereinfachen. Es gibt sicherlich noch weitere, vielleicht auch bessere Werkzeuge. Ich würde mich freuen, wenn Sie mir eine E-Mail schreiben würden, sollten Sie ein derartiges Werkzeug kennen.

NDiff

NDiff ist ein Werkzeug, um die Unterschiede in zwei zeitlich versetzten Scans zu ermitteln. Ndiff ist seit der Version 4.85 Bestandteil von Nmap und liefert wertvolle Informationen. Ndiff enthält Werkzeuge zur Erzeugung von Baselines, erlaubt den täglichen vergleichenden Aufruf von Nmap und stellt die Ergebnisse in einer übersichtlichen Tabelle dar.

Die Manpage enthält sehr viele Beispiele, sodass ich hier auf sie verweisen möchte. Das folgende Beispiel ist aus dieser Manpage kopiert.

Um zwei Nmap-Läufe miteinander zu vergleichen, erzeugen Sie zunächst zwei XML-Nmap-Protokolle, die Sie anschließend vergleichen.

```
# nmap -F scanme.nmap.org -oX scanme-1.xml
# nmap scanme.nmap.org -oX scanme-2.xml
$ ndiff -v scanme-1.xml scanme-2.xml
           -Nmap 4.90RC2 at 2009-07-16 13:29
           +Nmap 4.90RC2 at 2009-07-16 13:33

            scanme.nmap.org (64.13.134.52):
            Host is up.
           -Not shown: 95 filtered ports
           +Not shown: 993 filtered ports
            PORT    STATE  SERVICE VERSION
            22/tcp  open   ssh
            25/tcp  closed smtp
            53/tcp  open   domain
           +70/tcp  closed gopher
            80/tcp  open   http
            113/tcp closed auth
```

Wenn sich zwischen den beiden Läufen etwas geändert hat, dann wird Ndiff das anzeigen.

Nmap-Parser

Seit einiger Zeit bietet Nmap die Ausgabe der Berichte in XML an. Diese Sprache bietet sich besonders zur Weiterverarbeitung durch Programme an. Nmap-Parser (`http://code.google.com/p/nmap-parser/`) ist eine Perl-Bibliothek, mit der Sie das XML-Format in Ihren Programmen parsen können. Dieses Werkzeug wird im Gegensatz zu den bisher vorgestellten Werkzeugen noch aktiv weiterentwickelt.

Nmap-Parser enthält auch Perl-Beispielprogramme in dem Verzeichnis `./tools/`, die diese Bibliothek nutzen:

» `scan.pl`: Ein Perl-Skript, das einen Rechner scannt und die Ausgabe komprimierter als Nmap aufarbeitet.

```
 Scan Host
-------------------------------------------------
[>] 192.168.7.52
      [+] Status: (UP)
      [+]  Hostname(s) :
               test.spenneberg.com
      [+] Operation System(s) :
               Linux Kernel 2.4.0 - 2.5.20
      [+] TCP Ports : (service) [version]
              22       ssh       OpenSSH 3.5p1
              25       smtp
              443      https
      [+] UDP Ports :
              111      rpcbind
```

» `nmap2sqlite.pl`: Dieses Programm nimmt die XML-Ausgabe von Nmap und speichert sie in einer SQLite DB. Dazu braucht dieses Skript eine Tabelle mit dem folgenden Schema:

```
ip             TEXT      PRIMARY KEY NOT NULL,
mac            TEXT,
status         TEXT      DEFAULT 'down',
hostname       TEXT,
open_ports     TEXT      DEFAULT 'none',
filtered_ports TEXT      DEFAULT 'none',
osname         TEXT,
osfamily       TEXT,
osgen          TEXT,
last_scanned   TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
UNIQUE (ip))
```

Das Skript kann sehr einfach für andere Datenbanken angepasst werden.

Auf der Homepage finden Sie in der Rubrik Articles weitere Hinweise und Beispielskripten.

15.7 Nessus und OpenVAS

Nessus (`http://www.nessus.org`) und der Fork OpenVAS (`http://www.openvas.org`) sind Open-Source-Vulnerability-Scanner. Eine komplette Vorstellung von Nessus oder OpenVAS würde den Rahmen dieses Buches sprengen, und es gibt auch bereits andere Bücher, die sich nur mit diesem Thema beschäftigen. Nessus und OpenVAS sind hervorragende Werkzeuge, um einen Audit eines Netzwerks durchzuführen. Dies gilt umso mehr, da sie sehr ausführliche Berichte verfassen, aus denen hervorgeht, welche Sicherheitslücke gefunden wurde, wie diese zu bewerten ist und wie Sie sich dagegen schützen können. Allerdings sind beide Werkzeuge immer nur so gut wie die Person, die sie bedient, und die Plug-Ins, die sie verwenden.

STOP

Nessus- und OpenVAS-Lizenzen

Nessus stand lange Zeit unter einer Open-Source-Lizenz zur Verfügung. Mit der Version 3 hat der Entwickler die Lizenz in eine Closed-Source-Lizenz umgewandelt. Dies war möglich, da der Quelltext der Software bis zu diesem Zeitpunkt im Wesentlichen von ihm geschrieben worden war. Der Urheber kann die Lizenz in neuen Versionen beliebig ändern. Die bereits vorher veröffentlichten Versionen sind weiterhin unter der GPL-Lizenz verfügbar.

Um die weitere Entwicklung im Open-Source-Umfeld zu fördern, wurde mit Unterstützung des Bundesamts für Sicherheit in der Informationstechnik (BSI) das OpenVAS-Projekt gegründet. Dieses stellt einen Fork der letzten Nessus-GPL-Version dar und wird weiterhin unter der GPL-Lizenz entwickelt.

Daher werde ich mich im weiteren Verlauf dieses Kapitels auf OpenVAS beziehen.

OpenVAS verfügt über sehr viele interessante Eigenschaften, die es für den Einsatz als Auditwerkzeug sehr wertvoll machen. Ich werde die wichtigsten im Folgenden aufzählen.

Intelligenter Scan: OpenVAS erkennt Netzwerkdienste auch auf unüblichen Ports.

Modulare Architektur: Der Client kann auf einem beliebigen Windows- oder Linux-System genutzt werden. Lediglich ein Linux-System für den Server ist erforderlich.

Plug-In-Architektur: Jede Sicherheitslücke wird durch ein externes Plug-In realisiert. Neue Tests können so einfach über den Import neuer Plug-Ins hinzugefügt werden. Dies können Sie sehr einfach über den Befehl `nessus-update-plugins` automatisieren. Die Plug-Ins werden in NASL (OpenVAS Attack Scripting Language) geschrieben. Jedes offizielle Plug-In referenziert CVE, CERT, Bugtraq etc.

Berichte: OpenVAS erzeugt sehr ausführliche Berichte. Die Berichte können in HTML, XML oder sogar LaTeX exportiert werden.

SSL-Unterstützung: OpenVAS kann auch von Secure Socket Layer geschützte Dienste testen. Hierzu kann OpenVAS auch ein Clientzertifikat zugewiesen werden.

Scanstufen: OpenVAS kann in unterschiedlichen Angriffsstärken gestartet werden. Während es per Default keinen Schaden auf dem Zielsystem anrichtet, können Sie OpenVAS auch so konfigurieren, dass es zu einem Einbruch oder Denial-of-Service auf dem Zielsystem kommen kann.

Zusätzliche Software: OpenVAS kann weitere externe Software nutzen. So erkennt OpenVAS die Installation von `nmap`, Nikto und Hydra und kann diese Werkzeuge für den Scan nutzen. Nikto (`http://cirt.net/nikto2`) ist ein Scanner für Webserver, während Hydra (`http://thc.org/thc-hydra/`) ein Brute-Force-Angriffswerkzeug für kennwortgeschützte Dienste ist.

Lokale Checks: OpenVAS kann auch lokale Sicherheitsprüfungen auf den Systemen durchführen. Hierfür meldet sich OpenVAS über die Secure Shell (SSH) auf den Systemen an und führt dort die Überprüfungen lokal durch. Im Moment handelt es sich bei den Überprüfungen um die Analyse des Patchlevels für einige spezielle Betriebssysteme. Die aktuelle Liste der unterstützten Systeme finden Sie auf der Homepage.

Wenn Sie OpenVAS zunächst nur einfach kennenlernen möchten, können Sie sich vom BSI die BSI Live-CD BOSS (BSI OSS Security Suite) herunterladen. Diese enthält ein Knoppix mit vorbereitetem und modifiziertem Nessus, dem OpenVAS-Vorgänger. Sie finden weitere Informationen auf der BSI-Homepage.[1]

Wenn Sie OpenVAS testen möchten, sollten Sie nicht das OpenVAS Ihrer Distribution verwenden. Die Entwicklung an OpenVAS schreitet zu schnell voran, als dass eine Distribution hiermit Schritt halten könnte. Laden Sie stattdessen OpenVAS von der Homepage.

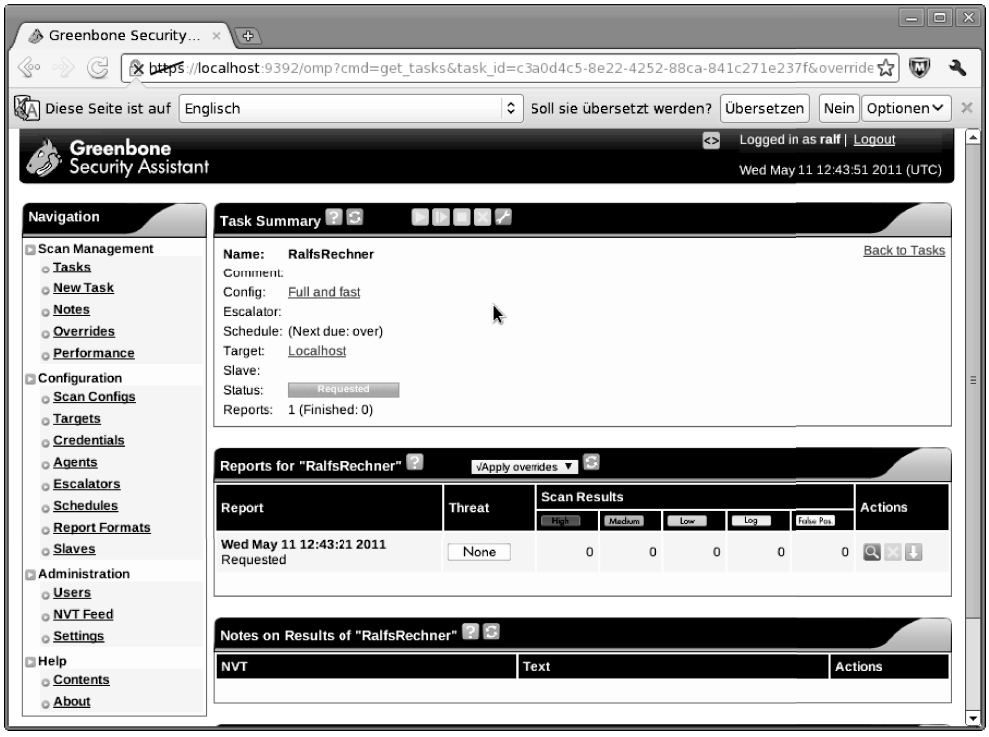


Abbildung 15.10: **Der Greenbone Security Assistent ist eine webbasierte Oberfläche.**

1 `https://www.bsi.bund.de/ContentBSI/Themen/ProdukteTools/BOSS/BSIOSS.html`

Nach der Installation erzeugen Sie auf dem OpenVAS-Server mit dem Befehl `openvas-mkcert` zunächst ein Zertifikat. Die Kommunikation mit dem OpenVAS-Server `openvasd` wird mit SSL geschützt. Sie können für einen ersten Test die Default-Werte für die Zertifikatserzeugung übernehmen. Nun legen Sie mit `openvas-adduser` einen Benutzer an. Bei der Erzeugung des ersten Benutzers können Sie die Default-Werte übernehmen. Wählen Sie als Login einen beliebigen Namen (am einfachsten Ihren Linux-Login). Bei der Authentification wählen Sie `pass`. Wenn Sie nach den Regeln für den Benutzer gefragt werden, geben Sie direkt Strg-D ein. Damit beschränken Sie den Benutzer nicht. Als Kennwort geben Sie ein beliebiges Kennwort an.

Vor dem ersten Start des OpenVAS-Servers sollten Sie die Plug-Ins aktualisieren. Hierzu nutzen Sie den Befehl `openvas-nvt-sync`. Dieser Vorgang dauert bei dem ersten Aufruf einige Minuten. Hiermit können Sie auch später die Plug-Ins im CE aktualisieren.

Jetzt können Sie den OpenVAS-Scanner-Server mit `openvassd -D` oder über ein Startskript starten. Der erste Start dauert ein wenig länger, da zunächst die Plug-In-Datenbank initialisiert wird.

Den OpenVAS-Client starten Sie mit `OpenVAS-Client` (siehe Abbildung 15.12). Alternativ können Sie auch die Weboberfläche Greenbone Security Assistant (GSA, Abbildung 15.10) oder die grafische Applikation Greenbone Security Desktop (GSD, Abbildung 15.11) nutzen. Diese sehen wesentlich schicker aus. Dann ist die Installation jedoch ein wenig umfangreicher. Um dies zu vereinfachen, existiert das Skript `openvas-check-setup`. Dieses Skript können Sie von der Seite `http://www.openvas.org/setup-and-start.html` herunterladen. Dort finden Sie auch weitere Hinweise zur Konfiguration.

Um sich mit dem OpenVAS-Scanner-Server zu verbinden, rufen Sie den entsprechenden Menüpunkt unter DATEI auf oder wählen das entsprechende Icon aus. Hier geben Sie dann den

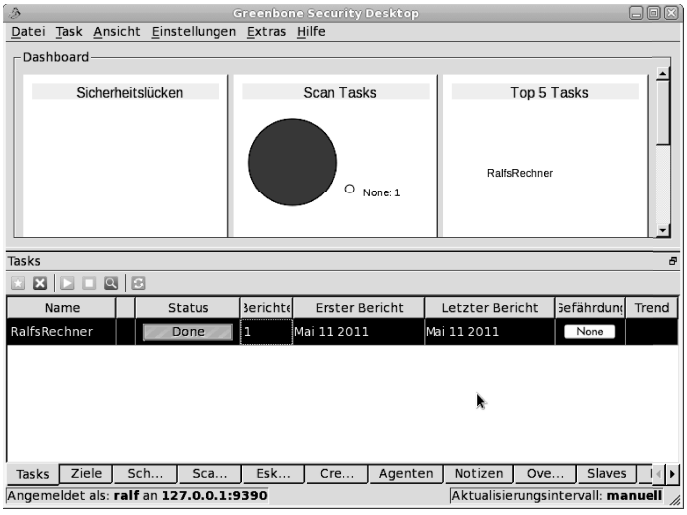


Abbildung 15.11: **Der Greenbone Security Agent ist eine native grafische Oberfläche.**

CE Fehlt hier ein Wort? Bitte prüfen und ggf. ergänzen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

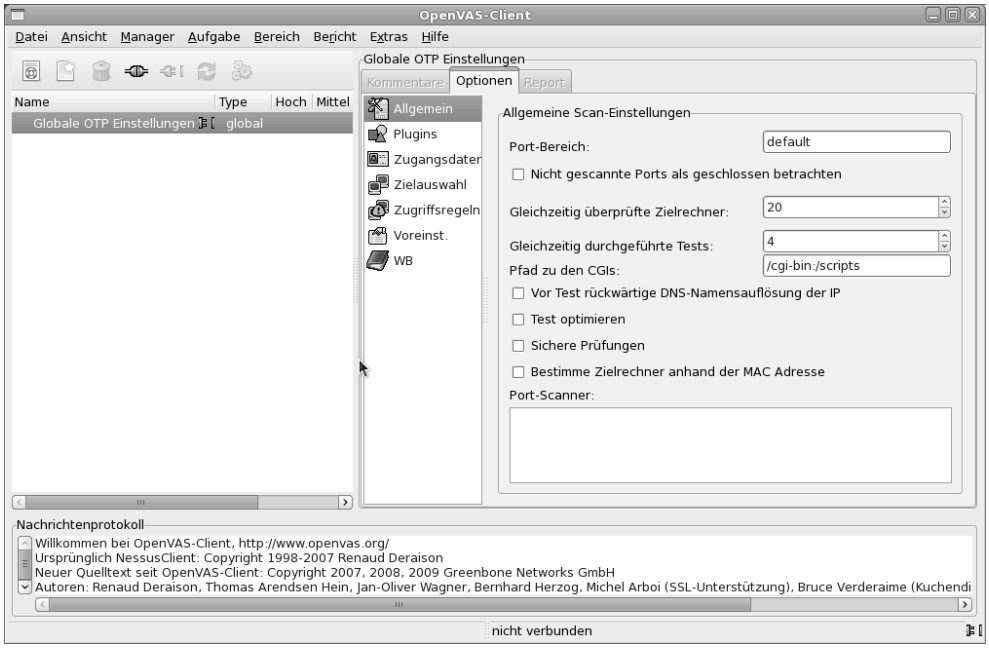


Abbildung 15.12: **Der OpenVAS Client bietet eine übersichtliche grafische Oberfläche.**

Benutzernamen und das Kennwort an, das Sie beim Anlegen des OpenVAS-Benutzers verwendet haben (siehe Abbildung 15.13).

Abbildung 15.13: **Der OpenVAS-Scanner-Server verlangt eine Anmeldung.**

Abbildung 15.14: **Durch Doppelklicken eines Plug-Ins erhalten Sie weitere Informationen über das Plug-In und seine Hintergründe und können es teilweise auch konfigurieren.**

Bei der ersten Anmeldung fragt der Client Sie, ob er dem Zertifikat des OpenVAS-Servers vertrauen soll. Wenn Sie sicher sind, dass kein Mann-in-der-Mitte-Angriff auf die Verbindung möglich ist, oder nachdem Sie das Zertifikat kontrolliert haben, wählen Sie OK und können den Server benutzen.

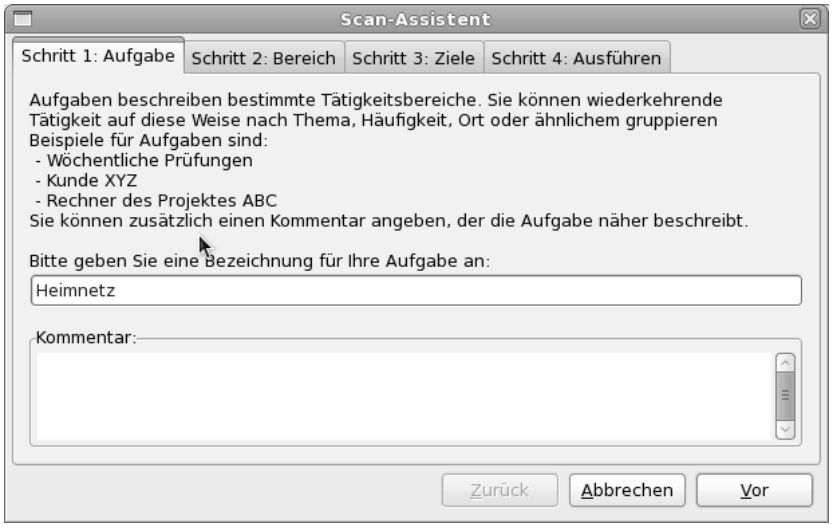


Abbildung 15.15: **Der Scan-Assistent hilft Ihnen bei der Definition des Scans.**

TS² Bitte auf Bild 15.14 im Text verweisen.

Editor's or typesetter's annotations (will be removed before the final TEX run)

Per Default sind direkt nach der Anmeldung sämtliche Plug-Ins aktiviert.TSz

Natürlich können Sie die Plug-Ins einzeln an- und abschalten. Die grafische Oberfläche erleichtert die Wahl der Plug-Ins mithilfe von Filtern, die Sie verwenden können, um nur die Plug-Ins eines bestimmten Autors oder mit einem bestimmten Begriff in der Beschreibung oder im Namen anzuzeigen.

Um nun einen Scan zu definieren und zu starten, wählen Sie unter dem Menüpunkt Datei den Unterpunkt Scan-Assistent (siehe Abbildung 15.15).

Anschließend wählen Sie die Plug-Ins für den Scan ausgewählt.

Des Weiteren können Sie auf der Registerkarte Optionen weitere Daten angeben. OpenVAS wird diese Informationen nutzen, um sich per SSH auf dem Zielsystem anzumelden. Die Registerkarte Scan Options bietet die Möglichkeit, den vorhergehenden Portscan zu optimieren und anstelle des eingebauten Scanners Nmap zu verwenden.

Unter den Voreinstellungen finden Sie Konfigurationsmöglichkeiten für jeden Aspekt von OpenVAS inklusive der IDS-Evasion, Web-Traversal-Angriffen, Web-Mirroring etc. Für den ersten Scan sollten Sie diese Parameter nicht modifizieren.TS

Sehr interessant ist auch der Punkt WB. Hier verbirgt sich die Möglichkeit, das Ergebnis eines Scans in einer Datenbank (Wissensbasis, daher WB)TSazu speichern. Spätere Scans können dann in Abhängigkeit von der Datenbank durchgeführt werden!

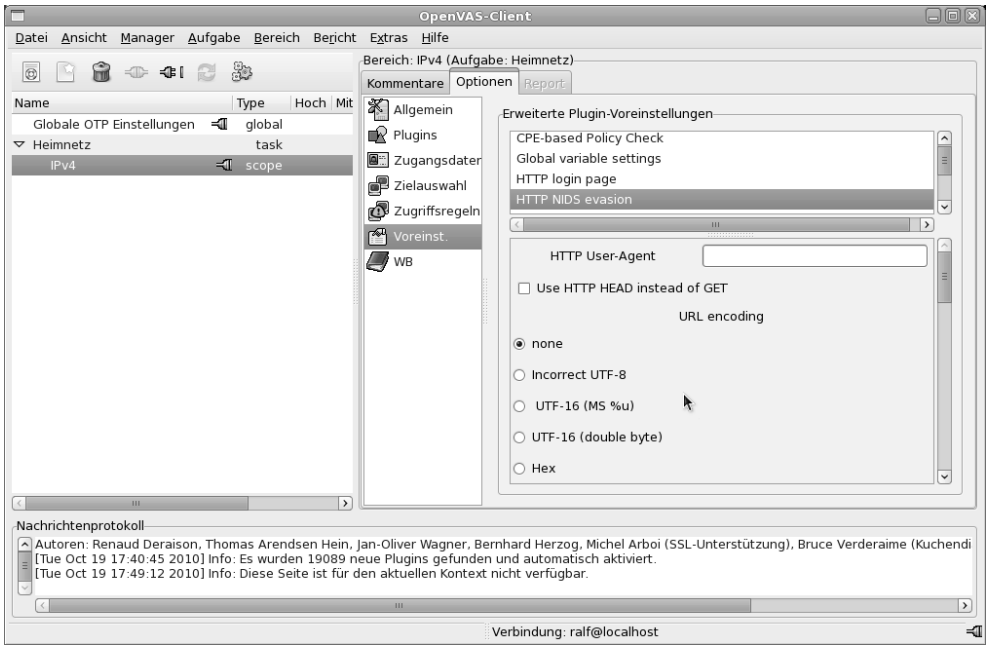


Abbildung 15.16: **OpenVAS bietet Evasionstechniken zur Verschleierung des Angriffs.**

TS Bitte auf Bild 15.16 im Text verweisen.

TSa Bitte diese Erklärung bestätigen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

Haben Sie alles konfiguriert, starten Sie den Scan (siehe Abbildung 15.17). Dieser Scan dauert nun je nach der Netzgeschwindigkeit, den ausgewählten Plug-Ins und möglicher Firewalls wenige Minuten – oder sogar Stunden.

Sobald OpenVAS mit dem Scan fertig ist, zeigt es in einem grafischen Browser den Bericht an. In dem Bericht aus Abbildung 15.18 wurde nicht viel Erwähnenswertes gefunden. Dennoch sollten Sie in der Abbildung erkennen, dass OpenVAS die gefundenen Probleme beschreibt, Lösungen vorschlägt, Risikoabschätzungen abgibt und Referenzen aufzählt. Sie können den Bericht nun als NBE, XML, HTML mit Grafiken und LaTeX abspeichern. Wenn Sie den Bericht in dem NBE-Format abspeichern, können Sie ihn später wieder in OpenVAS laden und auch in anderen Formaten sichern. Daher sollten Sie immer auch dieses Format wählen!

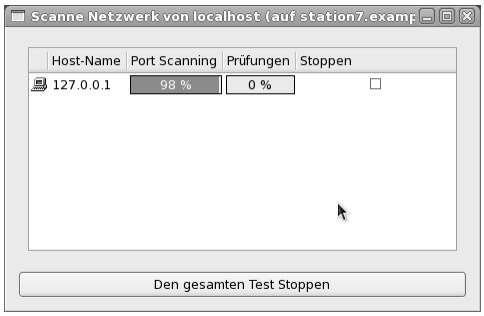


Abbildung 15.17: **Während des Scans zeigt OpenVAS den Fortschritt an.**

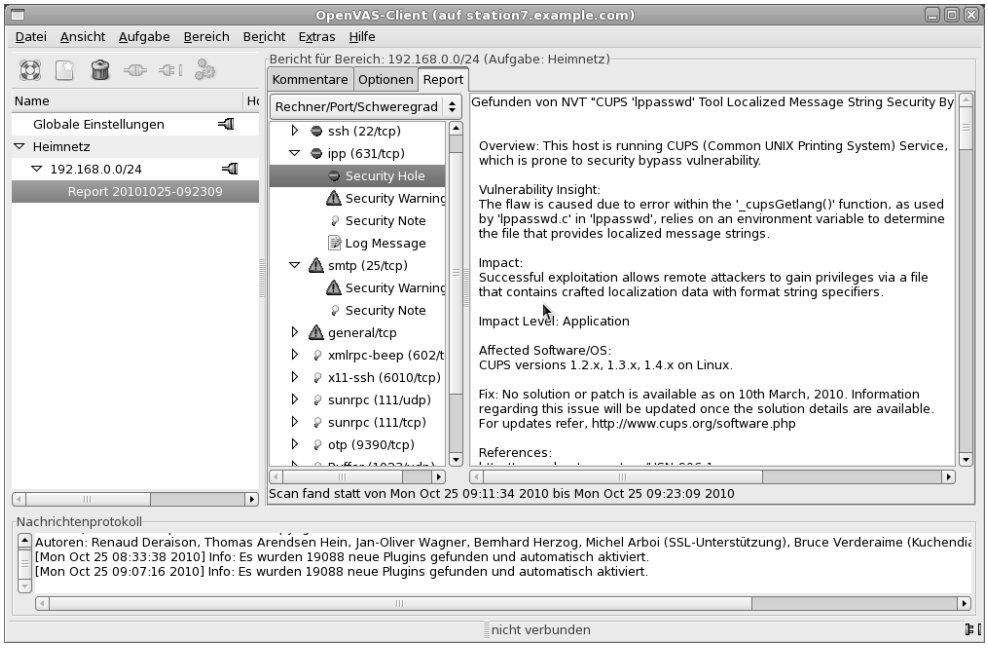


Abbildung 15.18: **OpenVAS erzeugt einen ausführlichen Bericht mit den gefundenen Problemen.**

Ich hoffe, Sie haben mit dieser kurzen Einführung in OpenVAS einen ersten Einblick erhalten. Wenn Sie weitere Informationen über OpenVAS wünschen, finden Sie auf der OpenVAS- und Nessus-Homepage einige Artikel.

Teil V

Fortgeschrittene Konfiguration

16. Die Iptables-Standardtests

Mit dem IPtables-Befehl können Sie verschiedenste Eigenschaften eines Paketes prüfen. Dieses Kapitel führt alle diese verschiedenen Prüfungen auf, die in dem Linux-Kernel 2.6.35 und Iptables 1.4.9 enthalten sind.

Im Folgenden werden die verschiedenen Funktionen zur Prüfung der Pakete alphabetisch aufgeführt.

16.1 Eingebaute Funktionen

Netfilter/Iptables verfügt über eingebaute Tests und Erweiterungen. Während die Erweiterungen über Kernelmodule realisiert werden und immer mit der Option `-m <extension>` in jeder Regel aktiviert werden müssen, stehen die eingebauten Tests immer zur Verfügung. Die folgenden eingebauten Tests können Sie immer verwenden.

16.1.1 -p, --protocol

Dieser Test prüft das in dem Paket verwendete IP-Protokoll. Hier können Sie den Namen des Protokolls (`tcp`, `udp` etc.) oder die Protokollnummer verwenden. Die Definition der Protokollnamen erfolgt in der Datei `/etc/protocols`. Wenn Sie den Test negieren möchten, können Sie das Ausrufezeichen dem Protokoll voranstellen: `--protocol ! tcp`.

16.1.2 -s, --source

Dieser Test prüft die Quell-IP-Adresse des Pakets. Dies kann eine IP-Adresse, ein Netzwerk oder ein DNS-Name sein. Bei der Angabe des Netzwerks können Sie die Netzmaske ausschreiben (`255.255.255.0`) oder die CIDR-Notation (`/24`) verwenden. Wenn Sie einen DNS-Namen verwenden, wird Iptables den DNS-Namen in dem Moment, in dem Sie die Regel hinzufügen, auflösen und anstelle des DNS-Namens die aufgelöste IP-Adresse nutzen. Dynamische DNS-Namen, deren IP-Adresse sich ändert, werden nicht unterstützt. Falls Sie `ip6tables` verwenden, können Sie entsprechend IPv6-Adressen verwenden.

16.1.3 -d, --destination

Hiermit prüfen Sie die Ziel-IP-Adresse. Im Weiteren gelten dieselben Beschränkungen wie bei der Quell-IP-Adresse.

16.1.4 -i, --in-interface

Hiermit können Sie prüfen, über welche Netzwerkkarte das Paket Ihren Rechner erreicht hat. Diesen Test können Sie nur in den `PREROUTING`-, `INPUT`- und `FORWARD`-Ketten benutzen. Die Verwendung in den `OUTPUT`- oder `POSTROUTING`-Ketten ist nicht erlaubt.

Die Netzwerkkarte muss in dem Moment, in dem Sie die Regel hinzufügen, noch nicht existieren. Sie können also die Regeln erzeugen, bevor die Netzwerkkarten initialisiert wurden.

Wenn Sie mehrere Netzwerkkarten in einer Regel testen möchten, können Sie das + als Wildcard nutzen. Ein `eth+` trifft dann auf alle Ethernet-Netzwerkkarten zu.

16.1.5 -o, --out-interface

Hiermit können Sie prüfen, über welche Netzwerkkarte ein Paket den Rechner verlässt. Diesen Test dürfen Sie nur in den `FORWARD`-, `OUTPUT`- und `POSTROUTING`-Ketten verwenden. Ansonsten gelten die gleichen Einschränkungen wie oben.

16.1.6 -f, --fragment

Hiermit prüfen Sie, ob es sich bei einem fragmentierten Paket um das zweite oder ein weiteres Fragment handelt. Da Iptables ab dem zweiten Fragment keinen Zugriff mehr auf die Informationen des Transport-Protokolls hat, können Sie hiermit diese Fragmente herausfiltern und bei Bedarf zulassen.

Da sich fragmentierte Pakete nur schlecht filtern lassen, sollten grundsätzlich alle Pakete vor der Filterung defragmentiert werden. Während Sie auf alten Linux-Kerneln dies speziell anschalten müssen, ist diese Funktion auf dem Linux-Kernel 2.4. und 2.6 automatisch aktiv, sobald Sie Connection Tracking nutzen. Sobald das Kernelmodul `xx_conntrack` geladen wurde, werden alle Pakete defragmentiert!

16.2 TCP-Tests

Wenn es sich bei dem Paket um ein TCP-Paket handelt und Sie dies mit der Option `-p tcp` in der Regel prüfen, können Sie weitere Prüfungen des TCP-Headers vornehmen. Dabei können Sie die Ports, die TCP-Flags und die TCP-Optionen prüfen. Im Grunde wird immer, wenn Sie mit `-p tcp` auf ein TCP-Paket prüfen, die Iptables-Erweiterung `tcp` geladen. Grundsätzlich könnten Sie dies auch zusätzlich (was aber überflüssig ist) in der IPtables-Kommandozeile tun: `-m tcp -p tcp`

16.2.1 --sourceport

Mit der Option `--sourceport` (oder auch `--sport`) können Sie den Quellport prüfen. Hier können Sie einen Port oder einen Portbereich (`<port>:<port>`) angeben. Mehrere verschiedene Ports können Sie nicht angeben. Hierfür benötigen Sie die `multiport`-Erweiterung.

16.2.2 --destinationport

Mit der Option `--destinationport` (oder auch `--dport`) können Sie den Zielport prüfen. Hier können Sie einen Port oder einen Portbereich (`<port>:<port>`) angeben. Mehrere verschiedene Ports können Sie nicht angeben. Hierfür benötigen Sie die `multiport`-Erweiterung.

16.2.3 --tcp-flags

Mit dieser Option können Sie jedes einzelne der TCP-Flags prüfen. Ingesamt gibt es sechs verschiedene: `SYN`, `ACK`, `FIN`, `RST`, `URG` und `PSH`. Dieser Test unterstützt auch noch `NONE` und `ALL`. Um die Flags zu testen, müssen Sie zunächst eine Maske mit den Flags definieren, die Sie prüfen möchten, und geben dann, durch ein Leerzeichen getrennt, die Flags an, die gesetzt sein müssen. Wenn Sie zum Beispiel prüfen möchten, ob das SYN-Flag gesetzt ist, aber das RST- und ACK-Flag nicht gesetzt sind, verwenden Sie: `--tcp-flags SYN,ACK,RST SYN`. Dieser Test betrachtet alle drei Flags (`SYN`, `ACK` und `RST`). Aber nur `SYN` darf gesetzt sein. Der Zustand der anderen Flags ist unerheblich.

Das Netzwerk-Scan-Werkzeug `hping2` (`http://www.hping.org`) verwendet per Default TCP-Pakete, in denen kein einziges Flag gesetzt ist. Wenn Sie diese Pakete erkennen möchten, können Sie den folgenden Test nutzen:

```
$IPTABLES -A INPUT -p tcp --tcp-flags ALL NONE -j LOG --log-prefix " ↵
        Hping2 Scan:"
```

16.2.4 --syn

Dieser Test (`--syn`) ist eine Kurzform für `--tcp-flags SYN,ACK,RST,FIN SYN` und prüft, ob ein Paket das erste SYN-Paket in einer Verbindung ist. Wenn Sie diesen Test negieren möchten, setzen Sie ein Ausrufezeichen vor ihn.

16.2.5 --tcp-option

Das TCP-Protokoll unterstützt verschiedene Optionen, die das Verhalten des TCP-Protokolls beeinflussen. Eine dieser Optionen ist zum Beispiel die Maximum Segment Size (MSS, siehe Abschnitt 17.25). Mit diesem Test können Sie prüfen, ob eine bestimmte Option gesetzt ist.

16.3 UDP-Tests

Das UDP-Protokoll verfügt nicht über so viele Informationen wie das TCP-Protokoll. Daher können Sie hier nur die Ports testen. Auch hier wird automatisch bei der Prüfung des UDP-Protokolls die entsprechende Erweiterung geladen: `-m udp -p udp`.

16.3.1 --sourceport

Mit der Option `--sourceport` (oder auch `--sport`) können Sie den Quellport prüfen. Hier können Sie einen Port oder einen Portbereich (`<port>:<port>`) angeben. Mehrere verschiedene Ports können Sie nicht angeben. Hierfür benötigen Sie die `multiport`-Erweiterung.

16.3.2 --destinationport

Mit der Option `--destinationport` (oder auch `--dport`) können Sie den Zielport prüfen. Hier können Sie einen Port oder einen Portbereich (`<port>:<port>`) angeben. Mehrere verschiedene Ports können Sie nicht angeben. Hierfür benötigen Sie die `multiport`-Erweiterung.

16.4 ICMP-Tests

Das ICMP-Protokoll besitzt keine Ports. Bei ICMP werden die verschiedenen Nachrichten durch Typ und Code unterschieden. Damit auch Sie prüfen können, um was für eine ICMP-Nachricht es sich handelt, unterstützt Iptables für ICMP-Pakete den Test `--icmp-type`. Der Iptables-Befehl zeigt die verschiedenen ICMP-Typen an, die Sie testen können:

```
$ iptables -p icmp -h
...
ICMP v1.3.0 options: --icmp-type [!]              typename match icmp type
                                                  (or numeric type or type ↵
                                                          /code)

Valid ICMP Types:
any
echo-reply (pong)
destination-unreachable
   network-unreachable
   host-unreachable
   protocol-unreachable
   port-unreachable
   fragmentation-needed
   source-route-failed
   network-unknown
   host-unknown
   network-prohibited
   host-prohibited
   TOS-network-unreachable
   TOS-host-unreachable
   communication-prohibited
   host-precedence-violation
   precedence-cutoff
source-quench
redirect
```

```
   network-redirect
   host-redirect
   TOS-network-redirect
   TOS-host-redirect
echo-request (ping)
router-advertisement
router-solicitation
time-exceeded (ttl-exceeded)
   ttl-zero-during-transit
   ttl-zero-during-reassembly
parameter-problem
   ip-header-bad
   required-option-missing
timestamp-request
timestamp-reply
address-mask-request
address-mask-reply
```

Die Filterung des ICMP-Protokolls wird ausführlich in Kapitel 35, „ICMP“, besprochen. Wenn Sie den Befehl `ip6tables` verwenden, können Sie auch die ICMP-Nachrichten des IPv6-Protokolls prüfen.

16.5 Weitere Erweiterungen

16.5.1 addrtype

Der Routing-Code des Linux-Kernels kategorisiert jede IP-Adresse im Netzwerkstack. Dieser Test kann diese Kategorien testen und entsprechend Pakete verwerfen oder protokollieren. Die folgenden Kategorien werden von Linux unterstützt: `UNSPEC`, `UNICAST`, `LOCAL`, `BROADCAST`, `ANYCAST`, `MULTICAST`, `BLACKHOLE`, `UNREACHABLE`, `PROHIBIT`, `THROW`, `NAT` und `XRESOLVE`. Leider existiert kaum Dokumentation, in der die Kategorien erläutert werden.

Sie können sowohl die Quell-IP-Adresse als auch die Ziel-IP-Adresse testen:

```
-m addrtype --src-type <type>
-m addrtype --dst-type <type>
```

Ich persönlich kenne keine sinnvolle Verwendung für `--src-type` und `--dst-type` und führe daher diese Option nur der Vollständigkeit halber auf. Falls Sie eine sinnvolle Anwendung haben, würde ich mich über einen kurzen Hinweis freuen.

16.5.2 ah

Mit diesem Test können Sie die Security-Parameter-Indizes (SPI) (`--ahspi`) des IPsec-AH-Protokolls testen. Da der Security-Parameter-Index üblicherweise dynamisch von dem IKE-

Daemon ausgehandelt wird, ist eine statische Prüfung nicht besonders sinnvoll, außer Sie verwenden manuell definierte IPsec-AH-Verbindungen (z. B. mit `setkey`).

```
-m ah --ahspi [!] <spi>[:<spi>]
```

16.6 cluster

Hiermit können Sie einen lastverteilenden Cluster ohne einen Lastverteiler (Load-Balancer) aufbauen. Dabei ist dieser Cluster speziell auch für Router und Firewalls geeignet. Natürlich können Sie aber auch einen Cluster für ein Endgerät (z. B. einen Webserver) realisieren. Sie müssen lediglich sicherstellen, dass alle Knoten in Ihrem Cluster alle Pakete sehen können. Hierzu müssen alle Clusterknoten dieselbe Multicast-MAC-Adresse benutzen. ARP-Anfragen auf die IP-Adresse des Clusters werden dann mit der Multicast-MAC-Adresse beantwortet. Die Pakete werden dann auf der Schicht 2 an die Multicast-MAC-Adresse gesendet und von dem Switch[1] an alle Knoten des Clusters ausgeliefert. Diese prüfen dann mithilfe eines Hashes, ob Sie das Paket verarbeiten müssen oder nicht.

Diese Erweiterung kennt die folgenden Optionen:

» `--cluster-total-nodes`: Hiermit geben Sie an, wie viele Knoten Ihr Cluster aktuell aufweist. Diese Zahl sollte aus Gründen der Ausfallsicherheit von einem externen Cluster-Manager ermittelt werden.

» `--cluster-local-node`: Hiermit definieren Sie die lokale Nummer des Knotens. Jeder Knoten muss eine eindeutige Nummer in dem Cluster besitzen.

» `--cluster-local-nodemask`: Alternativ können Sie auch eine Maske für die Erkennung des lokalen Knotens verwenden.

» `--cluster-hash-seed`: Hiermit geben Sie den Startwert für den Hash an.

Am einfachsten lässt sich die Funktion an einem Beispiel aus der IPtables-Manpage erklären. Zunächst müssen Sie eine Multicast-MAC-Adresse wählen. Diese Adressen beginnen mit 01:XX:XX:XX:XX:XX. Die von Ihnen gewählte Adresse muss von dem System für die Kommunikation mit der Außenwelt verwendet werden. Das Beispiel geht von einem Routing-Cluster aus. Hier verfügt jedes System über zwei Netzwerkkarten: `eth1` und `eth2`. Diese erhalten jeweils eine Multicast-MAC-Adresse:

```
# Setze die Multicast-MAC-Adressen
ip maddr add 01:00:5e:00:01:01 dev eth1
ip maddr add 01:00:5e:00:01:02 dev eth2
# Modifiziere die eingehenden und ausgehenden Pakete auf eth1
arptables -A OUTPUT -o eth1 --h-length 6 -j mangle --mangle-mac-s 01:00:5 ↵
        e:00:01:01
arptables -A INPUT -i eth1 --h-length 6 --destination-mac 01:00:5e ↵
        :00:01:01 -j mangle --mangle-mac-d 00:zz:yy:xx:5a:27
```

1 Der Switch muss diese Funktion unterstützen, und er muss möglichweise auch konfiguriert werden!

```
# Modifiziere die eingehenden und ausgehenden Pakete auf eth2
arptables -A OUTPUT -o eth2 --h-length 6 -j mangle --mangle-mac-s 01:00:5 ↵
        e:00:01:02
arptables -A INPUT -i eth2 --h-length 6 --destination-mac 01:00:5e ↵
        :00:01:02 -j mangle --mangle-mac-d 00:zz:yy:xx:5a:27
```

Nun werden die ankommenden Pakete markiert und alle nicht markierten Pakete auf der entsprechenden Netzwerkkarte verworfen. Durch die Markierung erkennt das System, welche Pakete es selbst verarbeiten muss.

```
# Auf einem Cluster mit zwei Knoten ist dies der erste Knoten
iptables -A PREROUTING -t mangle -i eth1 -m cluster --cluster-total-nodes ↵
         2 --cluster-local-node 1 --cluster-hash-seed 0xdeadbeef -j MARK ↵
         --set-mark 0xffff
iptables -A PREROUTING -t mangle -i eth2 -m cluster --cluster-total-nodes ↵
         2 --cluster-local-node 1 --cluster-hash-seed 0xdeadbeef -j MARK ↵
         --set-mark 0xffff
# Alle Pakete, die nicht markiert wurden, sind auf diesem Knoten nicht ↵
        relevant
iptables -A PREROUTING -t mangle -i eth1 -m mark ! --mark 0xffff -j DROP
iptables -A PREROUTING -t mangle -i eth2 -m mark ! --mark 0xffff -j DROP
```

Diese Erkennung erfolgt in Abhängigkeit von der Source-IP-Adresse. Pakete derselben Source-Adresse werden daher auch von demselben Knoten im Cluster verarbeitet.

Dabei wird die folgende Formel zugrunde gelegt:

$$(jhash(sourceIP) modtotal - nodes) \& node - mask \qquad (16.1)$$

Als Hash wird der *Jenkins-Hash* verwendet. Hierbei handelt es sich um eine nichtkryptografische Hash-Funktion.[2] Sie können für diese Hash-Funktion optional einen Seed-Wert vergeben. Dieser muss dann jedoch auf allen Knoten des Clusters identisch sein. Mithilfe des Seed-Wertes, der Anzahl der verfügbaren Knoten und der eigenen Knotennummer ermittelt dann die Erweiterung auf der Basis der Source-IP-Adresse, ob das Paket von diesem Knoten verarbeitet werden muss.

16.7 comment

Dies ist kein wirklicher Test, da Sie hiermit keine Eigenschaft des Pakets prüfen können. Vielmehr können Sie einen Kommentar (`--comment`) zu einer Regel hinzufügen. Damit können Sie nun die Regeln selbst dokumentieren, sodass bei einer späteren Auflistung der Regeln mit `Iptables -vnL` die Kommentare angezeigt werden. Der Kommentar darf maximal 256 Zeichen lang sein.

```
-m comment --comment "Kommentar"
```

2 http://en.wikipedia.org/wiki/Jenkins_hash_function

16.8 connbytes

Mit diesem Test können Sie prüfen, wie viele Pakete oder Bytes in einer Verbindung bereits übertragen wurden. So können Sie langlebige Downloads mit einer geringeren Priorität versehen, sodass sie den restlichen Verkehr nicht negativ beeinflussen. Sie können mit diesem Test die übertragenen Pakete oder Bytes in einer oder beiden Richtungen analysieren. Außerdem können Sie die durchschnittliche Paketgröße testen.

Dieser Test verfügt über drei zusätzliche Optionen:

» `[!] --connbytes <from>:[<to>]`: Hiermit prüfen Sie, ob die übertragenen Daten sich zwischen `from` und `to` befinden. Wenn Sie `to` nicht angeben, ist die obere Grenze nicht gesetzt.

» `--connbytes-dir [original|reply|both]`: Hiermit geben Sie die zu überwachenden Richtungen an.

» `--connbytes-mode [packets|bytes|avgpkt]`: Hiermit wählen Sie den Modus aus. Entweder zählt `Connbytes` die übertragenen Pakete oder Bytes, oder `Connbytes` ermittelt die durchschnittliche Paketgröße.

```
-m connbytes --connbytes 100000: --connbytes-dir both --connbytes-mode ↵
        bytes
```

Da die Zähler 64-Bit-Zähler sind, ist ein Überlauf der Zähler sehr unwahrscheinlich. Sie können sehr langlebige Verbindungen hiermit überwachen.

16.8.1 connlimit

Hiermit können Sie die Anzahl der gleichzeitigen parallelen Verbindungen je Client beschränken. Hierzu bietet diese Erweiterung zwei Optionen:

» `--connlimit-above`: Hiermit geben Sie die maximale Anzahl der gleichzeitig erlaubten Verbindungen an.

» `--connlimit-mask`: Hiermit können Sie den Kreis der zu überwachenden Rechner von einem Rechner auf ein Netz erweitern. Hierzu definieren Sie die Netzmaske der zu überwachenden Rechner.

Am einfachsten erklärt sich die Option wieder an einem Beispiel. Der folgende Aufruf lehnt alle SSH-Verbindungsversuche ab, sobald zwei bereits zwei Verbindungen existieren:

```
iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above ↵
        2 -j REJECT
```

Um die Anzahl der HTTP-Verbindungen pro Class-C-Subnetz auf 32 zu beschränken, nutzen Sie:

```
iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above ↵
        32 --connlimit-mask 24 -j REJECT
```

16.8.2 connmark

Verbindungen, die Sie in der NAT-Tabelle mit dem Target `CONNMARK` markiert haben, können Sie hiermit testen. Dazu können Sie entweder exakt die Markierung (`--mark`) angeben oder auch zusätzlich noch eine Maske definieren. Diese Maske wird vor dem Test mit der Markierung der Verbindung Und-verknüpft.

```
-m connmark --mark <markierung>[/<maske>]
```

16.8.3 conntrack

Dieses Modul gibt Ihnen erweiterten Zugriff auf die internen Werte der Connection-Tracking-Tabelle. Hiermit können Sie diese Werte in Regeln ausnutzen. Sie können auf diese Weise prüfen, welchen Zustand (`INVALID`, `RELATED`, `NEW`, `SNAT`, `DNAT`, `UNTRACKED`), welche originale und welche genattete Adresse die dazugehörige Verbindung in der Zustandstabelle aufweist.

Da mir keine sinnvolle Anwendung bekannt ist, verweise ich den interessierten Leser auf die `iptables(8)`-Manpage.

16.9 dccp

Das Datagram Congestion Control Protocol (DCCP) ist ein nachrichtenorientiertes Transport-Protokoll. Es implementiert wie TCP einen Datenfluss, aber bietet keine Übertragungssicherheit. DCCP befindet sich aktuell in der Entwicklung und wird meines Wissens noch nicht produktiv eingesetzt.

Mit dieser Erweiterung können Sie das DCCP-Protokoll testen. Der Test unterstützt die folgenden Optionen:

- `--source-port`: Prüft den Quellport.
- `--destination-port`: Prüft den Zielport.
- `--dccp-types`: DCCP unterstützt die folgenden Pakettypen: `REQUEST`, `RESPONSE`, `DATA`, `ACK`, `DATAACK`, `CLOSEREQ`, `CLOSE`, `RESET`, `SYNC`, `SYNCACK` und `INVALID`.
- `--dccp-option`: Hiermit können Sie prüfen, ob eine numerische DCCP-Option gesetzt ist.

16.9.1 dscp

Das 6 Bit lange DiffServ-Code-Point-(DSCP-)Feld befindet sich in dem TOS-Feld des IP-Headers. DSCP hat TOS in den aktuellen IP-Protokoll-Standards abgelöst.

Sie können entweder den Wert numerisch (`--dscp`) oder durch Angabe der DSCP-Klasse (`--dscp-class`) testen:

```
-m dscp --dscp <nummer>
-m dscp --dscp-class <klasse>
```

16.9.2 ecn

Mit diesem Test können Sie prüfen, ob die Explicit-Congestion-Notification-(ECN-)Bits in dem Paket gesetzt sind. Mit den folgenden Optionen können Sie die einzelnen Bits testen:

» `--ecn-tcp-cwr`: Ist das Congestion-Window-Received-Bit gesetzt?

» `--ecn-tcp-ece`: Ist das ECN-Echo-Bit gesetzt?

» `--ip-ect <nummer>`: Ist ein ECN-Capabable-Transport-Bit gesetzt?

Weitere Informationen über ECN und die verwendeten Bits im IP- und TCP-Header finden Sie auf `http://www.icir.org/floyd/ecn.html`. Eine sinnvolle Anwendung ist mir unbekannt, da Sie die ECN-Bits entfernen können, ohne vorher ihr Vorhandensein zu prüfen.

16.9.3 esp

Hiermit können Sie ähnlich der `ah`-Erweiterung die Security-Parameter-Indizes (`--espspi`) des IPsec-ESP-Protokolls testen. Da auch diese dynamisch ausgehandelt werden, ist die Anwendung nicht sinnvoll.

```
-m esp --espspi [!] <spi>[:<spi>]
```

16.9.4 hashlimit

Dieser sehr interessante Test ähnelt der `limit`-Erweiterung. Wenn Sie die `limit`-Erweiterung noch nicht kennen, sollten Sie erst dort nachlesen (siehe Abschnitt 16.9.8). Kehren Sie anschließend hierher zurück.

Diese Erweiterung kann in zwei unterschiedlichen Varianten auftreten. Die ältere Version wird zunächst beschrieben. Um zu prüfen, welche Version Sie verwenden können, rufen Sie `iptables -m hashlimit -h` auf. Werden hier die Optionen `--hashlmit-upto` und `--hashlmit-above` erwähnt, so verwenden Sie die neuere Version und springen zum Abschnitt „Hashlimit (Neue Version)“. TS[b]

Hashlimit (Alte Version)

Die `hashlimit`-Erweiterung ermöglicht Ihnen im Gegensatz zum allgemeinen Limit-Test die Definition von Grenzwerten in Abhängigkeit von der Ziel/Quell-IP-Adresse und des -Ports.

Die Erweiterung besitzt eigene Zähler für jede IP-Adresse und/oder jeden Port und nicht nur einen Zähler für die gesamte Regel.

Die Erweiterung unterstützt die folgenden Optionen:

» `--hashlimit <rate>`: siehe `limit`.

» `--hashlimit-burst <burst>`: siehe `limit`.

» `--hashlimit-mode destip|srcip|dstport|srcport`: Hashlimit-Modus.

» `--hashlimit-name <name>`: Dies erzeugt einen Eintrag in `/proc/net/ipt_hashlimit/<name>` für diese Regel.

TS[b] Bitte prüfen: Verweis auf unnummerierte Abschnitte nicht möglich.

Editor's or typesetter's annotations (will be removed before the final TeX run)

» `--hashlimit-htable-size <num>`: Größe der Hash-Tabelle in Buckets.

» `--hashlimit-htable-max <max>`: Maximale Anzahl der Einträge in der Hash-Tabelle.

» `--hashlimit-htable-gcinterval <intvl>`: Garbage Collector Interval (ms).

» `--hashlimit-htable-expire <ttl>`: Lebensdauer (in Millisekunden) der Einträge in der Hash-Tabelle.

Mit diesem Test lassen sich zum Beispiel die SSH-Brute-Force-Angriffe der letzten Jahre hervorragend abwehren:

```
$IPTABLES -A INPUT -m tcp -p tcp --dport 22  -m hashlimit --hashlimit 1/ ↵
        min --hashlimit-mode srcip --hashlimit-name ssh -m state --state ↵
         NEW -j ACCEPT
```

Diese Regel akzeptiert nur eine SSH-Verbindungsanfrage pro Minute und pro Quell-IP-Adresse. Allerdings kann natürlich ein Angreifer auch einen DoS ausführen, falls er die IP-Adresse erraten kann, von der aus Sie sich mit dem System verbinden möchten. Er spooft dann einfach Ihre IP-Adresse und führt häufige Verbindungsaufbauten durch!

Weil das exakte Verhalten von der Hashlimit-Erweiterung häufig zu Missverständnissen führt, gibt es unter `http://wodny.org/special/hashlimit.html` einen Simulator, der bei gegebenen Werten zeigt, wie sich das Hashlimit verhält.

Die Manpage des Iptables-Kommandos hat hier in einigen Versionen einen Fehler und behauptet, dieser Test könnte nur die Ziel-IP-Adresse mit einer Beschränkung belegen. Testen Sie Ihren Kernel mit:

```
$ iptables -m hashlimit -h hashlimit
v1.3.0 options:
--hashlimit <avg>               max average match rate
                                [Packets per second unless followed by
                                 /sec /minute /hour /day postfixes]
--hashlimit-mode <mode>         mode is a comma-separated list of dstip,
                                srcip,dstport,srcport
--hashlimit-name <name>         name for /proc/net/ipt_hashlimit/
[--hashlimit-burst <num>]       number to match in a burst, default 5
[--hashlimit-htable-size <num>] number of hashtable buckets
[--hashlimit-htable-max <num>]  number of hashtable entries
[--hashlimit-htable-gcinterval] interval between garbage collection runs
[--hashlimit-htable-expire]     after which time are idle entries
                                expired?
```

Hashlimit (Neue Version)

Diese Version stellt eine aktuellere Variante des Hashlimit-Tests dar. Hiermit können Sie Regeln definieren wie: Gibt es mehr als 100 Pakete je Sekunde für die einzelnen Dienste eines Rechners?

Die Optionen lauten:

» `--hashlimit-upto`: Hiermit prüfen Sie, ob die Rate kleiner oder gleich dem angegebenen Wert ist. Dabei können Sie als Zeiteinheit nutzen:

- second
- minute
- hour
- day

Ein Beispiel: `--hashlimit-upto 500/second`. Der Default ist `3/hour`.

» `--hashlimit-above`: Hiermit prüfen Sie, ob die Rate den angegebenen Wert überschritten hat.

» `--hashlimit-burst`: Dies ist ein initialer Puffer, der zunächst erreicht werden muss, bevor die Rate gemessen wird. Der Default beträgt 5.

» `--hashlimit-mode`: Hiermit geben Sie in einer kommaseparierten Liste an, was Sie messen möchten:

- `srcip`
- `dstip`
- `srcport`
- `dstport`

» `--hashlimit-srcmask`: Hiermit können Sie die Source-IP-Adressen in einzelne Netze für die Zählung gruppieren.

» `--hashlimit-dstmask`: Dies erlaubt analog die Gruppierung der Ziel-IP-Adressen.

» `--hashlimit-name`: Unter diesem Namen werden die Daten unterhalb von `proc/net/ipt_hashlimit/` abgelegt.

» `--hashlimit-htable-size`: Dieser Wert bestimmt die Anzahl der Buckets in der Hash-Table.

» `--hashlimit-htable-max`: Hiermit bestimmen Sie die maximale Anzahl der Einträge.

» `--hashlimit-htable-expire`: Hiermit geben Sie an, wie lange (in Millisekunden) die Einträge in der Tabelle verbleiben.

» `--hashlimit-htable-gcinterval`: Hiermit geben Sie an, wie häufig (in Millisekunden) die Garbage Collection die Hash-Table abarbeiten soll.

16.9.5 helper

Dieser Test prüft, ob ein Paket durch ein bestimmtes Conntrack-Helfermodul erkannt wurde. Hierzu geben Sie lediglich den Namen des Helfermoduls an. Um alle FTP-Pakete (Control- und Data-Connection) zu erkennen, können Sie folgenden Ausdruck verwenden: `-m helper --helper ftp`. Wenn Sie von den Default-Ports abgewichen sind, können Sie den Port mit angeben: `-m helper --helper ftp-2121`. Andere Helfermodule arbeiten genauso. Damit können Sie zum Beispiel RELATED-Pakete nur akzeptieren, wenn es sich gleichzeitig um

Pakete einer FTP-Verbindung handelt. ICMP-Fehlermeldungen oder IRC-Datenverbindungen werden von der folgenden Regel nicht akzeptiert:

```
$IPTABLES -A FORWARD -m state --state RELATED -m helper --helper ftp -j ↵
        ACCEPT
```

16.9.6 iprange

In vielen Fällen möchte man prüfen, ob eine IP-Adresse sich innerhalb eines bestimmten Bereichs befindet. Jedoch kann in vielen Fällen nicht ein Netzwerk mit Subnetzmaske verwendet werden, da der Bereich nicht an Netzwerkgrenzen beginnt und endet. Dann können Sie diesen Test verwenden. Sie können die Quell- (`--src-range`) und Ziel-IP-Adresse (`--dst-range`) prüfen:

```
-m iprange [!] --src-range <ip>-<ip>
-m iprange [!] --dst-range <ip>-<ip>
```

Wenn Ihr DHCP-Server zum Beispiel im Bereich von 192.168.0.100-150 dynamische IP-Adressen verteilt und Sie Regeln für die dynamischen Clients definieren möchten, können Sie folgenden Ausdruck verwenden: `-m iprange --src-range 192.168.0.100-192.168.0.150`.

16.9.7 length

Dieser Test prüft einfach die Länge eines Pakets. Sie können mit `--length` einen spezifischen Wert oder einen Bereich angeben:

```
-m length --length <länge>[:<länge>]
```

Damit ist es zum Beispiel möglich, ungewöhnlich große Ping-Pakete zu erkennen, die auf einen versteckten Kommunikationskanal hinweisen:

```
$IPTABLES -A FORWARD -p icmp --icmp-type echo-request -m length --length ↵
        100:1500 -j LOG --log-prefix "Unusual Ping Size: "
```

16.9.8 limit

Dies ist der klassische Limit-Test des Iptables-Kommandos. Dieser Test ist in modernen Linux-Kerneln durch den `hashlimit`-Test im Wesentlichen abgelöst worden. Eine Regel, die diesen Test verwendet, trifft so lange zu, bis das Limit für die Regel erreicht wurde. Dabei besitzt die Regel nur einen einzigen Zähler, der unabhängig von den verwendeten IP-Adressen und -Ports die Ereignisse zählt. Hashlimit kann hier für jede IP-Adresse und jeden Port eigene Zähler verwalten. Dieser Test verfügt über zwei Optionen:

- `--limit <rate>`: Dies ist die maximale Paketrate. Sie können die Einheit in `/second`, `/minute`, `/hour` oder `/day` angeben. Default ist `3/hour`.
- `--limit-burst <rate>`: Hiermit können Sie einen initialen Schwellenwert definieren, der erst erreicht werden muss, bevor das Limit greift. Der Default-Wert ist 5.

Dieser Test eignet sich speziell zur Beschränkung der Protokollierung einzelner Pakete.

```
$IPTABLES -A FORWARD -p icmp --icmp-type echo-request  -m limit --limit ↵
        1/minute -j LOG --log-prefix "Ping-Paket: "
```

16.9.9 mac

Mit diesem Test können Sie die Quell-MAC-Adresse (`--mac-source`) testen. So können Sie sich vor ARP-Spoofing schützen, da Sie nur Pakete annehmen, die von einer bestimmten Quell-IP- und -MAC-Adresse stammen.

```
-m mac --mac-source <XX:XX:XX:XX:XX:XX>
```

Wenn Sie diese Überprüfung für viele Systeme durchführen möchten, sollten Sie sich den `ipset`-Befehl mit der `macipmap` genauer ansehen (siehe Kapitel 26).

16.9.10 mark

Sie können in der Mangle-Tabelle Pakete mit dem `MARK`-Target markieren. Diese Firewall-Markierung können Sie mit dieser Erweiterung prüfen. Hierzu können Sie die Markierung (`--mark`) genau angeben oder zusätzlich auch noch eine Maske definieren, die vor der Überprüfung mit der Markierung der Pakete Und-verknüpft wird.

```
-m mark --mark <markierung>[/<maske>]
```

Die Paket-Markierung ist nur auf dem lokalen System gültig. Sobald das Paket das System verlässt, ist die Markierung verloren.

16.9.11 multiport

Diese Option erlaubt es Ihnen, mehrere Quell- oder Zielports in einer Regel anzugeben. Normalerweise erlaubt Iptables nur die Angabe eines Ports oder eines Port-Bereichs. Aktuelle Kernel (>= `2.6.11`) können bei dieser Option auch Port-Bereiche verwenden. Insgesamt dürfen Sie 15 Ports angeben. Ein Port-Bereich zählt als zwei Ports.

- `--source-ports [!] <port>[,<port>[,<port>:<port>]]`
- `--destination-ports [!] <port>[,<port>[,<port>:<port>]]`
- `--ports [!] <port>[,<port>[,<port>:<port>]]`

Mit diesem Test können Sie mehrere Regeln in einer zusammenfassen:

```
$IPTABLES -A FORWARD -p tcp -m multiport --destination-ports ↵
        21,22,25,80,443 -j ACCEPT
```

16.10 osf

OpenBSD hat es vorgemacht, und nun ist es auch unter Linux möglich: Passives Betriebssystem-Fingerprinting in den Firewall-Regeln. Sie können mit diesem Patch Regeln aufsetzen,

die nur für bestimmte Betriebssysteme gelten. Einigen Linux-Fanatikern mag die folgende Regel gefallen:

```
iptables -A FORWARD -p tcp -m osf --genre Windows -j REJECT
```

Dieser TCP-Patch kennt folgende Optionen:

» `--genre <os>`

» `--log <0|1|2>`: Wenn Sie diese Option (`0`) angeben, protokolliert die Regel das Betriebssystem, wenn es nicht mit dem in der Regel angegebenen übereinstimmt. Ist der Wert `1`, so wird nur das erste mögliche nicht übereinstimmende Betriebssystem protokolliert. Bei einer `2` werden alle möglichen Betriebssysteme protokolliert.

```
ipt_osf: Windows [2000:SP3:Windows XP Pro SP1, 2000 SP3]: ↵
        11.22.33.55:4024 -> 11.22.33.44:139
```

» `--ttl <0|1|2>`. Hiermit geben Sie an, ob der TTL-Wert bei der Erkennung genutzt werden soll.

- 0: Hier werden die TTL-Werte genutzt und mit den Default-Werten des erkannten Betriebssystems verglichen. Dies funktioniert normalerweise recht gut in lokalen Netzen.
- 1: Hier wird geprüft, ob der TTL-Wert kleiner dem Default-Wert des erkannten Betriebssystems ist. Dies funktioniert recht gut für globale IP-Adressen.
- 2: Hier wird der TTL-Wert nicht genutzt.

Die notwendigen Fingerprints können Sie von `http://www.openbsd.org/cgi-bin/cvsweb/src/etc/pf.os` herunterladen. Diese Fingerprints müssen dann mithilfe des Kommandos `nfnl_osf` in den Kernel geladen werden.

16.10.1 owner

Dieser Test kann nur in der `OUTPUT`-Kette angewendet werden, da der Linux-Kernel nur den Erzeuger lokaler Pakete ermitteln kann. Mit diesem Test können Sie prüfen, welcher Benutzer (`--uid-owner`) und welche Gruppe (`--gid-owner`) ein Paket erzeugt hat. Damit können Sie zum Beispiel dem von dem Webserver genutzten Benutzer verbieten, selbst Verbindungen aufzubauen:

```
$IPTABLES -A OUTPUT -m owner --uid-owner apache -j REJECT
```

16.10.2 physdev

Diese Option wird bei dem Linux-Kernel 2.6 verwendet, um die in einer Bridge verwendeten Netzwerkkarten zu erkennen und in einer Regel zu testen. Die genaue Verwendung dieses Tests wird in Kapitel 30, „Iptables auf einer Bridge“, bereits erläutert und soll hier nicht wiederholt werden.

16.10.3 pkttype

Dieser Test prüft den Pakettyp der Schicht 2. Sie können damit prüfen, ob es sich um ein `unicast`-, `broadcast`- oder ein `multicast`-Paket handelt.

Mit diesem Test können Sie alle lokalen Broadcast-Pakete einfach auf Ihrer Firewall verwerfen.

```
$IPTABLES -A INPUT -m pkttype --pkt-type broadcast -j DROP
```

16.10.4 policy

Diese Erweiterung wird bei dem Linux-Kernel 2.6 verwendet, um IPsec-Verbindungen richtig zu filtern. Diese Erweiterung wird in Kapitel 34 genauer betrachtet und soll daher hier nicht wiederholt werden.

16.10.5 quota

Mit diesem Patch können Sie Netzwerkquoten vergeben. Dazu definieren Sie eine Quote mit der Option `--quota <bytes>`. Jedes Paket verringert die Quote. Ist die Quote erreicht, werden keine weiteren Pakete mehr akzeptiert.

16.10.6 rateest

Der Rate Estimator (Bandbreitenschätzer) kann basierend auf den vom `RATEEST`-Target gesammelten Daten die Bandbreiten schätzen.

- `--rateest1 name`: Name der ersten Bandbreitenschätzung.
- `--rateest2 name`: Name der zweiten Bandbreitenschätzung bei Berechnung einer Differenz
- `--rateest-delta`: Vergleiche die beiden Werte.
- `--rateest-bps1 value`: Vergleiche die erste Schätzung mit diesem Wert in Bytes pro Sekunde.
- `--rateest-bps2 value`: Vergleiche die zweite Schätzung mit diesem Wert in Bytes pro Sekunde.
- `--rateest-pps1 value`: Nutze anstelle von Bytes/Sekunde die Maßeinheit Pakete/Sekunde.
- `--rateest-pps2 value`: CE
- `--rateest-lt`: Prüfe, ob die erste Rate geringer als die zweite Rate ist.
- `--rateest-gt`: Prüfe, ob die erste Rate größer als die zweite Rate ist.
- `--rateest-eq`: Prüfe, ob beide Raten gleich groß sind.

Auch dieser Match kann am besten anhand des Beispiels der Iptables-Manpage erläutert werden. Hierbei gehen wir von einem FTP-Server aus, der über zwei Anbindungen an das Internet verfügt. Diese Anbindungen sollen gleichmäßig ausgenutzt werden. Der FTP-Server

CE Hier fehlt vermutlich Text. Bitte ergänzen.

baut aktive Datenverbindungen zu den Clients auf, über die er die einzelnen angeforderten Dateien an die Clients versendet. Dabei handelt es sich um eine Ethernet-(eth0-) und eine DSL-(ppp0-)Verbindung. Zunächst müssen die aktuellen Bandbreiten ermittelt werden. Dies übernehmen die beiden folgenden Zeilen und das RATEEST-Target:

```
iptables -t mangle -A POSTROUTING -o eth0 -j RATEEST --rateest-name eth0 ↵
        --rateest-interval 250ms --rateest-ewma 0.5s
iptables -t mangle -A POSTROUTING -o ppp0 -j RATEEST --rateest-name ppp0 ↵
        --rateest-interval 250ms --rateest-ewma 0.5s
```

Nun können neue Datenverbindungen in Abhängigkeit von der aktuell genutzten Bandbreite markiert werden. Hierzu wird eine benutzerdefinierte Kette namens balance genutzt. Die folgenden Zeilen prüfen, ob die ppp0- oder die eth0-Verbindung über mehr freie Bandbreite verfügt. Dazu ermittelt der Test die Differenz aus der aktuellen Bandbreite und der maximalen Bandbreite (ppp0: 2mbit, eth0: 2.5mbit). Steht auf der Ethernet-Verbindung mehr Bandbreite zur Verfügung, so wird die Verbindung mit 1 markiert. Ansonsten erfolgt die Markierung mit dem Wert 2.

```
iptables -t mangle -A balance -m conntrack --ctstate NEW -m helper -- ↵
        helper ftp -m rateest --rateest-delta --rateest1 eth0 --rateest- ↵
        bps1 2.5mbit --rateest-gt --rateest2 ppp0 --rateest-bps2 2mbit - ↵
        j CONNMARK --set-mark 1
iptables -t mangle -A balance -m conntrack --ctstate NEW -m helper -- ↵
        helper ftp -m rateest --rateest-delta --rateest1 ppp0 --rateest- ↵
        bps1 2mbit  --rateest-gt --rateest2 eth0 --rateest-bps2 2.5mbit  ↵
        -j CONNMARK --set-mark 2
```

Um diese Markierung der Verbindung in eine Markierung der Pakete umzuwandeln, die anschließend auch von dem Routing-Code genutzt werden kann, ist die folgende Zeile noch nötig:

```
iptables -t mangle -A balance -j CONNMARK --restore-mark
```

16.10.7 realm

Dynamische Routing-Protokolle wie das Border-Gateway-Protokoll (BGP) verwenden *Routing Realms* für die Konfiguration. Dieser Test kann prüfen, ob das Paket zu einem bestimmten Realm (--realm) gehört. Wenn Sie auf Ihrem System kein BGP einsetzen, benötigen Sie diesen Test nicht.

```
-m realm --realm [!] <realm>[<maske>]
```

16.10.8 recent

Diese Erweiterung erlaubt es Ihnen, dynamisch eine Liste der gerade aktiven Quell-IP-Adressen zu erzeugen und in Ihren Regeln zu testen. Der Test hat die folgenden Optionen:

- `--name <name>`: Sie können verschiedene Listen für unterschiedliche Zwecke anlegen. Wenn Sie den Namen nicht angeben, wird die `DEFAULT`-Liste verwendet.
- `[!] --set`: Dies fügt die IP-Adresse zur Liste hinzu. Falls die IP-Adresse bereits in der Liste enthalten ist, wird der Eintrag aktualisiert. Dieser Test ist immer wahr, außer Sie geben das Ausrufezeichen an.
- `[!] --rcheck`: Dies prüft, ob die aktuelle IP-Adresse in der Liste enthalten ist.
- `[!] --update`: Identisch mit `--rcheck`, jedoch wird zusätzlich der Eintrag aktualisiert.
- `[!] --remove`: Hiermit entfernen Sie eine IP-Adresse aus der Liste. Falls die IP-Adresse nicht in der Liste vorhanden ist, trifft die Regel nicht zu.
- `[!] --seconds <sekunden>`: Diese Option können Sie gemeinsam mit `--rcheck` oder `--update` verwenden. Diese Optionen treffen dann nur zu, wenn die IP-Adresse innerhalb der angegebenen Zeitdauer gesehen wurde.
- `[!] --hitcount <treffer>`: Diese Option kann gemeinsam mit `--rcheck` und `--update` verwendet werden und prüft, ob die IP-Adresse bereits so häufig gesehen wurde. Mit `--seconds` können Sie zusätzlich den Zeitraum einschränken.
- `--rttl`: Hiermit prüfen Sie, ob der TTL-Wert des aktuellen Pakets identisch mit dem Paket ist, das mit `--set` hinzugefügt wurde.

Das Modul legt in `/proc/net/xt_recent/<name>`[3] Dateien an, die auch direkt gelesen und geschrieben werden können. Um eine IP-Adresse hinzuzufügen, können Sie diese in die Datei schreiben:

```
echo +<ip> > /proc/net/ipt_recent/<name>
```

Genauso können Sie auch eine IP-Adresse entfernen:

```
echo -<ip> > /proc/net/ipt_recent/<name>
```

Um die gesamte Liste zu löschen, schreiben Sie den Slash `/` in die entsprechende Datei. Das Modul `xt_recent` unterstützt beim Laden auch noch einige Optionen. Die Klammern geben die Default-Werte an:

- `ip_list_tot`: Anzahl der IP-Adressen pro Liste (100).
- `ip_pkt_list_tot`: Anzahl der Pakete pro IP-Adresse (20).
- `ip_list_hash_size`: Größe der Hash-Tabelle (0, Berechnung in Abhängigkeit von der `ip_list_tot`).
- `ip_list_perms`: Rechte der Dateien in `/proc/net/ipt_recent/*` (0644).

3 Auf älteren Kerneln kann diese Datei auch `/proc/net/ipt_recent/<name>` heißen.

» `ip_list_uid` und `ip_list_gid` definieren den Eigentümer und die Gruppe der Dateien.

» `debug`: Debug-Informationen (0).

Um diese Optionen zu setzen, müssen Sie das Modul laden, bevor Sie in Ihren Regeln den Test verwenden:

```
modprobe xt_recent ip_list_tot=200
```

Dieses Modul können Sie nun einsetzen, um sich in Ihren Regeln spezifische IP-Adressen zu merken und zusätzliche Regeln auf diese IP-Adressen anzuwenden.

Um zum Beispiel Brute-Force-SSH-Angriffe abzuwehren, könnten Sie anstelle des `hashlimit`-Tests auch die folgenden beiden Regeln verwenden:

```
iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent --name ↵
         SSH --set

iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent --name ↵
         SSH --update --seconds 60 --hitcount 4 -j DROP
```

Die erste Regel fügt jede neue Quell-IP-Adresse, die eine SSH-Verbindung öffnet, der Liste `SSH` hinzu. Die zweite Regel prüft, ob von einer IP-Adresse innerhalb der letzten 60 Sekunden 4 Verbindungsanfragen erhalten wurden, und verwirft alle weiteren Anfragen. Sie können diese Regeln einsetzen, wenn der `hashlimit`-Test auf Ihrem Kernel nicht zur Verfügung steht.

16.10.9 sctp

Das Stream Control Transmission Protocol (SCTP, RFC 2960) ist ein Protokoll, das ähnlich wie TCP die Datenübertragung garantiert und verlorene oder beschädigte Pakete automatisch erneut sendet. Im Gegensatz zu TCP unterstützt SCTP mehrere Streams in einer Verbindung sowie Multihoming. Multihoming bedeutet, dass ein Kommunikationsendpunkt unter verschiedenen IP-Adressen erreicht werden kann. SCTP wird aktuell von MPI-Parallel-Programmen und Telefonie-Software genutzt.

Mit diesem Test können Sie die Eigenschaften des Protokolls testen. Hierzu haben Sie die folgenden Optionen:

» `--source-port [!] <port>[:<port>]`

» `--destination-port [!] <port>[:<port>]`

» `--chunk-types [!] all|any|only <chunktype>[:<flags>]`

16.10.10 set

`set` ist der Nachfolger des obsoleten `pool`-Patches. Sie benötigen für die Anwendung zusätzlich den `ipset`-Befehl, den Sie ebenfalls auf der Netfilter-Homepage finden (`http://ipset.netfilter.org`). Diese Erweiterung erlaubt es Ihnen, Gruppen von IP-Adressen zu definieren und diese in einer Regel zu testen. Die IP-Adressen können auch von Iptables-Regeln

dynamisch zu diesen Gruppen hinzugefügt werden. Diese Erweiterung ist so interessant und wichtig, dass sie in einem eigenen Kapitel besprochen wird (siehe Kapitel 26, „Ipset“).

INFO

Die meisten Kernel unterstützen diese Funktion aktuell nicht. Teilweise gibt es ein Xtables-Add-Ons-Paket, das diese Funktion nachrüstet.

16.10.11 socket

Diese Erweiterung prüft, ob es zu dem gegebenen Paket einen lokalen offenen Socket gibt.

```
-m socket
```

16.10.12 state

Dieser Test ermittelt den Zustand der Verbindung, zu der dieses Paket gehört. Die folgenden Zustände sind möglich:

- `NEW`: Das Paket gehört zu einer neuen Verbindung.
- `ESTABLISHED`: Das Paket gehört zu einer aufgebauten Verbindung.
- `RELATED`: Das Paket bezieht sich auf eine aufgebaute Verbindung (z. B. ICMP-Fehlernachricht).
- `INVALID`: Der Zustand des Pakets kann nicht ermittelt werden (z. B. ICMP-Fehlernachricht, die sich nicht auf eine bekannte Verbindung bezieht).

Wenn Ihr Kernel über die `raw`-Tabelle verfügt, existiert zusätzlich noch der Zustand `UNTRACKED`. Der Zustand dieser Verbindung wird von dem Kernel nicht überwacht. Weitere Informationen finden Sie in Abschnitt 5.5 und 5.10.

16.10.13 statistic

Diese Erweiterung bietet einige statistische Funktionen. Da diese nur in Testfällen interessant sind, verweise ich hier auf die Manpage.

16.10.14 string

Ab dem Kernel 2.6.14 der `string`-Test im Linux-Kernel fest verfügbar. Während ältere Versionen des `iptables`-Kommandos den Test nicht kannten, ist er heute in den aktuellen Versionen enthalten.

Er verwendet die folgende Syntax:

- `--from <offset>`: Suche ab diesem Offset im Paket.
- `--to <offset>`: Suche bis zu diesem Offset im Paket.
- `--algo <algorithmus>`: Der Kernel 2.6.14 unterstützt im Moment zwei verschiedene Algorithmen: Boyer-Moore (BM) und Knuth-Morris-Pratt (KMP).
- `--icase`: Ignoriere die Groß- und Kleinschreibung

» `--string <muster>`: Suche diese Zeichenkette.

» `--hex-string <muster>`: Suche diese hexadezimal notierte Zeichenkette.

16.10.15 tcpmss

Hiermit können Sie prüfen, ob in dem Paket eine Maximum Segment Size (MSS) gesetzt wurde. Dieser Test ist nur bei TCP-Paketen erlaubt. Sie können exakt einen Wert (`--mss`) oder einen Bereich prüfen.

```
-m tcpmss [!] --mss <wert>[:<wert>]
```

Dieser Test ist nicht besonders hilfreich. Ich empfehle, bei Problemen mit der MSS grundsätzlich in allen Paketen die MSS zu ändern. Hierzu können Sie das `MSS`-Target einsetzen (siehe Abschnitt 17.25).

16.10.16 time

Mit diesem Patch können Sie Regeln definieren, die nur zu bestimmten Zeiten aktiv sind. Anstatt die Regelsätze per Cron austauschen zu lassen, können Sie die Uhrzeiten direkt in den Regeln hinterlegen. Zusätzlich erlaubt der Test folgende Optionen:

» `--timestart HH:MM`

» `--timestop HH:MM`

» `--days Mon,Tue,Wed,Thu,Fri,Sat,Sun`

» `--datestart YYYY[:MM[:DD[:hh[:mm[:ss]]]]]`

» `--datestop YYYY[:MM[:DD[:hh[:mm[:ss]]]]]`

» `--monthdays 1-31`

» `--weekdays Mon - Sun`

» `--utc`. Als Zeitbasis wird UTC verwendet.

» `--localtz`. Als Zeitbasis wird die lokale Uhrzeit verwendet (Default).

16.10.17 tos

Dieser Test ermöglicht die Prüfung des Type-of-Service-Wertes eines Pakets. Meines Erachtens gibt es keine wirklich sinnvolle Anwendung.

```
-m tos --tos <tos>
```

Sie können die folgenden Werte mit `--tos` testen:

```
$ iptables -m tos -h
TOS match v1.3.0 options:
[!] --tos value Match               Type of Service field from one of the ↵
         following numeric or descriptive values:
               Minimize-Delay      16 (0x10)
               Maximize-Throughput  8 (0x08)
               Maximize-Reliability 4 (0x04)
               Minimize-Cost        2 (0x02)
               Normal-Service       0 (0x00)
```

16.10.18 ttl

Mit der `ttl`-Erweiterung können Sie den TTL-Wert eines Pakets prüfen. Sie können den genauen Wert (`--ttl-eq`) testen oder prüfen, ob der Wert größer (`--ttl-gt`) oder kleiner (`--ttl-lt`) einem Vergleichswert ist. Hiermit können Sie zum Beispiel Windows-Systeme im lokalen Netz von Linux-Systemen unterscheiden, da Windows-Systeme eine andere initiale TTL verwenden. Es gibt aber bessere Methoden (z. B. `osf` in Patch-O-Matic, siehe Abschnitt 16.10).

```
-m ttl --ttl-eq <ttl> # gleich
-m ttl --ttl-gt <ttl> # größer
-m ttl --ttl-lt <ttl> # kleiner
```

16.10.19 u32

Wenn Sie immer schon mal etwas in einem Paket testen wollten, aber der Test dafür bisher nicht in Netfilter enthalten war, ist dies der Test für Sie. Hiermit können Sie ein beliebiges Bit an einer beliebigen Stelle in einem IP-Paket prüfen.

Hierzu stellt das Modul den Test `--u32` zur Verfügung. Diese Option erlaubt Ihnen die Beschreibung des Pakets in einer sehr mächtigen Sprache. Die Syntax wird im Kernel-Quelltext wie folgt angegeben:

```
--u32 tests
tests := location = value | tests && location = value
value := range | value , range
range := number | number : number
location := number | location operator number
operator := & | << | >> | @
```

Es gibt ein paar Beschränkungen bei der Anwendung dieses Moduls:

» Sie dürfen nur maximal 10 = (Gleichheitszeichen) und damit 9 && als Argument verwenden.

» Pro Wert (value) dürfen Sie nur maximal 10 Bereiche (range) verwenden.

» Pro Ort (location) dürfen Sie ebenfalls nur maximal 10 Nummern (number) angeben.

Es werden immer 4 Bytes von dem angegebenen Ort gelesen. Zunächst ein einfaches Beispiel. Im IP-Header befindet sich das Längenfeld in den Bytes 2 und 3 des Headers. Um nun die ersten 4 Bytes des Pakets zu lesen, verwenden Sie als Ortsangabe 0. Da Sie sich jedoch nur für die Bytes 2 und 3 interessieren, wird dieses Ergebnis mit 0x0000FFFF Und-verknüpft. Damit fallen die beiden Bytes 0 und 1 weg: 0&0xFFFF. Wenn Sie prüfen möchten, ob die Länge des Pakets kleiner als 512 Bytes ist, so können Sie dann den folgenden Test verwenden: 0&0xFFFF=0x0:0x200. Die Angabe 0x0:0x200 prüft, ob der Wert innerhalb dieses Bereichs (0x200=512) liegt.

Ein weiteres kompliziertes Beispiel prüft, ob die Bytes 0–3 in den TCP-Daten den Wert 25 oder 50 haben. Der komplette Test lautet: 6&0xFF=6 && 0»22&0x3C@12»26&0x3C@0=25,50. Schauen wir ihn uns der Reihe nach an. Zunächst ermitteln wir, ob es sich überhaupt um ein TCP-Paket handelt. Dies erkennen wir im IP-Header an dem Next-Protocol-Feld. Dieses Feld ist das Byte 9 im Header. Wir lesen ab dem Byte 6 die Länge von 4 Bytes und Und-verknüpfen dies mit 0xFF. Ist der Wert 6, handelt es sich um ein TCP-Paket, und wir machen weiter in dem Test.

Nun müssen wir eigentlich prüfen, ob es sich um ein IP-Fragment handelt. Da aber Iptables meist unter Einsatz der Connection-Tracking-Funktion verwendet wird, sind die Pakete in den Tabellen Mangle, NAT und Filter bereits defragmentiert. Fragmente treffen wir hier nicht mehr an. Sie könnten aber diesen Test einfügen: 4&0x3FFF=0. Er liest ab dem Byte 4 die Länge von 4 Bytes und extrahiert die letzten 6 Bits von Byte 6 und das komplette Byte 7. Hier werden das MF-Bit und der Fragment-Offset gespeichert. Sind diese 0, so handelt es sich um ein komplettes Paket. Möchten Sie das erste Fragment genauso prüfen wie ein komplettes Paket, müssen Sie einfach nur die letzten 5 Bits des Byte 6 analysieren: 4&0x1FFF=0. Im ersten Fragment ist das MF schon gesetzt, aber der Offset noch null.

Um die gesuchten Bytes in den TCP-Daten zu finden, müssen Sie nun die Länge des IP-Headers und die Länge des TCP-Headers ermitteln und entsprechend im Paket weiterspringen. Leider wird die Länge im IP-Header in 4-Byte-Worten angegeben. Das bedeutet, dass Sie diese Zahl noch mit 4 multiplizieren müssen. Die Länge wird in den letzten 4 Bits des ersten Bytes gespeichert. Dazu liest der Test 0»22&0x3c@ die Bytes 0–3 und verschiebt diese um 22 Bits nach rechts. Es bleiben 10 Bits. Wäre die Verschiebung um 24 Bits erfolgt, so wäre nur das erste Byte übrig geblieben. Da wir um 2 Bits weniger schieben, wird dieses Byte aber automatisch mit $2^2 = 4$ multipliziert. Jetzt muss der Test nur noch die restlichen gesetzten Bits entfernen. Hierfür führt er eine Und-Verknüpfung mit 0x3c (0000111100) durch. Stellen Sie sich vor, der IP-Header hätte seine Standardlänge von 20 Bytes. Dann befindet sich bei einem IPv4-Paket im ersten Byte die Zahl 0x45, binär: 01000101. Nach der Verschiebung bleiben 10 Bits: 01000101yy. Der Wert der Bits yy ist unbekannt. Nun führen Sie eine Und-Verknüpfung mit 0x3c (0000111100) durch und es bleibt: 0000010100 = 0x14 = 20. Das @ im

Test definiert diese Zahl als neuen Offset für den nächsten Test. Dieser funktioniert analog zu `12»26&0x3C@`. Die Länge des TCP-Headers wird wieder in 4-Byte-Worten in der linken Hälfte des Bytes 12 im TCP-Header angegeben. Sie lesen also ab Byte 12 die Länge von 4 Bytes und verschieben nun um 26 Bits nach rechts. Dadurch bleiben 6 Bits übrig. Die 4 höchstwertigen Bits sind automatisch mit 4 multipliziert worden. Nun setzen Sie die beiden niedrigwertigen Bits auf null, indem Sie eine Und-Verknüpfung mit 0x3c durchführen. Dies können Sie wieder als neuen Offset verwenden. Jetzt liest der Test ab diesem Offset wieder 4 Bytes und prüft, ob deren Inhalt entweder 25 oder 50 ist (`0=25,50`).

Wie Sie sehen, können Sie mit diesem Test alle nur denkbar möglichen Tests durchführen. Um dies auf ganze Verbindungen anzuwenden, wäre es zum Beispiel möglich, in Abhängigkeit von diesem Test eine Verbindung zu markieren, um sie anschließend anders zu behandeln.

17. Alle Ziele

Dieses Kapitel führt alle Ziele (Aktionen, Targets) auf, die in dem Linux-Kernel 2.6.35 enthalten sind.

Im Folgenden werden die verschiedenen Ziele alphabetisch aufgeführt. Da einzelne Ziele nur in bestimmten Tabellen und Ketten erlaubt sind, wurden sie bereits in vielen Fällen in den entsprechenden Kapiteln erläutert. Hier wird dann nur auf das Kapitel verwiesen.

17.1 ACCEPT

Dieses Ziel akzeptiert ein Paket. Das bedeutet: Das Paket durchläuft die Kette erfolgreich.

17.2 CLASSIFY

Dieses Ziel ist nur in der Mangle-Tabelle erlaubt und wird daher bei dieser besprochen (siehe Abschnitt 21.3).

17.3 CLUSTERIP

Dies ist ein neues experimentelles Ziel der aktuellen Linux 2.6-Kernel. Mit diesem Ziel können Sie einen Cluster mit Lastverteilung und Hochverfügbarkeit ohne Loadbalancer aufbauen. Dies ist kein Firewall-Cluster, sondern kann zum Beispiel als Web- oder E-Mailserver genutzt werden.

Hierzu benötigen Sie einen Switch, der Multicast-Linklayer-Adressen unterstützt und Pakete an diese Adressen an mehrere Ports versenden kann. Die meisten professionellen Switches unterstützen diese Funktion. Möglicherweise müssen Sie die Funktion aktivieren.

Das `CLUSTERIP`-Target sorgt dann dafür, dass ARP-Anfragen auf die Cluster-IP-Adresse mit einer Cluster-MAC-Adresse beantwortet werden. Diese MAC-Adresse ist eine Multicast-MAC-Adresse. Multicast-MAC-Adressen beginnen mit `01:00:5e:xx:xx:xx`. Anschließend werden die Anfragen an den Cluster an diese MAC-Adresse gesendet. Der Switch wird die Pakete an alle Ports senden, auf denen die MAC-Adresse registriert ist. Die Nodes verwenden intern eine Markierung der Verbindungen, um zu erkennen, welche Verbindungen sie akzeptieren und welche Verbindungen sie ignorieren müssen. Hierzu muss jeder Node wissen, wie viele Nodes es insgesamt in dem Cluster gibt und welche Nummer er selbst besitzt.

Sie müssen auf jedem Node lediglich eine Regel für das CLUSTERIP-Target aufrufen:

```
# Node 1 iptables -A INPUT -d 192.168.0.200 -j CLUSTERIP --new  -- ↵
        hashmode sourceip --clustermac 01:00:5e:11:11:11 --total-nodes 2 ↵
         --local-node 1

# Node 2 iptables -A INPUT -d 192.168.0.200 -j CLUSTERIP --new  -- ↵
        hashmode sourceip --clustermac 01:00:5e:11:11:11 --total-nodes 2 ↵
         --local-node 2
```

Bei der Definition einer neuen Cluster-IP-Adresse müssen Sie in der ersten Regel immer die Option `--new` verwenden. Mit der Option `--hashmode` wählen Sie den Verteilungsmechanismus aus. Zur Verfügung steht `sourceip`, bei dem die Verbindungen in Abhängigkeit von der Quell-IP-Adresse verteilt werden. Verbindungen von derselben IP-Adresse landen bei demselben Node. Außerdem gibt es `sourceip-sourceport` und `sourceip-sourceport-destport`, die zusätzlich noch die Ports in die Verteilung mit einbeziehen. Die Option `--clustermac` definiert die identische Multicast-MAC-Adresse auf allen Nodes. Iptables kümmert sich dann selbst um die ARP-Antworten. Sie müssen keine darüber hinausgehende Konfiguration der MAC-Adresse vornehmen. Die Optionen `--total-nodes` und `--local-node` definieren schließlich die Anzahl der Nodes im Cluster und die Nummer des lokalen Nodes. Bei Bedarf können Sie mit `--hash-init` auch noch eine Zahl für die Initialisierung der Hash-Tabelle angeben.

Wichtig ist, dass Sie auf beiden Nodes die identische Cluster-IP-Adresse und Cluster-MAC-Adresse verwenden. Eine darüber hinausgehende Konfiguration der Netzwerkkarten ist nicht erforderlich.

Sobald die Regeln aktiviert wurden, finden Sie in `/proc/net/ipt_CLUSTERIP/192.168.0.200` eine Datei, die die Nummer des Nodes enthalten sollte.

Damit ist die Lastverteilung bereits implementiert. Um zusätzlich auch eine Hochverfügbarkeit zu erreichen, benötigen Sie eine Software, die die Gesundheit der Nodes in dem Cluster überwacht (z. B. Heartbeat, `http://www.linux-ha.org`). Diese muss bei dem Ausfall eines Nodes auf dem jeweils anderen Node nur einen Befehl ausführen. Beim Ausfall des Nodes 1 muss auf Node 2 der folgende Befehl gestartet werden:

```
echo "+1" > /proc/net/ipt_CLUSTERIP/192.168.0.200
```

Damit erhält der Node 2 die Information, auch die Verbindungen für Node 1 zu übernehmen. Dies funktioniert jedoch nur für neue Verbindungen. Bereits aufgebaute Verbindungen gehen verloren.

17.4 CONNMARK

Dieses Ziel ist nur in der NAT- und in der Mangle-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 20 und Kapitel 21).

17.5 CONNSECMARK

Dieses Ziel ist nur in der Mangle-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 21).

17.6 DNAT

Dieses Ziel ist nur in der `PREROUTING`- und `OUTPUT`-Kette der `nat`-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 20).

17.7 DROP

Dieses Ziel verwirft ein Paket. Das Paket wird aus der Kette entfernt. Es erfolgt keine Protokollierung oder Benachrichtigung des Absenders.

17.8 DSCP

Dieses Ziel ist nur in der `mangle`-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 21).

17.9 ECN

Dieses Ziel ist nur in der `mangle`-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 21).

17.10 LOG

Hiermit können Sie eine Regel erzeugen, die eine Protokollierung über den Syslog auslöst. Dieses Target unterscheidet sich von den meisten anderen Targets dadurch, dass es nicht das Schicksal des Pakets entscheidet. Daher werden die Pakete, auf die das `LOG`-Target angewendet wird, von den weiteren Regeln in der Kette analysiert.

Für eine aussagekräftige Protokollierung unterstützt dieses Target die folgenden Optionen:

» `--log-level <level>`: Hiermit definieren Sie die Priorität der Meldung (`debug`, `info`, `notice`, `warning`, `error`, `crit`, `alert`, `emerg`).

» `--log-prefix „Zeichenkette“`: Hiermit können Sie bis zu 29 Zeichen angeben, die der Meldung vorangestellt werden. Dies erleichtert später die Suche mit `grep` oder die Protokollierung mit dem Syslog-ng oder ähnlichen Syslog-Daemons.

» `--log-tcp-sequence`: Dies protokolliert die TCP-Sequenznummern.

» `--log-tcp-options`: Dies protokolliert verwendete TCP-Optionen.

» `--log-ip-options`: Dies protokolliert verwendete IP-Optionen.

» `--log-uid`: Dies protokolliert bei lokal erzeugten Paketen den Benutzer, der das Paket erzeugt hat.

Wenn Sie ein Paket protokollieren und verwerfen möchten, benötigen Sie zwei identische Regeln. Die erste Regel testet das Paket und protokolliert es. Die zweite Regel testet das Paket auf dieselbe Weise und verwirft es. Um hier den zusätzlichen Aufwand des doppelten Tests zu sparen, erzeugen Sie sich einfach eine benutzerdefinierte Kette `LOGDROP`:

```
$IPTABLES -N LOGDROP
$IPTABLES -A LOGDROP -j LOG --log-prefix "Log-and-Drop: "
$IPTABLES -A LOGDROP -j DROP
```

Wenn Sie nun ein Paket protokollieren und verwerfen möchten, verwenden Sie das Target `-j LOGDROP`.

17.11 MARK

Dieses Ziel ist nur in der `mangle`-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 21).

17.12 MASQUERADE

Dieses Ziel ist nur in der `POSTROUTING`-Kette der `nat`-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 20).

17.13 NETMAP

Dieses Ziel ist nur in der `nat`-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 20).

17.14 NFLOG

Hiermit können Sie eine Regel erzeugen, die eine Protokollierung über den Syslog auslöst. Dieses Target unterscheidet sich von den meisten anderen Targets dadurch, dass es nicht

das Schicksal des Pakets entscheidet. Daher werden die Pakete, auf die das NFLOG-Target angewendet wird, von den weiteren Regeln in der Kette analysiert. Dieses Target ist eine Weiterentwicklung des ULOG-Targets auf modernen Kerneln.

17.15 NFQUEUE

Dieses Target ist eine Weiterentwicklung des QUEUE-Targets, das auf modernen Kerneln nicht mehr zur Verfügung steht. Hiermit können Sie Pakete an unterschiedliche Userspace-Programme übergeben, da Sie bei diesem Target die Queue über eine 16-Bit-Zahl auswählen können.

17.16 NOTRACK

Dieses Ziel ist nur in der raw-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 22).

17.17 RATEEST

Dieses Target erlaubt die Sammlung von statistischen Informationen. Diese können dann als Grundlage für eine Bandbreitenabschätzung genutzt werden.

- `--rateest-name name`. Hiermit werden die Pakete gezählt. Der Name ist frei wählbar.
- `--rateest-interval amount<s|ms|us>`. Diese Option definiert den Zeitraum, über den die Zählung erfolgt.
- `--rateest-ewma value`. Dies ist der Zeitraum, über den die Gewichtung erfolgt.

17.18 REDIRECT

Dieses Ziel ist nur in der PREROUTING- und der OUTPUT-Kette der NAT-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 20).

17.19 REJECT

Das REJECT-Target verwirft das Paket wie das DROP-Target. Es sendet jedoch eine Fehlermeldung an den Absender des Pakets. Dieses Target ist nur gültig in den INPUT-, FORWARD- und OUTPUT-Ketten. Sie können die zu verwendene Fehlermeldung mit der Option `--reject-with` angeben. Sie können die folgenden Fehlermeldungen benutzen:

- `icmp-net-unreachable`
- `icmp-host-unreachable`
- `icmp-port-unreachable` (Default)
- `icmp-proto-unreachable`

» icmp-net-prohibited

» icmp-host-prohibited

» icmp-admin-prohibited (Der Kernel muss dies unterstützen.)

» tcp-reset (Die Ablehnung muss sich auf eine TCP-Verbindung beziehen.)

```
$IPTABLES -A INPUT -p tcp --dport 113 -j REJECT --reject-with tcp-reset
```

17.20 RETURN

Dieses Ziel beendet die aktuelle benutzerdefinierte Kette und kehrt mit dem Paket zur aufrufenden Kette zurück, wo die restlichen Regeln abgearbeitet werden. Handelt es sich um eine eingebaute Kette, so wird bei einem RETURN die Default-Policy der Kette für das Paket ausgewertet.

17.21 SAME

Dieses Ziel ist nur in der nat-Tabelle erlaubt und wird daher dort besprochen (siehe 20). Es wurde jedoch in den neuen Kernel durch die Option --persistant bei dem DNAT- und SNAT-Ziel abgelöst.

17.22 SECMARK

Dieses Ziel ist nur in der Mangle-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 21).

17.23 SET

Dieses Ziel wird in dem Kapitel zum ipset Kommando besprochen (siehe Kapitel 26).

17.24 SNAT

Dieses Ziel ist nur in der POSTROUTING-Kette der NAT-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 20).

17.25 TCPMSS

Dieses Target ist eine der wichtigsten Funktionen von Iptables für alle Anwender von ADSL-Leitungen. Hiermit können Sie die Maximum Segment Size (MSS) für TCP-Verbindungen setzen oder ändern.

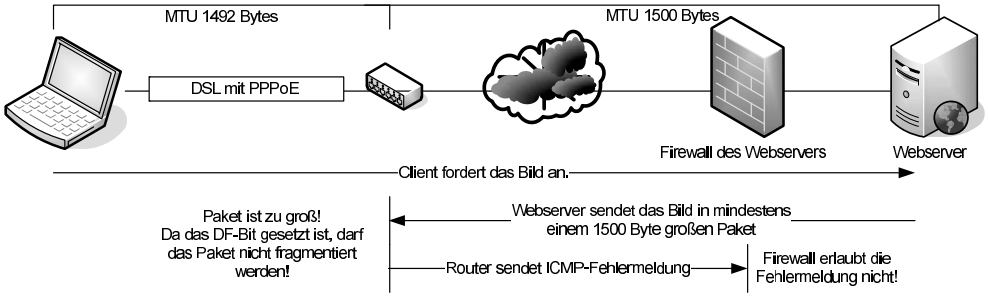


Abbildung 17.1: **Bei dem Einsatz von DSL können Probleme mit der MTU auftreten.**

INFO

Was ist die MSS und wofür brauche ich sie bei DSL?

DSL-Verbindungen werden üblicherweise mit dem PPPoE- oder PPTP-Protokoll realisiert. Als physikalisches Medium wird hierfür Ethernet eingesetzt. Jedes Medium besitzt eine Maximum Transmission Unit (MTU). Dies ist die maximale Größe der Pakete, die über das Medium transportiert werden können. Bei Ethernet ist die MTU 1500 Bytes groß. Wenn Sie nun das PPP-over-Ethernet-Protokoll (PPPoE) verwenden, so benötigt das PPP-Protokoll selbst 8 Bytes für seinen PPP-Header. Es bleiben 1492 Bytes für die transportierten Informationen. Also beträgt die MTU von PPPoE 1492 Bytes. Dass das darunterliegende Ethernet eine MTU von 1500 Bytes besitzt, ist für den Datentransport unerheblich, da wir für den Transport ja PPPoE und nicht direkt Ethernet verwenden. Betrachten Sie nun Abbildung 17.1. Dort sehen Sie ein Netzwerk, das mit dem Internet über DSL verbunden ist, wobei die DSL-Verbindung das Protokoll PPPoE nutzt. Im Internet befindet sich ein Webserver, der durch eine Firewall geschützt wird.

Ein Client in dem Netzwerk möchte nun auf den Webserver zugreifen. Der Client baut die Verbindung auf und fordert von dem Webserver ein Bild in der Größe von 800 Bytes an. Dies erzeugt keinerlei Probleme, da das resultierende Paket nur wenig größer als 800 Bytes wird und von PPPoE mit einer MTU von 1492 Bytes ohne Probleme transportiert werden kann. Sobald wir jedoch von dem Webserver ein Bild oder eine andere Datei anfordern, die größer als 1500 Byte ist, kann ein Problem auftreten. Der Webserver baut die IP-Pakete entsprechend der MTU seiner eigenen Netzwerkkarte. Setzen wir voraus, dass der Webserver über Ethernet angebunden ist, beträgt diese MTU 1500 Bytes. Der Webserver baut daher wenigstens ein Paket von 1500 Bytes. Ist die Datei sehr groß, können es auch mehrere Pakete sein. Dies sendet er durch seine Firewall an den DSL-Router auf der Seite des Providers. Dieser muss nun das Paket in PPPoE einpacken und über DSL versenden. Das Paket ist aber mit 1500 Bytes 8 Bytes zu groß. Die MTU von PPPoE beträgt ja nur 1492. Alle modernen Betriebssysteme nutzen die Path MTU Discovery, bei der sie in allen Paketen das DF-Bit im IP-Header setzen. Dieses Don't-Fragment-Bit verbietet einem Router die Fragmentierung eines zu großen Pakets. Er muss das Paket verwerfen und eine Fehlermeldung (`ICMP Destination Unreachable Fragmentation Needed`) zurücksenden. Die Fehlermeldung enthält auch die maximal erlaubte Größe des Pakets. Dann kann der Absender das Paket erneut mit der richtigen Größe versenden. Stellen Sie sich vor, die Firewall, die den Webserver schützt, lässt dieses Paket nicht durch. Die Fehlermeldung erreicht den Webserver nicht. Das originale Paket hat den Client aber auch nicht erreicht. Da der Webserver keine Bestätigung über den Empfang

des Pakets erhält, wird er es nach einiger Zeit unverändert neu versenden. Das Paket wird wieder verworfen, und die Firewall verwirft die Fehlermeldung. Die Verbindung hängt! Mit einem Trick können Sie das verhindern. Das TCP-Protokoll bietet Ihnen die Möglichkeit, bei der Verbindungsaufnahme Ihre Maximum Segment Size zu veröffentlichen. Die MSS entspricht in etwa der MTU für TCP. Wenn Sie eine MSS von 1400 an den Kommunikationspartner senden, wird dieser nie mehr als 1400 Bytes in einem TCP-Paket versenden. Das resultierende IP-Paket erhält zusätzlich nur noch den TCP-Header und den IP-Header. Beide betragen im Normalfall jeweils 20 Bytes. Das IP-Paket wird dann nicht größer als 1440 Bytes. Damit vergeuden Sie jedoch ein paar Bytes. Wir müssen ja nur sicherstellen, dass 1492 Bytes nicht überschritten werden. Also subtrahieren Sie zweimal 20 Bytes für den IP- und den TCP-Header und erhalten 1452 Bytes. Wenn Sie dafür sorgen, dass die MSS immer auf 1452 Bytes in allen TCP-Verbindungen gesetzt wird, haben Sie das Problem behoben.

Für andere Tunnel (IPsec, PPTP etc.) müssen Sie möglicherweise mit anderen Werten experimentieren. Ich habe auch schon PPPoE-Tunnel erlebt, die geringere Werte verlangt haben. Dort wurde zusätzlich noch ein weiteres Protokoll eingesetzt.

Die MSS kann nur in SYN-Paketen gesetzt werden. Sie haben zwei Möglichkeiten für das Setzen. Entweder Sie errechnen selbst die optimale MSS und verwenden mit dem Target `TCPMSS` die Option `--set-mss <mss>`, oder Sie überlassen die Rechnerei dem Kernel. Mit der Option `--clamp-mss-to-pmtu` subtrahiert der Kernel selbstständig 40 Bytes von der Path-MTU und setzt die entsprechende MSS.

```
iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-
        to-pmtu
```

17.26 TCPOPTSTRIP

Hiermit können Sie TCP-Optionen entfernen (`--strip-options`).

17.27 TEE

Das `TEE` Target dient dazu, Kopien der Pakete an einen weiteren Rechner zu senden. Dies kann zum Beispiel dazu verwendet werden, um auf einem Router/Firewall-System ein Intrusion-Detection-System anzubinden und mit Kopien sämtlicher Pakete zu versorgen.

Als `--gateway` geben Sie bei diesem Target den Rechner an, der die Kopie der Pakete erhalten soll.

```
-t mangle -A PREROUTING -i eth0 -j TEE --gateway 192.168.0.7
```

17.28 TOS

Dieses Ziel ist nur in der `mangle`-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 21).

17.29 TPROXY

Dieses Target (siehe auch Kapitel 28) erlaubt den Aufbau echt transparenter Proxies. Es ist nur in der Mangle-Tabelle erlaubt, daher wird es auch dort besprochen.

17.30 TRACE

Dieses Target erlaubt die Verfolgung von Paketen in den Regeln. Es ist nur in der `raw`-Tabelle erlaubt. Daher wird es dort besprochen (siehe Kapitel22.

17.31 TTL

Dieses Ziel ist nur in der Mangle-Tabelle erlaubt und wird daher dort besprochen (siehe Kapitel 21).

17.32 ULOG

Dieses Ziel wird in Kapitel 24, „Fortgeschrittene Protokollierung“, besprochen.

18. Patch-O-Matic

Der Netfilter/Iptables-Code befindet sich in konstanter Entwicklung. Die von den Distributoren verteilten Kernel und die auf `http://www.kernel.org` zur Verfügung gestellten Kernel enthielten in der Vergangenheit nicht alle verfügbaren Funktionen. Diese zusätzlichen Funktionen wurden im Patch-O-Matic gepflegt. In den letzten Jahren wurden diese Funktionen aber fast alle aufgenommen. Das Patch-O-Matic wurde daher eingestellt.

19. Connection Tracking

Das Connection Tracking ist eine wesentliche Funktion der Iptables/Netfilter-Firewall. Deshalb gebührt dieser Funktion auch ein eigenes Kapitel. Da diese Funktionalität aber für viele andere Funktionen (zustandsorientiertes Filtern, Network Address Translation) benötigt wird, wurden in den entsprechenden Kapiteln bereits ausführlich verschiedene Aspekte betrachtet. Hier soll es daher nur um die fortgeschrittene Konfiguration des Connection Trackings gehen.

19.1 Connection Tracking – Überblick

Das Connection Tracking ist für die Überwachung der Netzwerkverbindungen durch die Netfilter-Firewall verantwortlich. Das Connection Tracking überwacht sämtliche IP- und IPv6-Protokolle. Hierzu erkennt das Connection Tracking, aus welcher Richtung das erste Paket kommt. Dieses Paket erhält den virtuellen Zustand `NEW`. Wenn das Paket von den Firewall-Regeln akzeptiert wird, wird die Verbindung auch in der Zustandstabelle eingetragen. Das zweite Paket muss aus der anderen Richtung kommen. Nur dann erhalten das zweite Paket und die Verbindung in der Zustandstabelle den virtuellen Zustand `ESTABLISHED`. Alle weiteren Pakete erhalten unabhängig von ihrer Richtung den Zustand `ESTABLISHED`. Diese Funktion kann genutzt werden, um Verbindungen nur in bestimmten Richtungen zu erlauben.

Das Connection Tracking unterscheidet insgesamt 5 verschiedene Zustände:

`NEW`: Ein Paket ist neu, wenn die dazugehörige Verbindung noch nicht in der Verbindungstabelle existiert und das Paket bestimmte andere Eigenschaften hat, die für ein neues Paket verlangt werden. Diese Eigenschaften sind von dem Protokoll abhängig und werden weiter unten besprochen.

`ESTABLISHED`: Ein Paket hat den Zustand `ESTABLISHED`, wenn die Verbindung bereits in der Verbindungstabelle existiert und mindestens ein Antwortpaket bereits von dem Connection Tracking gesehen wurde oder dieses Paket das erste Antwortpaket ist.

`RELATED`: Ein Paket ist mit einer Verbindung verwandt, wenn es, obwohl es ein anderes Protokoll, eine andere IP-Adresse und andere Ports verwendet, sich auf eine Verbindung bezieht, die den Zustand `ESTABLISHED` hat. Dies können ICMP-Fehlermeldungen sein oder Verbindungen, die von einem Connection Tracking-Helfermodul hinzugefügt wurden. Auf modernen Kerneln werden die letzten Verbindungen in einer eigenen Verbindungstabelle `/proc/net/nf_conntrack_expect` verwaltet.

`INVALID`: Falls ein Paket zu keiner der in der Verbindungstabelle aufgeführten Verbindungen passt, aber dennoch nicht die Anforderungen für eine neue Verbindung bei dem

verwendeten Protokoll erfüllt, ist es ungültig. Dies können zum Beispiel ICMP-Fehlermeldungen sein, die sich auf eine Verbindung beziehen, die in der Verbindungstabelle nicht vorhanden ist.

`UNTRACKED`: Wenn Ihr Kernel über die `raw`-Tabelle verfügt, kennt das Connection Tracking auch den Zustand `UNTRACKED`. Das bedeutet, dass diese Verbindung nicht von dem Connection Tracking überwacht wird und daher auch nicht in der Verbindungstabelle auftaucht.

Da die verschiedenen Protokolle unterschiedliche Arten der Verbindung unterstützen und diese unterschiedlich auf- und abbauen, unterscheidet das Connection Tracking im Linux-Kernel auch diese Protokolle in der Behandlung.

19.1.1 Das TCP-Connection-Tracking

Das TCP-Connection-Tracking erkennt eine neue Verbindung daran, dass die verwendeten IP-Adressen und Ports des Pakets mit keiner der in der Verbindungstabelle eingetragenen Verbindungen übereinstimmen. Zusätzlich muss das Paket mit dem Zustand `NEW` entweder ein `SYN`- oder ein `ACK`-Paket sein.

Damit die Verbindung anschließend in den Zustand `ESTABLISHED` versetzt wird, muss auf ein `SYN`-Paket immer ein `SYN`/`ACK` -Paket aus der entgegengesetzten Richtung folgen. Wurde die Verbindung mit einem `ACK`-Paket im Zustand `NEW` begonnen, so befindet sich die Verbindung anschließend direkt in dem Zustand `ESTABLISHED`, ohne dass ein Antwort-Paket erforderlich ist. Dies erlaubt die Wiederaufnahme von Verbindungen nach einem Reboot der Firewall oder nach einem Failover in einem Firewall-Cluster. Weitere Informationen hierzu finden Sie in Abschnitt 27.3, „Hochverfügbarkeit bei zustandsorientierten Firewalls“.

Die aufgebaute Verbindung verbleibt für bis zu 5 Tage in der Verbindungstabelle, wenn keine weiteren Pakete von dem Connection Tracking erkannt werden. Jedes weitere Paket setzt die Lebensdauer des Eintrags in der Tabelle wieder auf 5 Tage zurück. Sobald das Connection Tracking ein TCP-`RST`-Paket oder ein TCP-`FIN`-Paket erkennt, wird die Verbindung auch aus der Zustandstabelle entfernt. Um diese verschiedenen Zustände sauber trennen zu können, verwaltet das Connection Tracking intern die Zustände viel detaillierter und versieht die einzelnen Zwischenzustände auch mit unterschiedlichen Timeouts. Diese verschiedenen Zwischenzustände werden in Abschnitt 19.4, „TCP-Zustände“, besprochen.

Zusätzlich analysiert das TCP-Connection-Tracking auch die TCP-Windows. Hierbei analysiert es die von den Kommunikationspartnern angegebenen TCP-Windows und die Sequenznummern der Pakete. Alle Pakete, die eine Sequenznummer verwenden, die nicht in die aktuellen TCP-Windows passt, werden von dem Connection Tracking als `INVALID` gekennzeichnet.

Die Funktionsweise des TCP-Window-Trackings basiert auf dem Artikel von Guido van Rooij „Real Stateful TCP Packet Filtering in IP-Filter“ (`http://www.usenix.org/events/sec01/invitedtalks/rooij.pdf`). Van Rooij beschreibt diese Funktionalität in dem IP-Filter für BSD und Solaris. Wenn Sie sich für den Hintergrund interessieren, ist dies eine sehr ausführliche Quelle zu den angestellten Überlegungen.

CE[d] Dies scheint etwas kurz für ein Kapitel. Ist dieser Abschnitt eventuell anderswo günstiger einzuordnen?

Die Überwachung der TCP-Windows und Sequenznummern kann auch abgeschaltet werden (siehe Abschnitt 19.3.10).

19.1.2 Das UDP-Connection-Tracking

Das Protokoll UDP unterscheidet sich von dem Protokoll TCP dadurch, dass dieses Protokoll das Konzept einer Verbindung nicht kennt. Daher gibt es auch keine Signalisierung eines Verbindungsaufbaus oder -abbaus in dem UDP-Protokoll. Die Überwachung der Verbindung kann nur durch reine Beobachtung der Pakete und Verwendung von Timeouts erfolgen.

Das Connection Tracking erkennt eine neue Verbindung daran, dass das Paket IP-Adressen und Portnummern verwendet, die zu keiner in der Tabelle gespeicherten UDP-Verbindungen passen. Ist das der Fall, erhält das Paket den Zustand `NEW`, und die Verbindung wird, wenn die Regeln es erlauben, als neue Verbindung in die Tabelle eingetragen.

Da es durchaus Applikationen gibt, die über UDP lediglich Informationen versenden und nie eine Antwort erhalten, muss das Connection Tracking die Verbindung selbstständig nach einiger Zeit wieder entfernen. Eine Verbindung, für die das Connection Tracking keine Antwort-Pakete aus der entgegengesetzten Richtung gesehen hat, wird automatisch nach 30 Sekunden aus der Tabelle entfernt. Kommt ein Antwort-Paket erst nach 31 Sekunden, so kommt es zu spät und wird nicht mehr als `ESTABLISHED` erkannt.

Erhält das Connection Tracking innerhalb der 30 Sekunden ein Antwort-Paket, so wird die Lebensdauer für die Verbindung in der Tabelle auf 180 Sekunden heraufgesetzt. Solange nun alle 180 Sekunden ein weiteres Paket mit identischen IP-Adressen und Portnummern ausgetauscht wird, verbleibt die Verbindung in der Tabelle. Ist das nicht der Fall, wird die Verbindung nach 180 Sekunden aus der Zustandstabelle entfernt.

19.1.3 Das ICMP-Connection-Tracking

Auch bei dem ICMP-Protokoll gibt es ähnlich wie beim UDP-Protokoll keine echten Verbindungen. Jedoch ist zum Beispiel der Ping mit den Nachrichten Echo-Request und Echo-Reply einer Verbindung ähnlich. Das Connection Tracking behandelt daher alle ICMP-Request- und -Reply-Pakete auch entsprechend. Sobald das Connection Tracking ein Request-Paket erkennt, trägt es dieses mit seinen IP-Adressen, seiner Identifikations- und Sequenznummer in der Verbindungstabelle ein. Das Request-Paket erhält den Zustand `NEW` zugewiesen. Die Identifikationsnummer erlaubt es, die Request-Pakete mehrerer gleichzeitiger Ping-Befehle auseinanderzuhalten. Die Sequenznummer ermöglicht es, die Reply-Pakete den entsprechenden Request-Paketen zuzuordnen. Das Connection Tracking richtet in der Verbindungstabelle für jedes eindeutige Request-Paket eine Verbindung ein. Sobald nun das Connection Tracking ein Reply-Paket erkennt, prüft es die IP-Adressen, die Identifikationsnummer und die Sequenznummer und vergleicht diese mit den Verbindungen in der Tabelle. Stimmen diese mit einer Verbindung überein, so erhält das Paket den Zustand `ESTABLISHED` und die Verbindung wird aus der Tabelle entfernt, denn pro Echo-Request-Paket ist nur ein Reply-Paket erlaubt. Stimmen die Daten nicht mit einer bekannten Verbindung überein, so erhält das Reply-Paket den Zustand `INVALID`. Damit die Verbindungen nicht unendlich lange in der Tabelle verbleiben,

wenn kein Reply-Paket gesendet wird, erhalten diese Verbindungen eine Lebensdauer von 30 Sekunden. Das Reply-Paket muss innerhalb dieser 30 Sekunden erkannt werden, sonst entfernt das Connection Tracking die Verbindung aus der Tabelle.

Das ICMP-Protokoll wird aber auch dafür verwendet, um Fehler in TCP- und UDP-Verbindungen anzuzeigen. Diese Fehlermeldungen können von IP-Adressen versendet werden, die mit der eigentlichen Verbindung nichts zu tun haben (z. B. Router). Das Connection Tracking erkennt die Verbindung, auf die sich die ICMP-Fehlermeldung bezieht, an dem Inhalt des ICMP-Pakets. Eine ICMP-Fehlermeldung muss immer den kompletten IP-Header und die ersten 64 Bits der IP-Daten enthalten. Bei TCP- oder UDP-Paketen befinden sich hier die Ports. Diese Header werden von dem Connection Tracking ausgewertet. Existiert eine Verbindung in der Tabelle, auf die diese Informationen passen, so erhält das Paket den Zustand `RELATED`, und die Verbindung, auf die sich die Fehlermeldung bezieht, wird aus der Tabelle entfernt.

19.1.4 Das Connection Tracking für alle weiteren Protokolle

Für alle weiteren Protokolle kennt das Connection Tracking keine besonderen Regelungen. Sie werden alle gleich behandelt. Ein neues Paket erkennt das Connection Tracking an der Tatsache, dass sich in der Tabelle noch keine Verbindung befindet, die die gleichen IP-Adressen und das gleiche Protokoll verwendet. Sobald sich eine derartige Verbindung in der Tabelle befindet, werden alle weiteren Pakete, die identische IP-Adressen und dasselbe Protokoll verwenden, mit dem Zustand `ESTABLISHED` versehen. Die Verbindung bekommt ab dem ersten Paket die Lebensdauer von 600 Sekunden zugewiesen. Das bedeutet, dass entweder innerhalb von 600 Sekunden ein weiteres Paket dieser Verbindung erkannt werden muss oder die Verbindung aus der Tabelle entfernt wird. Jedes weitere Paket setzt die Lebensdauer wieder auf 600 Sekunden zurück.

19.2 Das nf_conntrack-Kernelmodul

Das `nf_conntrack`-Kernelmodul unterstützt beim Laden die Definition des Parameters `hashsize`. Um diesen Parameter zu verstehen, müssen wir uns zunächst mit der Verbindungstabelle beschäftigen. Die Verbindungstabelle wird als Hash-Tabelle angelegt, damit der Kernel schnell in der Tabelle suchen kann. Während Sie die Größe der Verbindungstabelle während der Verwendung der Tabelle ändern können, ist die Größe des Hashes fix und wird bei der Erzeugung der Verbindungstabelle festgelegt. Anschließend kann die Größe des Hashs nicht geändert werden.

Auf einem normalen Linux-System berechnet der Kernel die Größe der Verbindungstabelle und des Hashs aus der Arbeitsspeichergröße. Auf der Intel-32-Bit-Architektur wird die Größe der Verbindungstabelle wie folgt berechnet:

```
CONNTRACK_MAX = RAM / 16384
```

Auf einem 512-Mbyte-Rechner hat die Verbindungstabelle also eine Größe von maximal 32.768 Verbindungen. Diese Größe sinkt unabhängig von der Speichergröße nie unter 128, und auch auf Systemen mit mehr als 1 Gbyte Speicher beträgt dieser Wert maximal immer 65.536. Ist

dieser Wert erreicht, werden weitere Verbindungen nicht mehr von der Firewall akzeptiert. Sie können diesen Wert aber manuell in der Variablen `nf_conntrack_max` setzen:

```
echo 128000 > /proc/sys/net/netfilter/nf_conntrack_max
```

Bei einigen Kernel müssen Sie alternativ auf diese Datei zugreifen:

```
echo 128000 > /proc/sys/net/ipv4/netfilter/ip_conntrack_max
```

Den aktuellen Wert können Sie hier auch auslesen. Die Größe des Hashs (`hashsize`) wird ebenfalls von dem Kernel berechnet und beträgt auf der Intel-32-Bit-Architektur *HASHSIZE* = *CONNTRACK_MAX*/8. Diese Berechnung wird nur einmal bei dem Laden des Moduls durchgeführt. Wenn Sie anschließend aber den Wert CONNTRACK_MAX erhöhen, ist die Größe des Hashs nicht mehr optimal, und die Suche nach einer Verbindung in der Tabelle durch den Kernel dauert unnötig lange.

Es gibt drei Möglichkeiten, dies zu verhindern:

» Aktuelle Kernel (ab 2.6.14) erlauben die Anpassung der Hashsize zur Laufzeit. Hierzu verwenden Sie eine der beiden folgenden Dateien:
  - 2.6.14-2.6.19: `/sys/module/ip_conntrack/parameters/hashsize`
  - ab `2.6.20`: `/sys/module/nf_conntrack/parameters/hashsize`

» Alternativ können Sie auch bei dem Laden des Moduls die Hashsize angeben:

```
modprobe nf_conntrack hashsize=16384
```

  Dies funktioniert natürlich nur, wenn `nf_conntrack` tatsächlich als Modul kompiliert wurde. Bei aktuellen Fedora-Distributionen ist diese Funktion fest im Kernel eingebaut.

» Sie geben die Hashsize als Boot-Option des Kernels an. Dies funktioniert auch, wenn die Funktionalität fest in den Kernel integriert wurde.

```
nf_conntrack.hashsize=16384
```

Zunächst müssen Sie entscheiden, wie viele Verbindungen Ihre Verbindungstabelle später unterstützen soll. Berechnen Sie ausgehend von dieser Zahl den passenden Wert für die Hashsize, indem Sie die maximale Anzahl der Verbindungen durch 8 teilen. Bei einem Kernel, der älter als Version 2.4.21 ist, sollten Sie aufgrund des verwendeten Hash-Algorithmus möglichst eine Primzahl benutzen. Seither verwenden die Kernel den Jenkins-Hash. Dieser arbeitet optimal mit einer Potenz von 2.

Wählen Sie eine geeignete Zahl aus, und verwenden Sie eine der drei oben aufgeführten Methoden, um die Hashsize zu setzen.

Auslesen und kontrollieren können Sie den Hashsize-Wert entweder aus den Kernelmeldungen bei dem Laden des Moduls oder, wenn Ihr Kernel es unterstützt, über die Datei `nf_conntrack_buckets` im `/proc`-Verzeichnis.

19.3 /proc-Variablen

Das gesamte Connection Tracking ist in den letzten Jahren mehrfach stark überarbeitet worden. Dabei wurde das ursprünglich nur für IPv4 verfügbare Connection Tracking neu geschrieben, sodass es nun auch protokollunabhängig IPv6 unterstützt. Um die Protokollunabhängigkeit auch im Namen nachvollziehbar zu machen, wurden die Kernelmodule und die Dateien im `/proc`-Verzeichnis von `ip_*` in `nf_*` umbenannt. Auch der Ort der Verzeichnisse hat sich geändert. Das ältere Connection Tracking hat ein Netfilter-Verzeichnis in dem Verzeichnis `/proc/sys/net/ipv4/` erzeugt. Nun befindet sich das Verzeichnis eine Ebene höher direkt in `/proc/sys/net/`. Die Bedeutung der Variablen ist im Wesentlichen aber gleich geblieben.

Denken Sie daran, dass nach einem Neustart des Systems diese Werte wieder auf Ihre Default-Werte zurückgesetzt werden. Es ist sinnvoll, zu Beginn Ihres Firewall-Skripts die Werte manuell zu setzen, die Sie verwenden möchten.

Dieses Verzeichnis enthält die folgenden neuen Variablen:

19.3.1 nf_conntrack_acct

Mit dieser Variablen schalten Sie das Accounting für das Connection Tracking ein. Eigentlich sollte diese Funktion bereits in Kernel 2.6.29 wieder entfernt werden. Jedoch benötigt die Erweiterung `connbytes` diese Funktion. Daher ist sie auch heute noch in den Kerneln enthalten. Wenn Sie diese Erweiterung nutzen, müssen Sie daher diese Variable auch aktivieren. Ansonsten können Sie darauf verzichten, da das Accounting lediglich Rechenleistung und Speicher benötigt und abgesehen von dieser Erweiterung keinen Nutzen hat.

19.3.2 nf_conntrack_buckets

In dieser Variablen können Sie die Hashsize der Verbindungstabelle auslesen (siehe Abschnitt 19.2).

19.3.3 nf_conntrack_checksum

Viele Netzwerkkarten können heute bereits die Prüfsummen von Paketen kontrollieren und ignorieren alle Pakete mit fehlerhaften Prüfsummen. Es ist daher in diesen Umgebungen nicht sinnvoll, dass Netfilter ebenfalls die Prüfsummen berechnet und kontrolliert. Fehlerhafte Pakete werden von der Netzwerkkarte bereits verworfen. Mit dieser Option können Sie diese Funktion abschalten. Ansonsten markiert das Connection Tracking diese Pakete als `INVALID`.

19.3.4 nf_conntrack_count

Diese Variable zeigt die Anzahl der aktuell in der Verbindungstabelle vorgehaltenen Verbindungen an.

19.3.5 nf_conntrack_generic_timeout

Diese Variable definiert das Timeout der Verbindungen, die nicht das TCP-, UDP- oder ICMP-Protokoll verwenden.

Default: 600 Sekunden.

19.3.6 nf_conntrack_icmp_timeout

Diese Variable definiert das Timeout von ICMP-Verbindungen wie Echo-Request/Echo-Reply.

Default: 30 Sekunden.

19.3.7 nf_conntrack_icmpv6_timeout

Diese Variable definiert das Timeout von ICMPv6-Verbindungen wie Echo-Request/Echo-Reply.

Default: 30 Sekunden.

19.3.8 nf_conntrack_log_invalid

Mit diesem Parameter können Sie die Protokollierung ungültiger Pakete für ein bestimmtes Protokoll anschalten. Hierzu müssen Sie die Nummer des Protokolls in diese Variable schreiben (TCP=6, siehe `/etc/protocols`). Um diese Funktion abzustellen, nutzen Sie das Protokoll 255.

Default: 0.

19.3.9 nf_conntrack_max

Diese Variable definiert die Größe Ihrer Verbindungstabelle. Mehr Verbindungen kann das Connection Tracking nicht überwachen. Sie können diesen Wert online ändern. Jedoch sollten Sie dann auch die Hashsize anpassen und den Abschnitt über das `nf_conntrack`-Kernelmodul (siehe Abschnitt 19.2) lesen.

19.3.10 nf_conntrack_tcp_be_liberal

Hiermit schalten Sie die Überwachung der Sequenznummern und der TCP-Windows für alle Pakete außer für `RST`-Pakete ab. Wenn Sie Netzwerkgeräte verwenden, die sich nicht ganz an den TCP-Standard halten, kann dies notwendig sein. Außerdem ist dies bei einer hochverfügbaren Firewall mit dem `conntrackd` erforderlich (siehe Abschnitt 27.5).

Default: 0.

19.3.11 nf_conntrack_tcp_loose

Wie bereits mehrfach erwähnt wurde, akzeptiert der Linux-Kernel auch ein `ACK`-Paket als ein neues Paket. Das können Sie mit dieser Variable abschalten. Setzen Sie dazu diese Variable

einfach auf 0. Wenn Sie das Verhalten wünschen, können Sie hier angeben, wie viele Pakete in beide Richtungen geflossen sein müssen, bevor das TCP-Window-Tracking aktiv werden darf.

Default: 1.

19.3.12 nf_conntrack_tcp_max_retrans

Dieser Wert definiert die maximale Anzahl von wiederholten TCP-Paketen ohne Bestätigung des Empfängers. Sobald dieser Wert erreicht wurde, fängt der `nf_conntrack_timeout_max_retrans` an zu zählen.

Default: 3.

19.3.13 nf_conntrack_tcp_timeout_close

Diese Variable definiert, wie lange eine Verbindung in dem Zustand `CLOSE` verbleibt (siehe Abschnitt 19.4).

Default: 10 Sekunden.

19.3.14 nf_conntrack_tcp_timeout_close_wait

Diese Variable definiert, wie lange eine Verbindung in dem Zustand `CLOSE_WAIT` verbleibt (siehe Abschnitt 19.4).

Default: 60 Sekunden.

19.3.15 nf_conntrack_tcp_timeout_established

Diese Variable definiert, wie lange eine Verbindung in dem Zustand `ESTABLISHED` verbleibt (siehe Abschnitt 19.4).

Default: 432000 Sekunden; das entspricht 5 Tagen.

19.3.16 nf_conntrack_tcp_timeout_fin_wait

Diese Variable definiert, wie lange eine Verbindung in dem Zustand `FIN_WAIT` verbleibt (siehe Abschnitt 19.4).

Default: 120 Sekunden.

19.3.17 nf_conntrack_tcp_timeout_last_ack

Diese Variable definiert, wie lange eine Verbindung in dem Zustand `LAST_ACK` verbleibt (siehe Abschnitt 19.4).

Default: 30 Sekunden.

19.3.18 nf_conntrack_tcp_timeout_max_retrans

Der Timeout für die Verbindung in der Verbindungstabelle, wenn nur Paketwiederholungen ohne Bestätigung des Empfängers beobachtet wurden.

Default: 300 Sekunden.

19.3.19 nf_conntrack_tcp_timeout_syn_recv

Diese Variable definiert, wie lange eine Verbindung in dem Zustand `SYN_RECV` verbleibt (siehe Abschnitt 19.4).

Default: 60 Sekunden.

19.3.20 nf_conntrack_tcp_timeout_syn_sent

Diese Variable definiert, wie lange eine Verbindung in dem Zustand `SYN_SENT` verbleibt (siehe Abschnitt 19.4).

Default: 120 Sekunden.

19.3.21 nf_conntrack_tcp_timeout_time_wait

Diese Variable definiert, wie lange eine Verbindung in dem Zustand `TIME_WAIT` verbleibt (siehe Abschnitt 19.4).

Default: 120 Sekunden.

19.3.22 nf_conntrack_udp_timeout

Hiermit definieren Sie den Timeout einer UDP-Verbindung, für die noch nicht Pakete in beiden Richtungen erkannt wurden.

Default: 30 Sekunden.

19.3.23 nf_conntrack_udp_timeout_stream

Hiermit definieren Sie den Timeout einer UDP-Verbindung, für die bereits Pakete in beiden Richtungen erkannt wurden.

Default: 180 Sekunden.

19.4 TCP-Zustände

Das Connection Tracking unterscheidet für TCP-Verbindungen die folgenden Zustände:

1. SYN_SENT: Ein erstes SYN-Paket wurde erkannt.
2. SYN_RECV: Das SYN/ACK-Paket wurde erkannt. Dies ist das zweite Paket des TCP-Handshakes.
3. ESTABLISHED: Das dritte Paket des TCP-Handshakes, das erste ACK-Paket, wurde erkannt.
4. FIN_WAIT: Das erste FIN-PAKET wurde erkannt. Die Verbindung wurde abgebaut.
5. CLOSE_WAIT: Das zweite FIN-Paket wurde erkannt.
6. LAST_ACK: Das letzte ACK-Paket als Antwort auf das zweite FIN-Paket wurde erkannt.
7. TIME_WAIT: Nach Ablauf des Zustands LAST_ACK wechselt die Verbindung in den Zustand TIME_WAIT, um noch Pakete zu akzeptieren, die während der Übertragung in eine falsche Reihenfolge gebracht wurden.
8. CLOSE: Nach dem Ablauf des Zustands TIME_WAIT wechselt die Verbindung in den Zustand CLOSE, um sicherzustellen, dass es nicht zum Konflikt mit neuen Verbindungen kommen kann.

Der Übergang der verschiedenen Zustände wird durch das entsprechende Paket ausgelöst. Lediglich die letzten Übergänge von LAST_ACK nach CLOSE erfolgen durch den Ablauf des Timeouts. Kommt es vorher zum Ablauf eines anderen Timeouts, wird die Verbindung aus der Tabelle entfernt!

20. Die nat-Tabelle

Mit Iptables können Sie auch eine NetworkAddress-Translation bei IPv4-Paketen durchführen. Hierfür verfügt der Kernel über eine eigene NAT-Tabelle, in der sich drei Ketten mit unterschiedlichen Aufgaben befinden: `PREROUTING`, `OUTPUT` und `POSTROUTING`.

20.1 Die NAT-Tabelle und ihre Ketten

Die NAT-Tabelle verfügt über drei Ketten: `PREROUTING`, `OUTPUT` und `POSTROUTING` (siehe Abbildung 20.1). Jede dieser drei Ketten hat eine besondere Aufgabe bei der NetworkAddressTranslation. Diese Aufgabe wurde bereits in Abschnitt 5.7 besprochen. Ich möchte sie hier aber kurz wiederholen.

Um die Tabelle anzuzeigen oder die enthaltenen Ketten zu modifizieren, müssen Sie immer bei dem Iptables-Befehl die Tabelle spezifisch angeben:

```
# iptables -vnL -t nat
Chain OUTPUT (policy ACCEPT 1937 packets, 94415 bytes)
pkts bytes target prot opt in out source destination

Chain POSTROUTING (policy ACCEPT 1934 packets, 94214 bytes)
pkts bytes target prot opt in out source destination

Chain PREROUTING (policy ACCEPT 27 packets, 3364 bytes)
pkts bytes target prot opt in out source destination
```

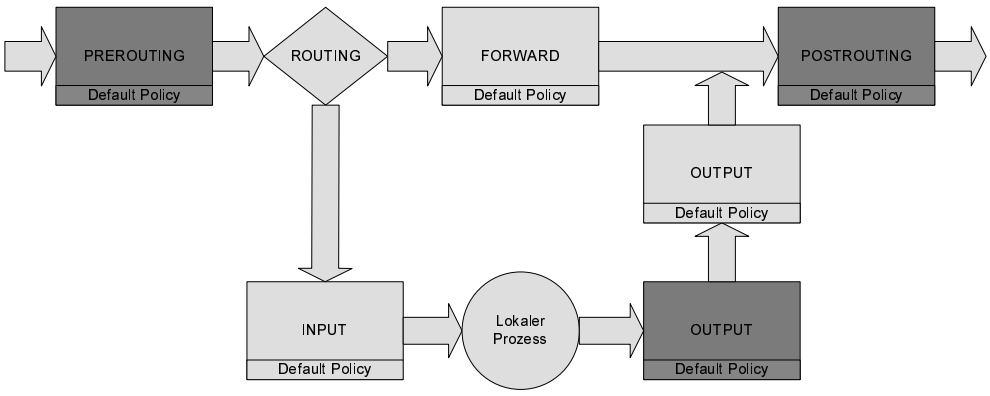


Abbildung 20.1: **Die NAT-Tabelle hat drei Ketten.**

Bei der NetworkAddressTranslation ändert das System die Adressierung eines Pakets. Es gibt grundsätzlich zwei verschiedene Arten der NetworkAddressTranslation (NAT):

» Source-NAT: Hier wird die Absenderadresse geändert.

» Destination-NAT: Hier wird die Zieladresse geändert.

Ein Source-NAT ist nur in der POSTROUTING-Kette der NAT-Tabelle erlaubt. Ein Destination-NAT ist nur in der PREROUTING- und der OUTPUT-Kette erlaubt. Anhand von Abbildung 20.1 wird auch sofort der Sinn klar. Bei dem Source-NAT ändert die Firewall die Absenderadresse. Da das in der POSTROUTING-Kette erfolgt, ist sichergestellt, dass die Filterregeln in der FORWARD- und der OUTPUT-Kette das Paket vor der Adressänderung analysieren. Ihre Filterregeln betrachten also das originale Paket mit der „echten" Absenderadresse.

Bei dem Destination-NAT ändert die Firewall die Zieladresse. Dies passiert in der PREROUTING- oder der OUTPUT-Kette der NAT-Tabelle und damit vor der Analyse durch die entsprechenden Filterketten. Bei diesem NAT wird die scheinbare Zieladresse durch eine neue tatsächliche Zieladresse ausgetauscht. Die Filtertabelle betrachtet nun also wieder das Paket mit der „echten" Zieladresse!

INFO

Die NAT-Ketten arbeiten anders als die Ketten der anderen Tabellen. Während die Ketten der anderen Tabellen jedes Paket einer Verbindung sehen und analysieren, ist das bei den NAT-Ketten nicht der Fall. Lediglich das erste Paket einer Verbindung wird von den NAT-Ketten und ihren Regeln analysiert. Wurde die Verbindung von dem Connection Tracking erkannt, die durchzuführende Adressumsetzung ermittelt und die Verbindung in die Tabelle aufgenommen, durchlaufen die weiteren Pakete der Verbindung die Tabelle nicht mehr. Diese Tabelle ist daher auch ein sehr schlechter Ort für die Filterung oder die Zählung von Paketen. Sie könnten hiermit nur Verbindungen zählen.

Die in der NAT-Tabelle verfügbaren Ziele hängen von der Art des NAT ab. Für ein Source-NAT können Sie die folgenden Ziele verwenden: MASQUERADE, NETMAP, SAME und SNAT. Bei dem Destination-NAT stehen die folgenden Ziele zur Verfügung: BALANCE, DNAT, NETMAP, REDIRECT, SAME und TPROXY. Die Ziele NETMAP und SAME können sowohl für ein Source-NAT als auch für ein Destination-NAT eingesetzt werden. Zusätzlich kann in der nat-Tabelle auch das CONNMARK-Target eingesetzt werden.

INFO

Linux ist bei dem NAT in beiden Richtungen sehr intelligent. Sie müssen mit Ihren Regeln immer nur das NAT in der Richtung des Verbindungsaufbaus definieren. Die Rückrichtung wird dann automatisch vom Kernel richtig gemacht. Bei einem SNAT müssen Sie also nur eine Regel definieren, die die Absenderadresse austauscht. Dass bei den Antwortpaketen der Verbindung dann die Zieladresse ausgetauscht werden muss, damit das Paket den Client erreicht, weiß das System dann automatisch. Bei dem DNAT ist es genauso. Sie müssen lediglich eine Regel definieren, mit der das Ziel ausgetauscht wird. Bei den Antwortpaketen wird dann automatisch der Absender wiederhergestellt.

20.2 Source-NAT

Das Source-NAT ändert die Absenderadresse eines Pakets. Das bedeutet, dass das Paket anschließend eine andere Absenderadresse besitzt. Dies ist häufig bei der Anbindung von Netzwerken an das Internet erforderlich. Lokale Netzwerke werden üblicherweise unter Zuhilfenahme von sogenannten privaten IPv4-Adressen aufgebaut. Diese Adressen werden in dem RFC1918 definiert. Es handelt sich um die folgenden Adressbereiche:

» 10.0.0.0–10.255.255.255
» 172.16.0.0–172.31.255.255
» 192.168.0.0–192.168.255.255

Sobald Sie Ihre eigenen Netze mit diesen IP-Adressen aufbauen, erzeugen Sie bei einer Anbindung an das Internet keinen IP-Adresskonflikt, da diese nicht im Internet genutzt werden. Für kleine Netze ist der Bereich 192.168.0.0/24 üblich.

Wenn Sie Ihr Netzwerk mit diesen privaten IP-Adressen aufbauen, müssen Sie jedoch am Übergang in das Internet die IP-Adressen umsetzen (NetworkAddressTranslation). Tun Sie das nicht, erhalten Sie auf Ihre Anfragen keine Antwort, denn der Client sendet das Paket mit seiner privaten Absenderadresse über seinen Router in das Internet, wo es zum Beispiel einen Webserver erreicht. Der Webserver beantwortet das Paket und schickt sein Antwortpaket an die private Adresse des Clients. Da diese Adressen jedoch im Internet verboten sind, verfügen die Internet-Router auch nicht über die notwendigen Routen. Das Paket wird mit der Fehlermeldung „Ziel nicht erreichbar" (Destination unreachable) verworfen.

INFO

Unter Umständen können Sie dennoch derartige Pakete über das Internet routen. Das IP-Protokoll erlaubt ein Source Routing. Dabei definiert der Absender die Router, über die das Paket sein Ziel erreicht. Selbst wenn der Router nicht über eine Route verfügt, kann er so das Paket weitersenden. Viele Router unterbinden aber heute das Source-Routing, da es als Sicherheitslücke angesehen wird.

Für diesen Zweck wurde nun das Source-NAT entwickelt. Dabei kann der Router des Clients dessen private IP-Adresse gegen eine offizielle IP-Adresse austauschen, deren Ort im Internet bekannt ist. Damit die Antworten auch bei dem Router wieder ankommen, verwendet man hier die offizielle Internet-Adresse des Routers selbst. Grundsätzlich wäre es auch möglich, hier irgendeine andere Adresse einzutragen, jedoch würden die Antwortpakete des Webservers nicht bei dem Router landen und damit nie den Client erreichen.

Sobald Sie nur eine offizielle IP-Adresse für das Source-NAT von mehreren Clients verwenden, handelt es sich eigentlich nicht mehr um ein NAT (Network Address Translation), sondern um ein NAPT (Network Address and Port Translation).

Bei dem NAPT wird nicht nur die Quell-IP-Adresse, sondern auch der Quellport geändert. Dies ist notwendig, da bei einem SNAT sich die Firewall auch für die Antwortpakete den Rückweg merken muss. Dies erfolgt über den Source-Port. Die Firewall weist jeder genatteten Verbindung eine eindeutige Kombination aus Source-IP-Adresse und -Port zu (siehe Abbildung 5.15). Wenn zwei Clients identische Source-Ports verwenden, versucht die Firewall bei

der ersten Verbindung den Source-Port beizubehalten. Die Firewall speichert diese Information in ihrer Verbindungstabelle. Bei der zweiten Verbindung verändert die Firewall nicht nur die IP-Adresse, sondern auch den Quellport, um die Antwortpakete den richtigen Clients wieder zuordnen zu können.

Bei der Wahl der Ports behandelt Iptables die Portbereiche unterschiedlich. Zunächst versucht der Kernel, immer den Quellport beizubehalten. Ist jedoch der Port bereits belegt, so versucht der Kernel, Ports <512 auf andere Ports <512 zu natten. Ports zwischen 512 und 1023 werden auf Ports wieder in diesem Bereich genattet. Hohe Ports werden durch andere hohe Ports ausgetauscht. Das können Sie durch spezifische Angaben der Ports teilweise modifizieren.

INFO

Dies ist übrigens auch der Grund, warum die IPsec-Protokolle[1] *und GRE (Generic Routing Encapsulation) so schwer mit SNAT zu behandeln sind. Diese Protokolle besitzen keine Ports. Bei dem NAT von zwei Verbindungen fehlt der Quellport als Differenzierungskriterium für die Antworten.*

20.3 Destination-NAT

Das Destination-NAT ist seltener anzutreffen, aber mindestens genauso interessant wie das Source-NAT. Während ein Source-NAT heute allgemein üblich und allen Administratoren bekannt ist, sind das Destination-NAT und die damit verbundenen Möglichkeiten häufig unbekannt. Das Destination-NAT wird häufig auch als *Port-Forwarding* bezeichnet.

Bei einem Destination-NAT wird die Zieladresse eines Pakets modifiziert. Damit können Sie ein Paket, das eigentlich an die Firewall gerichtet war, auf einen anderen Rechner weiterleiten. So können Sie zum Beispiel sämtliche Anfragen auf dem TCP-Port 80 Ihrer Firewall nach innen auf einen Webserver weiterleiten. Der Client im Internet merkt nicht, dass er in Wirklichkeit nicht mit der Firewall, sondern mit dem privaten Webserver spricht. So können Sie Dienste, die Sie nicht direkt auf der Firewall anbieten wollen, dennoch von außen erreichbar anbieten!

Sie können ein Destination-NAT in der `PREROUTING`- und in der `OUTPUT`-Kette durchführen. In der `PREROUTING`-Kette betrifft das DNAT sämtliche Pakete, die den Rechner von außen erreichen und entweder an ihn lokal gerichtet sind oder weitergeleitet werden müssen. In der `OUTPUT`-Kette betrifft das DNAT nur die lokal erzeugten Pakete.

20.4 MASQUERADE

Häufig möchten Sie ein SNAT durchführen, aber kennen bei dem Schreiben des Firewall-Skripts noch nicht die IP-Adresse, die Sie später verwenden möchten, da diese sich vielleicht auch dynamisch ändert. Das ist zum Beispiel bei den meisten Internet Service Providern (ISPs) der Fall, die Ihnen die Einwahl per ISDN oder ADSL erlauben. Die IP-Adresse, die der ISP

1 Bei IPsec behilft man sich inzwischen mit NAT-Traversal. Hier werden die ESP-Pakete wieder in UDP-Pakete eingepackt. Damit stehen wieder Ports zur Verfügung.

Ihnen zuweist, ändert sich von Einwahl zu Einwahl. Viele Provider führen auch nach einer bestimmten Zeit eine Zwangstrennung durch, um Sie so zu einer neuen Einwahl und einer neuen IP-Adresse zu zwingen. Da Sie jedoch bei dem Target SNAT die IP-Adresse fest definieren müssen, ist dieses Target für diese Zwecke nicht geeignet. Hier können Sie das Target MASQUERADE verwenden. Dieses Target verwendet für das SNAT immer die gerade auf der entsprechenden Schnittstelle aktive Adresse. Dazu prüft das MASQUERADE-Target, über welche Netzwerkkarte das Paket den Rechner verlässt, und verwendet deren Adresse für das NAT.

INFO

Wenn Sie die Netzwerkkarte herunterfahren, werden automatisch alle Verbindungen verworfen, die über diese Netzwerkkarte maskiert wurden. Das ist sinnvoll, da bei einer neuen Einwahl eine neue IP-Adresse zu verwenden ist. Daher sollten Sie, wenn Sie eine statische IP-Adresse verwenden, immer SNAT dem MASQUERADE vorziehen, auch wenn die Regel ein wenig umständlicher in der Definition ist.

Das MASQUERADE-Target hat eine einzige Option, mit der Sie die zu verwendenden Ports für das NAT vorgeben können. Dann weicht der Kernel von seiner üblichen Portwahl (siehe oben) ab und verwendet nur Ports in dem vorgegebenen Bereich (`--to-ports port[-port]`). Dann müssen Sie in der Regel aber auch das Protokoll spezifizieren, auf das sich der Portbereich bezieht (z. B. `-p tcp`). Natürlich kann keine weitere Verbindung maskiert werden, sobald die Ports aufgebraucht wurden. Das MASQUERADE-Target kann nur in der POSTROUTING-Kette verwendet werden.

20.5 NETMAP

Dieses Target erleichtert Ihnen das NAT von ganzen Netzwerken 1:1. Stellen Sie sich vor, dass Sie für den Zugriff auf ein anderes Netzwerk (zum Beispiel auch das Internet) jede IP-Adresse in Ihrem Netz durch genau eine IP-Adresse aus einem anderen Netz austauschen möchten. Wenn Sie dies statisch und nachvollziehbar mit dem SNAT-Target lösen möchten, benötigen Sie so viele Regeln, wie Sie IP-Adressen verwenden möchten. Dies können Sie mit dem NETMAP-Target vereinfachen.

Stellen Sie sich vor, dass Sie zwei Netze zusammenlegen möchten. Ihr Netz verwendet die IP-Adressen 192.168.0.0/24. Das entfernte Netz verwendet die IP-Adressen 172.16.0.0/16. Grundsätzlich besteht hier kein Problem, da unterschiedliche Adressbereiche verwendet werden. Nun kennt das entfernte Netz allerdings bereits ein weiteres Netz, in dem dieselben IP-Adressen verwendet werden (siehe Abbildung 20.2). Für den Zugriff müssen Sie daher Ihre IP-Adressen durch andere austauschen. Dies ist mit der folgenden Zeile sehr einfach möglich:

```
iptables -t nat -A POSTROUTING -d 172.16.0.0/16 -j NETMAP --to ↵
        172.17.0.0/24
```

Nun wird bei jedem Zugriff aus Ihrem Netz die Absenderadresse durch eine Adresse aus dem Bereich 172.17.0.0/24 ersetzt. Das entfernte Netz benötigt natürlich eine Route in das 172.17.0.0/24-Netz. Genauso kann aber auch ein Zugriff aus dem entfernten Netz auf Ihr

Hier „NAPT“ nach „NAT“ korrigiert. Korrekt?

Editor's or typesetter's annotations (will be removed before the final TeX run)

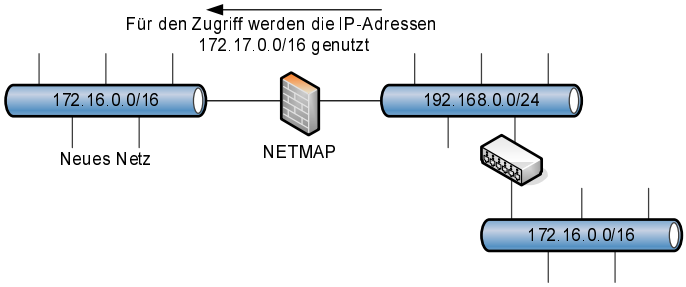


Abbildung 20.2: **NETMAP kann Adresskonflikte lösen.**

Netz erfolgen. Hierfür benötigen Sie dann ein `NETMAP` als Destination-NAT in der `PREROUTING`-Kette.

Sie definieren dies genauso einfach:

```
iptables -t nat -A PREROUTING -d 172.17.0.0/24 -j NETMAP --to ↵
        192.168.0.0/24
```

20.6 SNAT

Dieses Target ist das klassische Source-NAT-Target. Hiermit können Sie die Absender-IP-Adresse durch eine andere ersetzen. Als Ersatz können Sie auch einen Bereich von IP-Adressen angeben. Wenn Sie mehrere IP-Adressen zur Auswahl angeben, führt das `SNAT`-Target ein simples Round-Robin durch.

Um die IP-Adressen anzugeben, verwenden Sie die Option `--to-source <ip>`. Sie können auch einen Bereich angeben: `--to-source <ip-ip>`. Um Tipp-Arbeit zu sparen, können Sie die Option auch mit `--to` abkürzen.

INFO

Bis Kernel-Version 2.6.10 des Iptables-Befehls können Sie auch die Option mehrfach verwenden, um nicht zusammenhängende IP-Adressen für das `SNAT` zu definieren.

Optional können Sie auch noch einen Port-Bereich angeben und damit den Kernel anweisen, bei dem Source-NAT nur Ports aus diesem Bereich zu verwenden.

Die Verwendung von mehreren IP-Adressen zum Source-NAT ist in Umgebungen mit besonders vielen gleichzeitigen Verbindungen sinnvoll. Da der Kernel den Port als Unterscheidungskriterium zwischen den verschiedenen genatteten Verbindungen verwendet und maximal nur 65.536 Ports zur Verfügung stehen, können maximal pro IP-Adresse gleichzeitig nur diese Anzahl an Verbindungen genattet werden. Bei zwei IP-Adressen verdoppelt sich die Anzahl der möglichen Verbindungen bereits.

20.7 SAME

Dieses Target ähnelt dem SNAT- und dem DNAT-Target. Wenn Sie jedoch bei diesen Targets einen Bereich von IP-Adressen für das NAT angeben, führen die Targets ein Round-Robin durch. Das bedeutet, dass neue Verbindungen nacheinander jeweils eine andere IP-Adresse erhalten und dass, sobald alle IP-Adressen aufgebraucht wurden, wieder von vorne begonnen wird. Dieses Verfahren ist jedoch für einige Anwendungen kritisch. Diese Anwendungen verlangen, dass jede Verbindung eines Clients auf dieselbe IP-Adresse genattet wird.

Dies können Sie mit dem SAME-Target erreichen. Sie können dieses Target sowohl in der PREROUTING- als auch in der POSTROUTING-Kette anwenden, je nachdem, ob Sie ein Destination- oder ein Source-NAT machen möchten. Die Verwendung ist sehr einfach. Zwei Beispiele:

```
iptables -t nat -A POSTROUTING -j SAME --to 1.1.1.1-1.1.1.5 --nodst
iptables -t nat -A PREROUTING -j SAME --to 1.1.1.1-1.1.1.5
```

Wird das Target SAME in der POSTROUTING-Kette für das Source-NAT eingesetzt, so wählt der Kernel nur dann dieselbe IP-Adresse, wenn der Client auch auf denselben Server zugreift. Für den Zugriff auf einen anderen Server darf der Kernel eine andere IP-Adresse wählen. Die Option --nodst sorgt bei der Wahl dafür, dass die Ziel-IP-Adresse unberücksichtigt bleibt. Der Client erhält bei dem Source-NAT für alle Zugriffe auf alle Server immer dieselbe IP-Adresse.

20.8 DNAT

Dies ist das klassische Destination-NAT-Target. Hiermit können Sie die Ziel-IP-Adresse eines Pakets und auch den Zielport des Pakets ändern. Diese Funktion wird häufig für ein Port-Forwarding in ein anderes Netzwerk (zum Beispiel eine DMZ) verwendet. Dieses Target ist nur in der PREROUTING- und in der OUTPUT-Kette der NAT-Tabelle gültig.

Die neue Ziel-IP-Adresse geben Sie mit der Option --to-destination <ip> an. Sie können genauso wie bei SNAT auch einen Bereich angeben (--to-destination <ip-ip>) und die Option auch einfach als --to abkürzen. Bis Kernel-Version 2.6.10 können Sie die Option auch mehrfach angeben.

Wenn Sie keinen Port oder Port-Bereich angeben, wird bei dem Destination-NAT der Port nie modifiziert. Sie können aber auch einen Port oder Port-Bereich definieren, der für das NAT verwendet wird. Dann müssen Sie aber ebenfalls das Protokoll angeben:

```
iptables -t nat -A PREROUTING -p tcp --dport 8080 -j DNAT --to ↵
        172.16.0.5:3128
```

Wenn Sie einen Bereich von IP-Adressen definieren, führt auch DNAT wie SNAT ein Round-Robin durch.

20.9 REDIRECT

Dieses Target führt ein spezielles Destination-NAT durch. Daher ist es auch nur in der PREROUTING- und OUTPUT-Kette erlaubt. Es leitet die Pakete an eine lokale IP-Adresse um. Dabei verwendet das Target die lokale IP-Adresse der Netzwerkkarte, über die das Paket die Firewall erreicht hat. Lokal erzeugte Pakete werden an 127.0.0.1 umgeleitet. Diese Funktion kann für einen semi-transparenten Proxy verwendet werden. Der Client versucht, auf einen Server im Internet zuzugreifen. Die Firewall fängt die Verbindung ab und leitet sie an einen lokalen Proxy weiter, der anstelle des Clients die Verbindung aufbaut. Der Client bemerkt die Umleitung nicht. Es handelt sich jedoch nur um einen semi-transparenten Proxy, da der Proxy die Verbindung zum Server nicht mit der IP-Adresse des Clients, sondern mit seiner eigenen IP-Adresse aufbaut. Um alle HTTP-Anfragen an einen transparenten Squid-Proxy weiterzuleiten, können Sie die folgende Iptables-Befehlszeile verwenden:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0 -j REDIRECT --to- ↵
        ports 3128
```

Wenn Sie bei dem REDIRECT-Target keinen Port angeben, wird der Port nicht modifiziert.

Sie können Squid als semi-transparenten Proxy einsetzen. Hierfür müssen Sie bei Squid in den Versionen vor 2.6 die folgenden Parameter in der Konfigurationsdatei setzen:

```
http_port 3128
httpd_accel_host
virtual httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Ab der Version 2.6 vereinfacht sich die Konfiguration. Hier ist nur noch der folgende Parameter erforderlich:

```
http_port 3128 transparent
```

20.10 NAT-Helfermodule

Viele Protokolle sind so kompliziert, dass ein bloßer Austausch der Absender- oder Ziel-IP-Adresse für die Funktion des Protokolls nicht ausreicht. Dabei handelt es sich häufig um Protokolle wie FTP, die weitere Verbindungen dynamisch öffnen und schließen. Weitere Protokolle, die dieses tun, sind: Point-to-Point Tunneling Protocol (PPTP), Internet Relay Chat (IRC), Advanced Maryland Automatic Network Disc Archiver (Amanda), Trivial FTP (TFTP), DirectX8, CuSeeMe, H.323, Microsoft Streaming Media Services (MMS) etc.

Weitere Probleme können auftreten, wenn die IP-Adresse zwar im IP-Header vom NAT getauscht wird, aber zusätzlich noch in dem Paket auftaucht. Auch hier gibt es einige Protokolle, bei denen das der Fall ist. Ein klassischer Vertreter ist das Simple Network Management Protocol (SNMP).

Damit diese Protokolle richtig gehandhabt werden, müssen Sie zusätzliche Kernelmodule laden, die nicht nur die IP-Header der Pakete betrachten, sondern auch den Inhalt analysieren und dort vorhandene IP-Adressen austauschen beziehungsweise dynamisch ausgehandelte neue Verbindungen erkennen und als erwartete Verbindung (Expectation) in die Tabelle eintragen. Diese können Sie dann mit dem Zustand `RELATED` erlauben.

Das Laden dieser Module ist sehr einfach und erfolgt sinnvollerweise zu Beginn Ihres Firewall-Skripts:

```
modprobe nf_nat_ftp
```

Wenn die Module Abhängigkeiten besitzen, erkennt der `modprobe`-Befehl diese und lädt automatisch auch die weiteren benötigten Module. Bei einigen Modulen können Sie zum Lade-Zeitpunkt weitere Optionen angeben, mit denen Sie das Verhalten modifizieren können (zum Beispiel FTP, siehe Abschnitt 32.10). Die Informationen über die möglichen Optionen erhalten Sie am einfachsten mit dem `modinfo`-Befehl.

20.11 Das CONNMARK-Target

Mit diesem Ziel können Sie Verbindungen markieren. Diese Markierung unterscheidet sich von der Paket-Markierung mit `MARK`. Dieses Ziel unterstützt in der NAT-Tabelle die beiden folgenden Optionen:

» `--set-xmark <markierung>[<maske>]`: Hier wird die vorhandene Markierung mit dem Wert mit einer XOR-Operation verknüpft. Gleichzeitig können Sie die folgenden Optionen mitangeben:

- `--and-mark`:
- `--or-mark`:
- `--xor-mark`:

» `--set-mark <markierung>[/<maske>]`: Hiermit setzen Sie die Markierung. Wenn Sie eine Maske angeben, werden nur diese Bits modifiziert.

» `--save-mark [--nfmask <maske>] [--ctmask <maske>]`: Hiermit übertragen Sie eine Paketmarkierung (siehe Abschnitt 21.7) auf die Verbindung. Diese können Sie in der Mangle-Tabelle mit dem `CONNMARK`-Ziel wiederherstellen.

21. Die Mangle-Tabelle

Die Mangle-Tabelle ist der Ort, wo Sie spezielle Paket-Modifikationen vornehmen können. Sie können hier die Type-of-Service-Bits ändern, Pakete markieren, TCP-MSS-Werte setzen oder das ECN-Bit abschalten.

21.1 Die Ketten der mangle-Tabelle

Je nach Ihrem Kernel weist die Mangle-Tabelle zwei oder fünf Ketten auf. Bis einschließlich der Kernel-Version 2.4.17 gab es nur die beiden Ketten `PREROUTING` und `OUTPUT` in der Mangle-Tabelle. Anschließend kamen auch noch die Ketten `INPUT`, `FORWARD` und `POSTROUTING` hinzu. Diese Ketten werden immer vor ihren entsprechenden Brüdern in den Tabellen NAT und Filter durchlaufen. Das bedeutet, dass Sie die Pakete modifizieren können, bevor die NAT-Regeln und die Filterregeln das Paket verarbeiten.

21.2 Aktionen der Mangle-Tabelle

Es gibt einige Aktionen (Targets), die nur in der Mangle-Tabelle erlaubt sind. Hierbei handelt es sich um: `CLASSIFY`, `CONNMARK`, `DSCP`,`ECN`, `MARK`, `TOS`, `TTL` und `XOR`. Dabei ist das Target `CONNMARK` ein Sonderfall, da es sowohl in der NAT- als auch in der Mangle-Tabelle mit unterschiedlichen Optionen genutzt werden kann.

Diese Targets möchte ich nun in diesem Kapitel vorstellen. Natürlich können Sie auch `ACCEPT`, `DROP`, `REJECT` etc. in der Mangle-Tabelle nutzen. Diese Targets habe ich aber bereits an anderer Stelle vorgestellt.

21.3 CLASSIFY

Linux unterstützt Class-based Queueing (CBQ). Mit diesem Ziel können Sie direkt ohne Firewall-Markierung ein Paket einer bestimmten Klasse zuweisen. Hierzu geben Sie mit der zusätzlichen Option `--set-class` die Major:Minor-Klasse an:

```
$IPTABLES -t mangle -A POSTROUTING -p tcp --dport 22 -j CLASSIFY --set- ↵
        class 1:10
```

Sie müssen mit dem Kommando `tc` natürlich auch eine entsprechende Klasse angelegt haben.

21.4 CONNMARK

Mit diesem Target können Sie eine Verbindung markieren. Dies ist nicht dasselbe wie die Markierung eines Pakets mit dem `MARK`-Target (siehe Abschnitt 21.7). Wenn Sie dennoch die Markierung auf das Paket übertragen möchten, können Sie das mit der Option `--restore-mark` des `CONNMARK`-Targets erreichen. Diese Option dürfen Sie nur in der Mangle-Tabelle verwenden.

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j CONNMARK --set-mark 80
iptables -t mangle -A FORWARD -m connmark --mark 80 -j CONNMARK --restore ↵
        -mark
```

21.5 DSCP

Mit diesem Target können Sie die Diffserv-Class-Point-Bits im ToS-Header eines IP-Pakets setzen. Dieses DSCP-Feld wird in RFC2474 beschrieben. Diese Werte werden häufig von Routern und Switches verwendet, um die Priorität von Paketen festzulegen. Hierfür stehen Ihnen zwei Optionen zur Verfügung:

» `--set-dscp <wert>`: Hiermit können Sie das DSCP-Feld mit einem numerischen Wert füllen.

» `--set-dscp-class <klasse>`: Hiermit können Sie eine DiffServ-Klasse auswählen.

21.6 ECN

Die Explicit Congestion Notification (ECN) ist eine neue Methode, die in dem RFC3168 beschrieben wurde. Hiermit können zwei Kommunikationspartner, wenn beide ECN unterstützen, mögliche drohende Verstopfungen der Verbindung vor deren Auftreten erkennen und durch eine Reduktion der Sendeleistung reagieren. So ist die Wahrscheinlichkeit sehr groß, dass die Verstopfung erst gar nicht auftritt (siehe auch Abschnitt A.15.3).

Leider verwerfen viele alte und auch noch einige moderne Firewalls Pakete, die ECN verwenden, da die hier verwendeten Bits im IP- und TCP-Header bis zur Definition des RFC 3168 reserviert waren und lediglich von Werkzeugen wie `nmap` verwendet wurden. Wenn Sie dennoch ECN verwenden möchten, stellen Sie wahrscheinlich fest, dass die Kommunikation mit einigen Zielen nicht funktioniert. Dann können Sie für diese Ziele die ECN-Bits entfernen lassen:

```
iptables -t mangle -A PREROUTING -p tcp -d $ECN_BLACKHOLE -j ECN --ecn- ↵
        tcp-remove
```

21.7 MARK

Hiermit können Sie eine Netfilter-Markierung des Pakets vornehmen. Diese Firewall-Markierung können Sie später mit dem Test `mark --mark` wieder prüfen oder auch in Zusammenhang mit dem iproute2-Paket nutzen. Um zum Beispiel für bestimmte Pakete eine andere Routing-Tabelle zu nutzen, können Sie die folgenden Befehle nutzen:

```
iptables -t mangle -A PREROUTING -p tcp --dport 25 -j MARK --set-mark 25
ip route add table 25 default via 192.168.0.5
ip rule add fwmark 25 table 25
```

Hiermit markieren Sie alle TCP-Pakete an den Port 25 mit der entsprechenden Markierung. Anschließend erzeugen Sie eine zusätzliche Routing-Tabelle mit der Nummer 25, in der Sie ein weiteres Default-Gateway eintragen. Damit diese Routing-Tabelle auch genutzt wird, definieren Sie eine Regel, die alle Pakete mit der Markierung 25 über diese Tabelle routet.

21.8 TOS

Hiermit können Sie die Type-of-Service-Bits im IP-Header modifizieren. Einige Betriebssysteme, wie Linux, werten diese Bits aus. Hierfür verwenden Sie `-j TOS --set-tos <wert>`. Die zur Verfügung stehendenden Werte zeigt Ihnen der Befehl `Iptables -j TOS -h` an:

```
TOS target v1.3.0 options:
  --set-tos value             Set                 Type of Service field  ↵
          to one of the following numeric or descriptive values:
                              Minimize-Delay      16 (0x10)
                              Maximize-Throughput  8 (0x08)
                              Maximize-Reliability 4 (0x04)
                              Minimize-Cost        2 (0x02)
                              Normal-Service       0 (0x00)
```

21.9 TPROXY

Dieses Target (siehe auch Kapitel 28) erlaubt den Aufbau echt transparenter Proxies. Dabei kann der Proxy die Verbindung zum echten Server mit der Absender-IP-Adresse des echten Clients aufbauen. Hierzu benötigen Sie dann speziell angepasste Proxies wie bei der Zorp-Firewall, die von der Firma Balabit entwickelt wird (`http://www.balabit.com`). Diese Firma hat auch die Entwicklung des `TPROXY`-Targets vorangetrieben.

Das Target leitet das Paket an einen lokalen Socket um, ohne den Paket-Header zu verändern. Dadurch kann der Proxy die originale Ziel- und Quell-IP-Adresse auslesen. Das Target unterstützt drei Optionen:

» `--on-port port`: Hiermit wird das Paket an diesen lokalen Port weitergeleitet.

» `--on-ip ip`: Hiermit geben Sie die Ziel-IP-Adresse an. Dies ist per Default die IP-Adresse der eingehenden Netzwerkkarte.

» `--tproxy-mark value[/mask]`: Hiermit werden die Pakete zusätzlich markiert.

Dieses Ziel ist nur in der `PREROUTING`-Kette erlaubt.

21.10 TTL

Dieses Target ermöglicht es Ihnen, den TTL-Wert eines IP-Pakets zu modifizieren. Das Netfilter-Team rät stark von der Verwendung dieses Targets ab. Bei falscher Anwendung erzeugen Sie hiermit Endlosschleifen!

INFO

Mit diesem Target können Sie auf Ihrer Firewall den TTL-Wert jedes Pakets, das die Firewall passiert, um eins erhöhen. Wenn Ihre Firewall selbst nicht auf den `traceroute`-Befehl reagiert, aber die Time-Exceeded-Mitteilungen durchlässt, ist die Firewall scheinbar für Traceroute unsichtbar, da für jeden TTL-Wert eine Antwort zurückkommt. Die Reduktion des TTL-Wertes durch die Firewall, die zum Ausfall dieser Antwort führen würde, wird von Iptables durch die Erhöhung wieder rückgängig gemacht.

Das Target hat drei Optionen:

» `--ttl-set <wert>`: Hiermit setzen Sie fest einen neuen Wert fest.

» `--ttl-dec <wert>`: Hiermit ziehen Sie den Wert von dem aktuellen TTL-Wert ab.

» `--ttl-inc <wert>`: Hiermit addieren Sie die angegebene Zahl zum aktuellen TTL-Wert hinzu.

22. Die Raw-Tabelle

Die raw-Tabelle wurde eingeführt, um Pakete bereits behandeln zu können, bevor sie von dem Connection Tracking erfasst werden. Die wichtigste Idee ist hierbei die Umgehung des Connection Tracking für bestimmte Pakete. Dies ist erstmalig mit der Raw-Tabelle möglich.

22.1 Die Raw-Tabelle

Die Raw-Tabelle besitzt zwei Ketten: PREROUTING und OUTPUT. Außerdem existieren zwei Targets, die nur in der Raw-Tabelle verwendet werden dürfen: NOTRACK und TRACE.

Mit dem Erscheinen der Raw-Tabelle existiert nun auch ein neuer Zustand, der mit dem state-Test geprüft werden kann: UNTRACKED.

Die Raw-Tabelle bietet Ihnen den ersten Zugriff auf die ankommenden (PREROUTING) und die lokal erzeugten (OUTPUT) Pakete. Neben den beiden Spezial-Targets NOTRACK und TRACE können Sie natürlich auch die eingebauten Targets DROP und ACCEPT verwenden. Außerdem verfügen die Ketten natürlich auch über Default-Policies, die Sie mit dem Befehl `iptables -t raw -P PREROUTING DROP` setzen können.

Jedoch ist die Tabelle in erster Linie dazu gedacht, dass Sie Ausnahmen von der Verbindungsüberwachung definieren können. Damit können Sie bestimmte Pakete, bei denen eine Verbindungsüberwachung sinnlos oder überflüssig ist, von dieser ausnehmen.

So kann es ratsam sein, Protokolle auszuklammern, die Sie unbedingt annehmen möchten und bei denen eine Zustandsüberwachung nicht sinnvoll ist (zum Beispiel IPsec-ESP).

```
iptables -t raw -A PREROUTING -p 50 -j NOTRACK
iptables -A FORWARD -p 50 -j ACCEPT
```

Die Möglichkeit der Paketverfolgung ist mit dem TRACE-Target gegeben. Damit können Sie zu Testzwecken genau den Weg eines Pakets durch Ihr Regelwerk und durch alle Tabellen und Ketten genau verfolgen.

23. Das /proc-Dateisystem

Linux besitzt einen sehr mächtigen TCP/IP-Stack und Paketfilter. Das Verhalten dieses Stacks und des Paketfilters ist sehr flexibel. Wenn Sie die Flexibilität kennenlernen möchten und den Stack oder den Paketfilter anpassen und modifizieren möchten, müssen Sie sich mit dem `/proc`-Dateisystem beschäftigen. Dieses virtuelle Dateisystem gibt Ihnen lesenden und schreibenden Zugriff auf wesentliche Variablen des Betriebssystems. Hierzu gehören auch der Netzwerkstapel und der Paketfilter.

23.1 Einführung in /proc

Das `/proc`-Dateisystem ist ein virtuelles Dateisystem, in dem der Kernel viele Variablen des Betriebssystems für den lesenden und häufig auch schreibenden Zugriff anbietet. Wesentliche Informationen des Betriebssystems können (nur) über diese Struktur ausgelesen werden.

Wenn Sie sich den Inhalt des `/proc`-Verzeichnisses ansehen, finden Sie sowohl Verzeichnisse als auch Dateien vor (siehe Abbildung 23.1).

Ein großer Teil der Verzeichnisse stellt Informationen über die laufenden Prozesse zur Verfügung. Jedes Verzeichnis mit einer Zahl als Verzeichnisnamen enthält Informationen über den Prozess mit der entsprechenden Prozess-ID. Die weiteren Einträge und Verzeichnisse enthalten Informationen über das laufende Betriebssystem und erlauben teilweise auch die

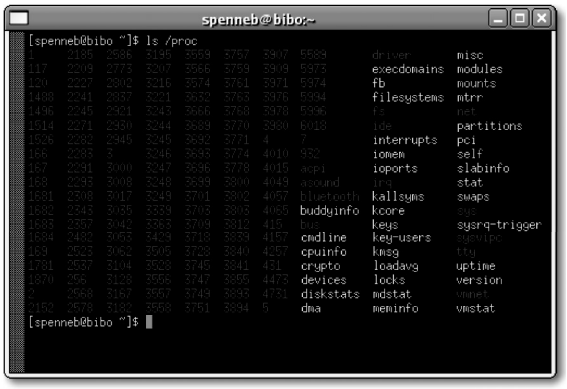


Abbildung 23.1: **Das /proc-Dateisystem enthält sowohl Informationen über Prozesse als auch über das laufende Betriebssystem.**

Konfiguration dieser Parameter. So enthält zum Beispiel der Eintrag `/proc/cpuinfo` Informationen über den von dem Betriebssystem gefundenen Prozessor. In der Datei `/proc/partitions` finden Sie Informationen über die von dem Betriebssystem erkannten Partitionen:

```
$ cat /proc/partitions
major minor #blocks  name

   3      0 78150744 hda
   3      1   104391 hda1
   3      2 78043770 hda2
  22      0 78150744 hdc
  22      1 78150712 hdc1
 253      0 75956224 dm-0
 253      1  2031616 dm-1
 253      2 20971520 dm-2
 253      3 20971520 dm-3
 253      4 20971520 dm-4
$ cat /proc/cmdline
ro root=/dev/VolGroup00/LogVol00
```

Die Datei `/proc/cmdline` enthält Informationen über die Kommandozeile, mit der der laufende Kernel aufgerufen wurde.

Teilweise können Sie die Werte in den (virtuellen) Dateien auch verändern. So definiert die Datei `/proc/sys/net/ipv4/icmp_echo_ignore_all`, ob der Kernel auf ICMP-Echo-Pakete reagiert oder diese ignoriert. Diese Datei kann den Wert 0 oder 1 enthalten. Ein Wert von 0 bedeutet, dass diese Funktion nicht aktiv ist. Ein Wert von 1 bedeutet in diesem Fall, dass der Kernel diese Pakete ignoriert.

Sie können das selbst sehr einfach prüfen und nachvollziehen. Prüfen Sie zunächst, ob die Datei für den Benutzer *root* über Schreibrechte verfügt, und zeigen Sie den aktuellen Wert der Datei an:

```
$ ls -l /proc/sys/net/ipv4/icmp_echo_ignore_all
-rw-r--r-- 1 root root 0 21. Jun 17:23
/proc/sys/net/ipv4/icmp_echo_ignore_all
$ cat /proc/sys/net/ipv4/icmp_echo_ignore_all
0
```

Starten Sie nun in einem eigenen Fenster den `ping`-Befehl (z. B.: `ping 127.0.0.1`). Sie sollten erfolgreich Ihren eigenen Rechner pingen können. Falls es nicht funktioniert, prüfen Sie bitte, ob eine Firewall aktiv ist, und deaktivieren Sie diese im Zweifelsfall.

Während der Ping läuft, ändern Sie nun als *root* den Inhalt der Datei:

```
$ echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all
```


CE: Auch etwas kurz. Vielleicht an anderem Ort einbinbar?

Nun sollte der Ping sofort aufhören. Sobald Sie die 0 wieder in die Datei schreiben, läuft der Ping weiter. Anstelle des `echo`-Kommandos können Sie auch den Befehl `sysctl` verwenden. Dieser Befehl erwartet den Namen der Variablen etwas anders:

```
$ /sbin/sysctl net.ipv4.icmp_echo_ignore_all
net.ipv4.icmp_echo_ignore_all = 1
```

Entfernen Sie `/proc/sys/` von dem Pfad der Datei, und ersetzen Sie alle weiteren `/` (Schrägstriche) durch einen Punkt. Allerdings kann der Befehl auch die Schrägstriche statt Punkte verwenden:

```
$ /sbin/sysctl net/ipv4/icmp_echo_ignore_all
net.ipv4.icmp_echo_ignore_all = 0
```

Wenn Sie den Befehl nur mit dem Namen der Variablen verwenden, zeigt er den aktuellen Wert an. Wenn Sie den Wert ändern möchten, verwenden Sie zusätzlich die Option `-w` und geben den neuen Wert an:

```
$ /sbin/sysctl -w net/ipv4/icmp_echo_ignore_all=1
net.ipv4.icmp_echo_ignore_all = 1
```

Alle so durchgeführten Änderungen sind nach einem Neustart verloren und müssen neu gesetzt werden. Viele Distributionen verwenden hierzu inzwischen die Datei `/etc/sysctl.conf`. Sie können die Änderungen, die von der Distribution automatisch bei dem Systemstart durchgeführt werden sollen, in dieser Datei angeben:

```
# Kernel sysctl configuration file for Red Hat Linux For binary values, 0 ↵
         is disabled, 1 is enabled.
# See sysctl(8) and sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Do not accept source routing
net.ipv4.conf.default.accept_source_route = 0
```

Diese Datei wird bei dem Start mit dem Befehl `sysctl -p /etc/sysctl.conf` eingelesen.

Im Folgenden werde ich die wichtigsten Variablen vorstellen, die für die Konfiguration des Paketfilters und TCP/IP-Stacks interessant sind.

23.2 /proc/net/

Alle Einträge in diesem Verzeichnis sind lediglich lesbar. Ein Schreibzugriff ist nicht vorgesehen. Die Dateien dienen lediglich informativen Zwecken.

23.2.1 ip_conntrack oder nf_conntrack

Dies ist die Connection-Tracking-Tabelle. Alle bekannten Verbindungen werden in diese Tabelle aufgenommen, sobald das Connection-Tracking-Modul geladen wurde. Der Name des Moduls lautet auf älteren Kerneln `ip_conntrack.ko`, während moderne Kernel das `nf_conntrack.ko`-Modul verwenden. Häufig ist diese Funktion aber in den Kernel eingebaut.

```
tcp  6 431842 ESTABLISHED src=192.168.0.108 dst=217.160.128.61 sport ↵
        =56468 dport=993 packets=79
bytes=5788 src=217.160.128.61 dst=192.168.0.108 sport=993 dport=56468  ↵
        packets=51 bytes=5864
[ASSURED] mark=0 use=1
```

Ab Kernelversion `2.6.10` und bei entsprechender Konfiguration (`CONFIG_IP_NF_CT_ACCT=y`) werden auch die Paket- und Byte-Zähler für die Verbindung angezeigt.

23.2.2 nf_conntrack_expect

Ab Kernel 2.6.9 werden die Expectations in dieser Datei angezeigt. Expectations sind Verbindungen, die von einem Conntrack-Helper-Modul erkannt werden und als `RELATED`-Verbindungen erlaubt werden. Auf älteren Kerneln heißt auch diese Datei `ip_conntrack_expect`.

23.2.3 ip_tables_*

Die drei Dateien `ip_tables_matches`, `ip_tables_targets` und `ip_tables_names` enthalten die Namen der aktuell geladenen Matches (Tests), Targets (Ziele) und Tables (Tabellen). Sobald eine Regel einen neuen Match oder ein neues Target benötigt, das durch ein Modul zur Verfügung gestellt wird, wird das Modul geladen, und dessen Name taucht in diesen Dateien auf.

23.3 /proc/sys/net/ipv4

Dieses Verzeichnis enthält Variablen, auf die Sie sowohl lesend als auch schreibend zugreifen können. In vielen Fällen sollten Sie sich jedoch sehr genau überlegen, ob es sinnvoll ist, die entsprechenen Parameter zu verändern, da dies durchaus Auswirkungen (positive wie negative) auf die Leistungsfähigkeit des Systems haben kann.

23.3.1 icmp_*

Diese Variablen betreffen das ICMP-Protokoll. Falls Sie mehr oder weniger Optionen besitzen, so hängt dies von der verwendeten Kernelversion ab.

icmp_echo_ignore_all

Diese Variable entscheidet, ob der Kernel alle ICMP-Echo-Pakete (Ping) ignorieren soll. Schreiben Sie in diese Datei eine 1, und Ihr System wird auf ein Ping grundsätzlich nicht mehr reagieren.

icmp_echo_ignore_broadcasts

Wenn Sie diese Variable auf den Wert 1 setzen, ignoriert der Kernel alle Echo-Request-Pakete (Ping) an die Broadcast-Adresse. Üblicherweise beantworten alle Linux/UNIX-Betriebssysteme einen Echo-Request an ihre Broadcast-Adresse. So können Sie sehr leicht in einem Netzwerk alle verfügbaren UNIX-Systeme ermitteln. Da aber viele Router in der Vergangenheit (und auch heute noch) derartige Broadcast-Ping-Anfragen nicht verworfen haben, konnte damit eine Verstärkung des Verkehrs für einen Angriff erzeugt werden. Sie senden ein Ping an die Broadcast-Adresse eines Netzwerks mit 10 UNIX-Rechnern und erhalten 10 Antworten zurück. Wenn Sie die Absenderadresse fälschen (IP-Spoofing), können Sie so einen anderen Rechner mit Ping-Antworten überschwemmen. Dieser Angriff trägt den Namen Smurf-Angriff. Die Netzwerke werden als Smurf Amplifier Networks bezeichnet. Auch heute gibt es noch derartige Netzwerke, deren Router die Broadcast-Pakete nicht verwerfen (`http://smurf.powertech.no/`). Um zu verhindern, dass Ihr System für diesen Angriff genutzt wird, sollten Sie die Beantwortung von Broadcast-Ping-Anfragen abschalten. Auf vielen modernen Distributionen ist dies heute der Default.

```
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
```

icmp_errors_use_inbound_ifaddr

Normalerweise versendet ein Linux-System eine Fehlermeldung mit der Absenderadresse der Netzwerkkarte, über die die Fehlermeldung das System verlässt. Dies ist nicht zwingend die Netzwerkkarte, auf der die Fehlermeldung aufgetreten ist. Wird diese Option (ab der Kernel-Version 2.6.12) gesetzt, so wird die Fehlermeldung mit der Absenderadresse der Netzwerkkarte gesendet, auf der der Fehler auftrat. Da dies die Fehlersuche stark vereinfacht, sollte diese Variable aktiviert werden.

icmp_ignore_bogus_error_messages

Einige Router verletzen den RFC1122 (Requirements for Internet Hosts – Communication Layers) und versenden Fehlermeldungen als Antwort auf Broadcast-Mitteilungen. Normalerweise protokolliert der Kernel diese Tatsache. Wenn Sie diese Variable aktivieren, protokolliert der Kernel dies nicht mehr und kann so Ihre Protokolldateien nicht mit unnötigen Meldungen füllen. Ich empfehle die Aktivierung dieser Option.

icmp_ratelimit

Um Denial-of-Service-Angriffe mit ICMP-Nachrichten zu verhindern und allgemein den Netzwerkverkehr zu reduzieren, bieten diese und die nächste Variable Ihnen die Möglichkeit, die

Versendung von ICMP-Meldungen zu beschränken. In dieser Variable definieren Sie, wie viel Jiffies der Kernel zwischen zwei bestimmten ICMP-Meldungen an einen bestimmten Rechner warten soll. Ein Jiffie entspricht auf der i386-Plattform 1/100 Sekunde. Der Default-Wert dieser Variable ist 100. Das bedeutet, dass maximal einmal pro Sekunde eine bestimmte IMCP-Nachricht an denselben Rechner versendet wird. Ein Wert von 0 schaltet dieses Verhalten ab.

icmp_ratemask

Diese Variable definiert, welche ICMP-Nachrichten das Rate-Limiting betrifft. Hierzu bedient sich diese Variable einer Bitmaske:

```
Bits:    IHGFEDCBA9876543210
Default: 0000001100000011000 (6168)

0 Echo Reply
1 Nicht definiert
2 Nicht definiert
3 Destination Unreachable *
4 Source Quench *
5 Redirect
8 Echo Request
B Time Exceeded *
C Parameter Problem *
D Timestamp Request
E Timestamp Reply
F Info Request
G Info Reply
H Address Mask Request
I Address Mask Reply
```

23.3.2 igmp_*

Das Internet-Group-Management-Protokoll (IGMP) wird für die Verwaltung von Multicast-Gruppen verwendet. Der Kernel verfügt über zwei Variablen, die das Verhalten dieses Protokolls beeinflussen.

igmp_max_memberships

Diese Variable definiert die maximale Anzahl an Multicast-Gruppen, bei denen das Linux-System Mitglied werden kann. Üblicherweise müssen Sie diesen Wert nicht modifizieren.

igmp_max_msf

Diese Datei definiert die maximale Anzahl der Multicast-Source-Filter (MSF). Das IGMPv3-Protokoll erlaubt es einer Applikation, Filter auf Multicast-Gruppen zu definieren, mit denen

die Absender der Multicast-Nachrichten eingeschränkt werden können. Dann akzeptiert der Linux-Kernel diese Nachrichten nur noch von wenigen Systemen. Diese Variable definiert, wie viele Multicast-Source-Filter gesetzt werden dürfen.

23.3.3 inet_peer_*

Ein Inet-Peer ist ein anderer Rechner im Internet, mit dem unser System im Moment kommuniziert. Für jeden derartigen Peer speichert unser System langlebige Informationen in einer eigenen Struktur. Aktuell handelt es sich hierbei nur um die IP-Identifikationsnummer (ID) des nächsten Pakets. Diese Nummer wird benötigt, um bei der Defragmentierung der Pakete die richtigen Fragmente eines Pakets zu erkennen. Diese weisen alle dieselbe IP-ID auf. Diese Variablen definieren, wie diese Strukturen verwaltet werden.

Da ein Linux-System per Default Path-MTU-Discovery verwendet und so eine Fragmentierung des Pakets ausschließen kann, benötigen die versandten Pakete keine IP-ID. Daher werden in diesem Fall der entsprechende Code und diese Variablen nicht verwendet.

Da diese Variablen in den meisten Umgebungen nicht genutzt werden, werden sie hier nicht weiter erläutert. Sie finden eine aktuelle Dokumentation in dem Quellcode Ihres Kernels unter `Documentation/networking/ip-sysctl.txt`.

23.3.4 ip_*

Diese Variablen beeinflussen das IP-Protokoll.

ip_autoconfig

Der Kernel kann selbst die Protokolle DHCP oder BOOTP verwenden, um eine Autokonfiguration der Netzwerkkarten vorzunehmen. Dazu muss der Kernel entsprechend konfiguriert sein (`IP_PNP=y`).

ip_conntrack_max

Diese Variable definiert auf alten Kerneln die maximal unterstützten Verbindungen in der Connection-Tracking-Tabelle. Der Default-Wert hängt von dem zur Verfügung stehenden Arbeitsspeicher ab. Auf einer Firewall ist es sinnvoll, diesen Wert zu erhöhen. Dann sollten Sie aber auch die Hashsize erhöhen (siehe Abschnitt 19.2).

Moderne Kernel verwenden die Variable `nf_conntrack_max` in dem Verzeichnis `/proc/sys/net/netfilter`.

ip_default_ttl

Hiermit setzen Sie den Time-To-Live-Wert (TTL) der von dem Linux-System versandten Pakete. Alle normalen Pakete erhalten diesen TTL-Wert. Nach der entsprechenden Anzahl von Hops wird das Paket dann verworfen. Jeder Router, der von einem Paket auf dem Weg zum Ziel passiert wird, ist ein Hop. Hiermit wird verhindert, dass eine Routerfehlkonfiguration

zu Paketen führt, die ewig transportiert werden. ICMP-Fehlermeldungen werden von Linux unabhängig von diesem Wert immer mit einem TTL von 255 versendet.

ip_dynaddr

Wenn Sie in Ihrer Firewall dynamische IP-Adressen auf der externen Netzwerkkarte bei gleichzeitigem Masquerading verwenden, sollten Sie diese Variable aktivieren, um fehlerfreie Verbindungen zu ermöglichen. Ansonsten kann es zu Problemen bei der ersten Verbindung kommen, die die Einwahl in das Internet antriggert. Um dies zu verstehen, stellen Sie sich vor, Ihr Firewall-System sei im Moment nicht mit dem Internet verbunden. Ein Client hinter der Firewall möchte auf das Internet zugreifen. Er versendet sein erstes Paket, um die Verbindung aufzubauen. Die Firewall filtert das Paket, maskiert es mit der aktuellen IP-Adresse der externen Netzwerkkarte und übergibt das Paket an die Netzwerkkarte. Dies triggert die Einwahl, und das System baut die Internetverbindung auf. Dabei erhält das System eine neue IP-Adresse auf der externen Netzwerkkarte. Wenn nun das erste Paket unverändert versendet werden würde, hätte es die falsche Absender-IP-Adresse. Diese Variable erzwingt eine erneute Maskierung dieses ersten Pakets.

Wenn Sie eine 1 in die Variable schreiben, aktivieren Sie das Verhalten. Ein Wert größer 1 führt zusätzlich jedes Mal zur Protokollierung.

ip_forward

Diese Variable schaltet für sämtliche Netzwerkkarten des Linux-Systems die Weiterleitung (Forwarding) ein. Wenn diese Variable den Wert 0 enthält, leitet der Linux-Kernel keine Pakete zwischen zwei Netzwerkkarten weiter.

ip_local_port_range

Diese Datei enthält zwei Zahlen, die den Bereich der lokal erlaubten Client-Ports definieren. Wenn Sie diese Variable verändern möchten, müssen Sie auch gleichzeitig beide Zahlen in diese Variable schreiben:

```
echo "4096 7000" > ip\_local\_port_range
```

Die erste angegebene Nummer ist der kleinste Port, der verwendet werden kann, während die zweite Nummer der höchste Port ist. Wählen Sie hier keine Ports kleiner 1024, um Problemen vorzubeugen. Üblich sind hier Ports zwischen 32768 und 61000.

ip_nonlocal_bind

Üblicherweise kann eine Applikation sich nur auf lokal definierte IP-Adressen binden. Wenn diese Variable gesetzt ist, kann eine Applikation auch IP-Adressen verwenden, die auf dem System nicht definiert sind, und mit diesen Adressen als Absenderadressen Pakete versenden und Pakete entgegennehmen. Diese Funktion wird für einen echten transparenten Proxy benötigt, der die Verbindung zum Server mit der IP-Adresse des echten Clients aufbaut.

ip_no_pmtu_disc

Der Linux-Kernel verwendet per Default die Path Maximum Transmission Unit Discovery (PMTU-Discovery, PMTUD). Damit verhindert der Linux-Kernel, dass eine Fragmentierung der Pakete unterwegs auftritt und bei gleichzeitigem Fragmentverlust unnötig viele Pakete wiederholt gesendet werden müssen. In einigen Umgebungen führt dieses Verhalten jedoch zu Problemen. Dann können Sie diese Funktion mit dieser Variablen abschalten (1).

23.3.5 ipfrag_*

Diese Parameter definieren, wie das Linux-System eine Defragmentierung durchführen soll.

ipfrag_*_thresh

Die Variablen `ipfrag_low_thresh` und `ipfrag_high_thresh` definieren den Speicherbereich, der für die Speicherung der Fragmente bei der Defragmentierung genutzt wird. Der genutzte Speicher schwankt zwischen dem `ipfrag_low_thresh`- und `ipfrag_high_thresh`-Wert.

ipfrag_time

Dieser Wert definiert, wie lange die Fragmente für eine Defragmentierung aufbewahrt werden. Ist innerhalb dieser Zeit keine Defragmentierung möglich, sendet der Rechner eine ICMP-Time-Exceeded-Fehlermeldung und verwirft das unvollständige Paket.

ipfrag_secret_interval

Ältere Kernel besitzen eine Sicherheitslücke in der Fragmentierung, die einen Denial-of-Service-Angriff auf die Defragmentierung ermöglicht (CAN-2003-0364). Um dieses zu verhindern, wurde ein Secret eingeführt, sodass der Angreifer die benötigten Pakete nicht mehr vorhersagen kann. Dieses muss zur Sicherheit in regelmäßigen Abständen neu berechnet werden. Das Intervall in Sekunden wird in dieser Variable definiert.

ipfrag_max_dist

Diese neue Variable des Linux-Kernels 2.6.14 TS9 definiert die maximale Unordnung, mit der Fragmente den Kernel erreichen dürfen.

23.3.6 tcp_*

Hier sind sehr viele Variablen vorhanden, die das Verhalten des TCP-Protokolls beeinflussen.

tcp_abc

Diese Datei steuert den „Appropriate Byte Count“ (ABC) nach RFC3465. ABC erlaubt die langsame Vergrößerung des Congestions Windows in Reaktion auf partielle Bestätigungen (ACKs).

TS9 Bitte prüfen: Hier “>” gelöscht.

Editor's or typesetter's annotations (will be removed before the final TEX run)

Mögliche Werte sind:

0: abgeschaltet, Vergrößerung des Windows bei jeder partiellen Bestätigung

1: Vergrößerung des Windows bei jeder Bestätigung eines vollständigen Segments um 1

2: Vergrößerung des Windows um 2 bei gleichzeitiger Bestätigung von zwei Segmenten und verspäteten ACKs

Default: 0

tcp_abort_on_overflow

Diese Variable erzwingt einen Reset neuer Verbindungen, wenn der dazugehörige Dienst die neuen Verbindungen nicht schnell genug entgegennehmen kann. Bevor Sie diese Variable setzen, sollten Sie immer versuchen, den Dienst zu optimieren. Wenn es sich nicht um einen Dauerzustand handelt, sondern nur punktuell Probleme auftreten, kann diese Funktion hilfreich sein.

tcp_adv_win_scale

Die Variable `tcp_rmem` definiert den für den Empfang von Paketen reservierten Speicher eines Sockets. Dieser Speicher wird für das TCP-Window und den Applikationspuffer aufgeteilt. Diese Variable definiert dieses Verhältnis. Ist der Wert der Variable positiv, wird für das TCP-Window $\frac{1}{2^{tcp\_adv\_win\_scale}}$ TS^h genutzt. Bei dem Default-Wert von 2 wird also ein Viertel des Speichers für das TCP-Window reserviert. Bei einem negativen Wert wird $\frac{1-1}{2^{tcp\_adv\_win\_scale}}$ verwendet – bei einem Wert von −2 also drei Viertel des Speichers.

tcp_app_win

Auch diese Variable definiert, wie viel Speicher für den Applikationspuffer eines Sockets maximal reserviert wird. Der Kernel nutzt folgende Formel: $\frac{window}{2^{tcp\_app\_win}}$. Der Wert muss immer mindestens der Maximum Segment Size (MSS) entsprechen.

tcp_available_congestion_control

Diese Variable zeigt die verfügbaren Mechanismen der Congestion Control.

tcp_base_mss

Dies ist der Startwert für die Ermittelung der Path MTU Discovery.

tcp_bic*

Diese Parameter sind auf aktuellen Kerneln meist nicht mehr vorhanden.

Die Binary-Increase-Congestion-(BIC-)Control versucht die Probleme des TCP-Protokolls bei der Übertragung großer Datenmengen über große Entfernungen zu beheben. TCP verschenkt bei großen Entfernungen einen Großteil der Bandbreite. BIC ist eine reine Modifikation des

TS^h Bitte prüfen: Im Nenner öffnende runde Klammer entfernt.

Absenders. Der Empfänger muss nicht ebenfalls das BIC-Protokoll beherrschen. Das BIC-Protokoll ist noch vergleichsweise jung (es stammt aus dem Jahr 2004) und stellt hier einige Variablen zur Verfügung, mit denen Sie sein Verhalten einstellen können. Die Erläuterung der Optionen benötigt aber recht viel Wissen über das Protokoll selbst. Sie finden Hintergrundinformationen unter `http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/`.

tcp_congestion_control

Der Linux-Kernel 2.6.13 und neuere Versionen können verschiedene Methoden verwenden, um drohende Verstopfungen der Netzwerkverbindungen zu erkennen und zu vermeiden. Aktuell stehen die folgenden zur Verfügung:

- `reno` (klassisches TCP)
- `bic`
- `highspeed` (Sally Floyd)
- `htcp` (Hamilton TCP)
- `hybla` (Speziell für Satellitenverbindungen)
- `scalable`
- `vegas`
- `westwood` (speziell für verlustreiche Netzwerke)ts

Sie können ganz einfach die Methode wählen:

```
sysctl -w net.ipv4.tcp\_congestion\_control=htcp
```

Sie finden weitere Informationen über die Methoden auf `http://www-iepm.slac.stanford.edu/bw/tcp-eval/`.

tcp_cookie_size

Die TCP Cookie Transactions sind eine Weiterentwicklung der originalen Syncookies. Um zu funktionieren benötigen sie sowohl die Unterstützung auf dem Client als auch auf dem Server und bieten dann Schutz gegen DoS-Attacken wie den SYN-Flood. Die Default-Größe des Cookies kann hiermit eingestellt werden. Zusätzlich kann die Applikation die Größe auch je Socket einstellen. Default: 0.

tcp_dma_copybreak

Dies ist die untere Grenze der Datenmenge, die an eine DMA-Copy-Engine abgegeben wird (Offloading), falls das System über einen entsprechenden Mechanismus verfügt.

tcp_dsack

Diese Variable bestimmt, ob doppelte selektive Acknowledgments (`sack`) versendet werden sollen. Damit kann einem Absender mitgeteilt werden, dass ein Paket doppelt empfangen worden ist. D-SACK ist eine Erweiterung des SACK-Standards und wird in dem RFC2883

ts Bitte diese Formatierung bestätigen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

genauer beschrieben. SACK und D-SACK beschleunigen die Übertragung größerer Datenmengen enorm, da ein Empfänger genau die empfangenen Pakete bestätigen kann und nicht immer nur das Paket mit der kleinsten Sequenznummer (siehe auch `tcp_sack`).

tcp_ecn

Die Explicit Congestion Notification (ECN) ist eine neue Methode, mit der zwei Kommunikationspartner Informationen über Netzwerk-Verstopfungen austauschen. Beide Kommunikationspartner müssen ECN unterstützen. Leider verwerfen immer noch viele Firewalls Pakete mit ECN-Informationen als bösartige Pakete, da sie oder ihre Administratoren diesen neuen Standard noch nicht kennen. Dann können Sie mit dieser Variable ECN abschalten. Ansonsten können Sie für bestimmte IP-Adressen auch in der Mangle-Tabelle das ECN-Target verwenden (siehe Abschnitt 21.6). ECN wird in den beiden RFCs RFC3168, „The Addition of Explicit Congestion Notification (ECN) to IP", und RFC2884, „Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", besprochen.

tcp_fack

Forward Acknowledgment ist eine spezielle Methode, die bei selektiven Acknowledgments wesentlich radikaler nichtbestätigte Pakete zur Wiederversendung auswählt. Dabei geht diese Methode davon aus, dass, sobald ein selektives Acknowledgment empfangen wurde, alle Pakete mit kleineren Sequenznummern verloren gegangen sind und neu gesendet werden müssen. Da dies nur dann korrekt ist, wenn eine Umsortierung der Paketreihenfolge nicht erfolgt ist, wird das Verfahren für Verbindungen abgeschaltet, bei denen eine Umsortierung erkannt wurde. Forward Acknowledgments sind nur aktiv, wenn auch selektive Acknowledgments unterstützt werden. Da dies die Performance erhöht, sollte die Option eingeschaltet sein. Weitere Informationen finden Sie auf der Homepage des Entwicklers Matthew Mathis unter `http://staff.psc.edu/mathis/`.

tcp_fin_timeout

Wenn das lokale System eine Verbindung mit einem `FIN`-Paket beendet, muss auch der Kommunikationspartner die Verbindung mit einem `FIN`-Paket beenden. Diese Variable definiert, wie lange Ihr System auf ein Antwort-`FIN`-Paket des Kommunikationspartners warten muss. Einige Systeme reagieren hier nicht entsprechend den TCP-Standards und beenden ihre Seite der Verbindung nicht korrekt.

Dies kann zu Speicherproblemen auf Webservern führen, wenn viele defekte Clients die Verbindungen aufbauen, da für jeden offenen Socket 1,5 Kbyte Speicher reserviert werden. Der Default-Wert von 60 Sekunden kann auf diesen Systemen auf 30 Sekunden oder weniger reduziert werden.

tcp_frto

Plötzliche Timeouts bei der wiederholten Sendung von verlorenen TCP-Paketen (Spurious Retransmission Timouts, RTO) können zu langsamen Übertragungsraten führen. Diese Time-

outs treten vor allem bei unzuverlässigen Medien, zum Beispiel WLANs, auf. Das RFC4138 beschreibt die Forward RTO Recovery, die nur von dem Absender implementiert wird. Diese Option schaltet diese Methode im Kernel an.

tcp_keepalive_*

Eine Applikation kann einen Socket mit der Option `SO_KEEPALIVE` öffnen. Der Kernel sendet dann, selbst wenn der Socket nicht verwendet wird, in regelmäßigen Abständen TCP-Keepalive-Pakete, um die Verbindung geöffnet zu halten und um einen Verbindungsabbruch zu erkennen.

Die folgenden Variablen definieren, wie diese Keepalive-Pakete versendet werden:

tcp_keepalive_time:

Diese Variable definiert, in welchen Abständen der Linux-Kernel ein Paket für eine ungenutzte Verbindung aussendet. Der Default-Wert beträgt 7200 Sekunden oder 2 Stunden.

tcp_keepalive_probes:

Diese Variable definiert, wie viele unbeantwortete Keepalive-Pakete ausgesendet werden, bevor der Kernel die Verbindung als unterbrochen ansieht (Default: 9).

tcp_keepalive_intvl:

Diese Variable definiert, in welchem Intervall der Kernel ein weiteres Keepalive-Paket aussendet, wenn das letzte Paket nicht beantwortet wurde (Default: 75 Sekunden).

Der Kernel sendet also alle 2 Stunden ein Paket. Wird dieses nicht beantwortet, sendet er anschließend alle 75 Sekunden 9-mal ein weiteres Paket. Werden auch diese Pakete nicht beantwortet, ist die Verbindung tot. In gewissen Umgebungen ist es sinnvoll, den Abstand von 2 Stunden auf 30 Minuten oder weniger zu reduzieren, damit die Keepalive-Pakete ihren Zweck erfüllen.

tcp_low_latency

Diese Option definiert, welches Ziel der Kernel bei den TCP-Verbindungen verfolgen soll:

» hoher Durchsatz (High Throughput, Default) oder

» niedrige Verzögerung (Low Latency)

Ein High-Performance-Rechencluster hätte diese Variable typischerweise gesetzt.

tcp_max_orphans

Diese Variable definiert, wie viele verwaiste (*orphaned*) TCP-Sockets der Kernel toleriert, die nicht mehr mit einem User-File-Handle verbunden sind. Sobald die Anzahl überschritten wird, setzt der Kernel alle (!) verwaisten Sockets zurück. Damit sollen einfache Denial-of-Service-Situationen behandelt werden, denn jeder verwaiste Socket benötigt 64 Kbyte Arbeitsspeicher, der nicht in den Swap ausgelagert werden kann.

Sobald diese Situation eintritt, protokolliert der Kernel Folgendes: `TCP: too many of orphaned sockets`. Dann sollten Sie die Variablen `tcp_fin_timeout` und `tcp_orphans_retries` modifizieren oder diesen Wert hochsetzen.

tcp_max_syn_backlog

Der Kernel hält alle halb offenen Verbindungen in einer Tabelle (SYN-Backlog) fest. Dies sind Verbindungen, bei denen der Client nach seinem `SYN` noch nicht das `ACK`-Paket gesendet hat, um die Verbindung zu bestätigen. Diese Tabelle ist per Default 128 Einträge für Systeme mit weniger als 128 Mbyte und 1024 Einträge für alle anderen Systeme groß. Auf sehr beschäftigten Systemen (z. B. Webservern) sollten Sie diesen Wert erhöhen.

Siehe auch: `tcp_syncookies`

tcp_max_tw_buckets

Nachdem eine Applikation eine Verbindung beendet hat, wird der Socket in den Zustand Time-Wait versetzt. Diese Variable definiert, wie viele Time-Wait-Sockets insgesamt von dem Kernel unterstützt werden. Der Default-Wert von 180000 sollte nie reduziert, sondern im Fehlerfall nur erhöht werden.

tcp_mem

Diese Variable enthält drei Werte und definiert, wie der TCP-Stack seinen Speicher verwaltet. Die drei Werte sind:

- `low`: Unterer Schwellenwert.
- `pressure`: Beim Überschreiten dieses Wertes versucht TCP, Speicher zu sparen und Informationen im Speicher zu komprimieren. Das erfolgt so lange, bis der Speicherverbrauch wieder unter `low` sinkt.
- `high`: Mehr Speicher darf TCP nie verbrauchen.

Der Speicher wird nicht in Bytes, sondern in Speicherseiten gemessen. Leider ist die Größe der Speicherseite bei den verschiedenen Linux-Plattformen nicht ganz einheitlich. Die i386-Plattform verwendet meist eine 4 Kbyte große Speicherseite. Modifikationen dieser Variable können große, auch positive Auswirkungen auf die Performance haben.

tcp_moderate_rcvbuf

Diese Option wurde in dem Kernel 2.6.7 eingeführt. Sie wird auch als Dynamic Right Sizing (DRS) bezeichnet. Sie passt die Größe des TCP-Windows automatisch an die Laufzeit (Round-Trip-Time, RTT) der Pakete an. Sie verursacht durch sehr große TCP-Windows mit einigen Routern und Firewalls Probleme. Falls es bei dem Zugriff auf einige Server zum scheinbaren Totalausfall von TCP kommt, können Sie versuchen, diese Variable auf 0 zu setzen. Der Fehler liegt nicht in den Servern selbst, sondern in den Firewalls und Routern.

Damit diese Funktion aktiv ist, müssen auch `tcp_adv_win_scal` und `tcp_window_scaling` aktiv sein.

tcp_mtu_probing

Diese Variable steuert, ob TCP eine Path-MTU-Discovery unterstützen soll. Die folgenden Werte sind möglich:

`0`: abgeschaltet

`1`: abgeschaltet, aber bei Erkennung eines ICMP-Blackholes aktiv

`2`: immer eingeschaltet

tcp_no_metrics_save

Der Linux-Kernel 2.4 und 2.6 hat ein senderseitiges Autotuning. Dazu speichert er den TCP-Slow-Start-Threshold-(`ssthresh`-)Wert in dem Routing-Cache für einen bestimmten Host. Dieses Verhalten wird mit dieser Variable für den Kernel 2.6 abgeschaltet. Das ist für Hochgeschwindigkeitsnetze sinnvoll, da das Autotuning hier häufig nicht korrekt funktioniert.

tcp_orphan_retries

Diese Variable definiert, wie häufig das lokale System versucht, eine Verbindung bei dem Kommunikationspartner zu beenden, bevor die Verbindung lokal gelöscht wird. Diese Variable hat den Defaultwert 7, das entspricht bis zu 16 Minuten. Auf einem sehr beschäftigten Server (z. B. Webserver) ist es sinnvoll, diese Variable zu reduzieren. Die tatsächliche Zeitspanne hängt von dem Retransmission-Timeout (RTO) ab. Dieses Timeout wird dynamisch für die Verbindung bestimmt (RFC793). Im Wesentlichen wird der Timeout von der beobachteten Laufzeit (Round Trip Time, RTT) abgeleitet.

tcp_reordering

Diese Variable definiert, wann ein Paket als verloren gilt. Der Default-Wert von 3 sollte nicht verkleinert werden (siehe auch die RTO-Erklärung bei `tcp_orphan_retries`).

tcp_retrans_collapse

Einige Systeme, speziell Drucker, besitzen einen Fehler in ihrem TCP/IP-Stack, der eine Kommunikation unmöglich macht. Diese Variable schaltet einen Workaround ein, so dass ein Linux-System dennoch mit diesen Geräten sprechen kann.

Es sind keine negativen Nebenwirkungen bekannt.

tcp_retries1

Diese Variable definiert, wie häufig der TCP-Stack ein Paket erneut senden soll, bevor er einen Fehler melden soll (Default: 3). Siehe auch die RTO-Erklärung bei `tcp_orphan_retries`.

tcp_retries2

Diese Variable definiert, wie häufig der TCP-Stack ein Paket versenden soll, bevor die laufende Verbindung beendet wird (Default: 15). Siehe auch die RTO-Erklärung bei `tcp_orphan_retries`.

tcp_rfc1337

Das RFC1337, „TIME-WAIT Assassination Hazards in TCP", beschreibt Probleme, die auftreten, wenn alte duplizierte Pakete neue Verbindungen beeinflussen. Dies kann geschehen, da `TIME_WAIT`-Sockets für neue Verbindungen wiederverwendet werden können. Das RFC beschreibt drei verschiedene Lösungen zu diesem Problem. Der Kernel ignoriert alle RST-Pakete an Sockets im `TIME_WAIT`-Zustand, wenn diese Variable gesetzt ist.

tcp_rmem

Diese Variable bestimmt die Größe des TCP-Receive-Buffers. Die Variable enthält drei verschiedene Werte:

- `min`: garantierte Mindestgröße des Receive-Buffers für jeden Socket (Default: 4 oder 8 Kbyte).
- `default`: Default-Größe des Receive-Buffers (Default: 87380). Dieser Wert überschreibt den Wert `/proc/sys/net/core/rmem_default` für das TCP-Protokoll.
- `max`: Dies ist die maximale Größe des Receive-Buffers für TCP. Dieser Wert wird von `/proc/sys/net/core/rmem_max` überschrieben, falls der IPv4-Wert größer ist.

Wenn Sie diese Werte ändern möchten, sollten Sie sich den TCP-Tuning-Guide für Linux auf `http://fasterdata.es.net/fasterdata/host-tuning/linux/` ansehen. Dort wird vorgeschlagen, die Maximalwerte folgendermaßen zu vergrößern:

```
# increase TCP max buffer size
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
# increase Linux autotuning TCP buffer limits
# min, default, and max number of bytes to use
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
```

tcp_rfc1337

Falls diese Variable gesetzt ist, verhält sich der TCP-Stack entsprechend dem RFC1337. Falls die Variable nicht gesetzt ist, wird das RFC ignoriert, aber dennoch die TCP TIME_WAIT Assassination verhindert.

tcp_sack

Diese Variable aktiviert selektive Acknowledgments entsprechend dem RFC2018, „TCP Selective Acknowledgement Options“, und dem RFC2883, „An Extension to Selective Acknowledgement (SACK) Option for TCP“. Selektive Acknowledgments erhöhen insbesondere bei Netzwerkmedien mit Paketverlusten den Durchsatz. Da SACKs keine negativen Auswirkungen haben, sollte diese Option immer eingeschaltet werden.

tcp_stdurg

Es gibt zwei verschiedene Varianten, wie ein TCP/IP-Stack mit dem Urgent-Pointer und dem `URG`-TCP-Flag umgehen kann. RFC793 beschreibt das Verhalten, das auch von dem Betriebssystem BSD 4.2 verwendet wird. Dies ist das Standardverhalten des Linux-Kernels. Wenn Sie diese Variable aktivieren, verhält sich jedoch der Linux-Kernel wie in RFC1122, „Requirements for Internet Hosts – Communication Layers“, beschrieben. Dieses Verhalten ist nicht kompatibel und kann zu Funktionsstörungen führen, daher ist die Funktion per Default nicht aktiv.

tcp_synack_retries

Diese Variable definiert, wie häufig ein Kernel versuchen soll, `SYN`/`ACK`-Pakete als Antwort auf einen Verbindungsaufbau zu senden. Der Default-Wert ist 5. Da ein Versuch etwa 30–40 Sekunden dauert, beträgt der gesamte Timeout 180 Sekunden.

tcp_syncookies

Hiermit aktivieren Sie SYN-Cookies im Linux-Kernel. Der Linux-Kernel wird anfangen, SYN-Cookies zu versenden, sobald der SYN-Backlog überläuft. Damit können Sie sich gegen einen SYN-Flood wehren.

Diese Funktion ist lediglich ein Fallback für den Fall eines Angriffs. Sie kann zu großen Problemen führen. Achten Sie darauf, dass Ihr Linux-System die SYN-Cookies nicht im Normalbetrieb versendet! Wenn Ihr Server so beschäftigt ist, dass der SYN-Backlog im normalen Betrieb überläuft, sollten Sie den SYN-Backlog vergrößern (siehe `tcp_max_syn_backlog`).

Die TCP-SYN-Cookies ermöglichen es dem Linux-System, bei einem SYN-Flood ein späteres ACK-Paket eines echten Clients zu erkennen und trotzdem die Verbindung aufzubauen. Die SYN-Cookies wurden u.a. von Dan Bernstein entwickelt und hier beschrieben:
`http://cr.yp.to/syncookies.html`.

tcp_syn_retries

Diese Variable definiert, wie häufig der Linux-Kernel versuchen soll, selbst eine Verbindung aufzubauen. Der Default-Wert beträgt 5. Da ein Versuch etwa 30–40 Sekunden dauert, beträgt der gesamte Timeout 180 Sekunden.

tcp_thin_dupack

Diese Funktion erlaubt die dynamische Triggerung von erneuten Sendungen (Retransmissions) nach doppelten Bestätigungen (`dupACK`) in „dünnen" TCP-Streams. Ein TCP-Stream gilt als dünn, wenn weniger als 4 Pakete aktuell übertragen werden. Weitere Informationen finden Sie in der Datei `Documentation/networking/tcp-thin.txt`.

tcp_thin_linear_timeouts

Diese Variable aktiviert die Triggerung von linearen Timeouts für „dünne" TCP-Streams. Dies verbessert die Latenz der Retransmissions.

tcp_timestamps

RFC1323, „TCP Extension for High Performance", beschreibt die Verwendung von Timestamps, um die Round Trip Time (RTT) zu ermitteln. Grundsätzlich verbessert diese Option die Funktion des TCP-Protokolls.

Nmap verwendet die TCP-Timestamps, um die Uptime eines Systems zu ermitteln. Wenn Sie das verhindern möchten, müssen Sie diese Option abschalten.

tcp_tso_win_divisor

Die TCP-Segmentation-Offload-(TSO-)Funktion erlaubt es, die Aufteilung großer Pakete in kleine Pakete an die Netzwerkkarte abzugeben. Der `e1000`-Treiber unterstützt dies seit dem Kernel 2.5.33. Der TCP-Stack übergibt 64 Kbyte große Pakete an die Karte, die sie in 1500 Byte große Pakete entsprechend der MTU aufteilt. Dies reduziert die Prozessorlast enorm.

INFO

Die Linux-Kernel 2.6.11 und ältere Versionen haben einen Fehler in der Implementierung. Hier sollten Sie die TSO-Funktion für die Netzwerkkarte abschalten:

```
ethtool -K eth0 tso off
```

Diese Variable definiert, welchen Anteil (in Prozent) des Netzwerkkartenpuffers ein TSO-Paket einnehmen darf.

tcp_tw_*

Diese Parameter beeinflussen die Behandlung der `TIME_WAIT`-Sockets. Die Variable `tcp_tw_reuse` bestimmt, ob eine Applikation einen TIME_WAIT-Socket wiederverwenden darf. Die Variable `tcp_tw_recycle` definiert, ob eine andere Applikation einen `TIME_WAIT`-Socket nutzen darf.

Die Verwendung dieser Optionen kann zu großen Problemen führen. Stellen Sie auf jeden Fall sicher, dass gleichzeitig auch die Variable `tcp_rfc1337` aktiviert wurde.

Sie sollten diese Optionen nur nutzen, wenn Sie nicht über genug Sockets für Ihre Applikation verfügen.

tcp_vegas_*

TCP-Vegas ist eine senderseitige Veränderung des TCP-Protokolls, die versucht, Verstopfungen durch Schätzung der Bandbreite zu ermitteln. Hierzu ermittelt es die Verzögerungen der Pakete, anstatt den Paketverlust auszuwerten (TCP-Reno). TCP-Vegas modifiziert dann die Sendegeschwindigkeit durch Modifikationen des Congestion-Fensters. TCP-Vegas erzeugt damit einen geringeren Paketverlust als TCP-Reno (Default-TCP). Sie finden weitere Informationen auf der Vegas-Homepage (`http://www.cs.arizona.edu/protocols/`).

Anstelle von TCP-Vegas verwenden moderne Linux-Kernel jedoch bereits TCP-BIC.

tcp_westwood

TCP-Westwood ist eine weitere Methode, mit der Verstopfungen der Netzwerkverbindung vorgebeugt werden kann. Auch hier handelt es sich um eine nur senderseitige Modifikation des Reno-TCP-Stacks. Auch diese Methode schätzt die Bandbreite der Verbindung, um das Congestion Window und die Slow Start Threshold (`ssthresh`) zu setzen.

Weitere Informationen über diese Methode finden Sie auf `http://c3lab.poliba.it/index.php/Westwood` und `http://www.cs.ucla.edu/NRL/hpi/tcpw/`.

tcp_window_scaling

Diese Variable schaltet das TCP-Window-Scaling an. Damit können Absender und Empfänger größere TCP-Windows als 64 Kbyte verwenden. Dies verbessert die Performance bei Netzwerkmedien mit hoher Bandbreite und hoher Latenz und reduziert die Verluste durch nicht ausreichende Nutzung der Bandbreite.

Größere Werte erlauben es einem Angreifer jedoch auch, gültige Sequenznummern leichter zu erraten und damit zum Beispiel einen TCP-Reset-Angriff durchzuführen.

tcp_wmem

Diese Variable verwaltet ähnlich `tcp_rmem` den Sende-Puffer des TCP-Stacks. Es handelt sich genauso um drei Werte: `min`, `default` und `max`. Sie finden bei `tcp_rmem` eine Beschreibung der Werte. Eine Anpassung des `max`-Wertes nach oben kann erstaunliche Leistungssteigerungen bringen.

tcp_workaround_signed_windows

Ist diese Variable gesetzt, so vermutet das Linux-System, dass der TCP-Stack der Gegenseite defekt ist, wenn keine Window-Scaling-Option erhalten wird.

23.3.7 udp_*

Aktuelle Kernel weisen auch einige Variablen zur Verwaltung des UDP-Protokolls auf.

udp_mem

Diese Variable entspricht der Variable `tcp_mem` und enthält ebenso drei Werte.

- `low`: unterer Schwellenwert
- `pressure`: Beim Überschreiten dieses Wertes versucht UDP, Speicher zu sparen und Informationen im Speicher zu komprimieren. Das erfolgt so lange, bis der Speicherverbrauch wieder unter `low` sinkt.
- `high`: Mehr Speicher darf UDP nie verbrauchen.

Der Speicher wird nicht in Bytes, sondern in Speicherseiten gemessen. Leider ist die Größe der Speicherseite bei den verschiedenen Linux-Plattformen nicht ganz einheitlich. Die i386-Plattform verwendet meist eine 4 Kbyte große Speicherseite. Modifikationen dieser Variable können große, auch positive Auswirkungen auf die Performance haben.

udp_rmem_min

Dies ist die minimale Größe des Empfangspuffers für UDP-Sockets. Die Einheit sind Bytes. Der Default beträgt 4096.

udp_wmem_min

Dies ist die minimale Größe des Sendepuffers für UDP-Sockets. Per Default entspricht er dem Empfangspuffer.

23.3.8 cipso_*

Aktuelle Kernel verfügen auch über CIPSO-Variablen. Diese steuern das Labeln von Netzwerkpaketen mit der Common IP Security Option nach einem IETF-Draft. So wie SELinux lokale Dateien mit Labeln versieht, bietet CIPSO eine ähnliche Funktion für IPv4-Pakete. Nähere Informationen können dem Dokument entnommen werden: `http://www.ietf.org/wg/concluded/cipso.html`.

Aktuell wird dieses Labeling jedoch von keiner Distribution genutzt.

23.3.9 /proc/sys/net/ipv4/conf

Dieses Verzeichnis enthält ein Unterverzeichnis für jede verfügbare Netzwerkkarte. Zusätzlich existieren die Verzeichnisse `all/` und `default/`. Änderungen der Variablen in dem Verzeichnis `all/` haben Auswirkungen auf alle Netzwerkkarten. Das Verzeichnis `default/` definiert, welche Werte den Variablen neuer Netzwerkkarten zugewiesen werden. Die Werte bereits aktivierter Netzwerkkarten werden nicht verändert.

```
$ ls -l /proc/sys/net/ipv4/conf
insgesamt 0
dr-xr-xr-x 2 root root 0 2. Nov 13:36 all
dr-xr-xr-x 2 root root 0 2. Nov 13:36 default
dr-xr-xr-x 2 root root 0 2. Nov 13:36 eth0
```

```
dr-xr-xr-x 2 root root 0 2. Nov 13:36 lo
dr-xr-xr-x 2 root root 0 2. Nov 13:36 wlan0
dr-xr-xr-x 2 root root 0 2. Nov 13:36 wwan0
```

accept_local

Hiermit akzeptiert diese Netzwerkkarte Pakete von lokalen Adressen. Default: abgeschaltet.

accept_redirects

Diese Variable definiert, ob das System ICMP-Redirect-Nachrichten auswertet. Diese Nachrichten werden von Routern versendet, um andere Router und Rechner auf eine bessere Route hinzuweisen. Ältere Linux-Kernel akzeptierten per Default ICMP-Redirects (1). Dies erlaubt aber auch ein Router-Spoofing, daher sollte es mindestens auf der Firewall ausgeschaltet werden.

accept_source_route

Dieser Parameter definiert, ob das System IP-Pakete mit der IP-Option Source-Routing akzeptiert. Da damit ein Angreifer Pakete über selbst gewählte Routen versenden kann, sollte diese Option abgeschaltet sein (0). Dies ist auch der Default.

arp_accept

Hiermit kontrollieren Sie, ob ARP-Antworten den ARP-Cache modifizieren, wenn der entsprechende Rechner noch nicht eingetragen ist. Befindet sich der Rechner bereits im Cache, so wird sein Eintrag immer, unabhängig von dieser Variable, aktualisiert.

arp_announce

Diese Variable definiert, welche Source-IP-Adresse in ARP-Requests verwendet wird.

- **0:** Verwendet eine beliebige IP-Adresse (Default).
- **1:** Vermeidet IP-Adressen, die sich nicht in demselben Subnetz wie das Ziel der ARP-Anfrage befinden.
- **2:** Verwendet die IP-Adresse, die auch für die Kommunikation mit dem Zielsystem verwendet werden würde.

arp_filter

Diese Variable definiert, wie der Kernel auf ARP-Anfragen antwortet. Wenn Sie mehrere Netzwerkkarten für das Load-Balancing in demselben Subnetz betreiben, sollten Sie diesen Parameter auf 1 setzen.

arp_ignore

Diese Variable definiert, wie das Linux-System auf ARP-Anfragen reagiert.

- 0: Beantworte jede ARP-Anfrage für jede lokale IP-Adresse, selbst wenn diese nicht auf der betroffenen Netzwerkkarte definiert ist (Default).
- 1: Beantworte nur ARP-Anfragen für IP-Adressen, die auf der ankommenden Netzwerkkarte definiert sind.
- 2: Beantworte nur ARP-Anfragen für IP-Adressen, die auf der ankommenden Netzwerkkarte definiert sind und die sich im selben Subnetz mit der Source-Adresse der ARP-Anfrage befinden.
- 3: Anfragen für lokale Adressen mit dem Scope Host werden nicht beantwortet.
- 8: Beantworte keine ARP-Anfragen.

INFO

Diese Variable ist dafür zuständig, dass Sie ein Linux-System auch unter Benutzung einer IP-Adresse anpingen können, die auf einer anderen Netzwerkkarte des Linux-Systems konfiguriert wurde. Dies funktioniert mit der Default-Einstellung auch dann, wenn das Forwarding auf dem Linux-System abgeschaltet wurde!

arp_notify

Hiermit steuern Sie, ob das Linux-System automatisch Gratuitous-ARP-Antworten versenden soll. Ist diese Option aktiviert, so versendet Linux, wenn eine Netzwerkkarte aktiviert wird oder die Hardware-Adresse sich ändert, automatisch ein Gratuitous-ARP.

bootp_relay

Diese Option muss gesetzt werden, wenn Sie auf dem Linux-System einen Bootp-Relay-Daemon betreiben möchten. Die Netzwerkkarte akzeptiert dann auch Pakete mit der Source-Adresse `0.b.c.d`.

disable_policy

Dies deaktiviert IPsec-Policies für diese Netzwerkkarte.

disable_xfrm

Dies deaktiviert IPsec-Verschlüsselung für diese Netzwerkkarte.

force_igmp_version

Hiermit können Sie die IGMP-Version für diese Netzwerkkarte erzwingen:

```
/sbin/sysctl -w net.ipv4.conf.eth0.force_igmp_version=3
```

forwarding

Hiermit aktivieren Sie das Forwarding nur für diese Netzwerkkarte. Wenn Sie diese Variable nutzen möchten, sollten Sie nicht die Variable `ip_forward` verwenden.

log_martians

Diese Variable protokolliert Pakete mit unmöglichen IP-Adressen. Dies sind Pakete, die von `rp_filter` erkannt wurden oder Source-Routing-IP-Optionen verwenden.

mc_forwarding

Wenn Sie einen Multicast-Routing-Daemon betreiben möchten, müssen Sie diese Variable einschalten.

medium_id

Diese Variable unterscheidet Geräte anhand des Netzwerkmediums, mit dem sie verbunden sind. Diese Variable wird für ProxyARP verwendet. ProxyARP wird von dem Kernel nur für Pakete aktiviert, die zwischen zwei Geräten weitergeleitet werden, die nicht mit demselben Medium verbunden sind. Ein Wert von -1 bedeutet, dass das Medium unbekannt ist. Der Wert 0 bedeutet, dass dieses Gerät das einzige ist, das mit dem Medium verbunden ist.

promote_secondaries

Wenn Sie einen Alias auf einer Netzwerkkarte definieren und anschließend die primäre IP-Adresse löschen, wird automatisch auch der Alias gelöscht. Die Option `promote_secondaries` sorgt dafür, dass der Alias automatisch zur primären IP-Adresse wird.

proxy_arp

Diese Variable aktiviert ProxyARP für diese Netzwerkkarte.

proxy_arp_pvlan

Hiermit arbeitet das ProxyARP nur innerhalb eines privaten VLANs nach RFC3069. Dabei werden ProxyARP-Anfragen nur auf der Netzwerkkarte beantwortet, auf der auch die Anfrage angenommen wurde. Moderne Switches erlauben die Konfiguration von privaten VLANs, in denen die Rechner im VLAN nicht untereinander, sondern nur mit einem zentralen Router kommunizieren dürfen. Für die Kommunikation untereinander kann dann die Kommunikation über den Router genutzt werden. Dieser kann dann auch den Verkehr mit Firewallregeln einschränken.

INFO

Diese Technologie hat unterschiedliche Namen. In dem angesprochenen RFC3069 wird sie als VLAN Aggregation *bezeichnet. Cisco bezeichnet die Funktionalität als* Private VLAN. *Hewlett Packard bezeichnet sie als* Port-Isolation *oder* Source-Port Filtering.

rp_filter

Diese Variable schaltet einen Reverse-Path-Filter für diese Netzwerkkarte an. Dieser Filter prüft, ob eine Antwort an den Absender des Pakets auch über diese Netzwerkkarte versendet werden würde. Diese Variable bietet so einen gewissen Schutz vor IP-Spoofing-Angriffen. Jedoch verursacht diese Funktion auch häufig Probleme, wenn Sie Policy-Routing, Advanced-Routing oder IPsec einsetzen. Dann ist es sinnvoll, diese Funktion abzuschalten. Der Schutz, den diese Funktion bietet, kann auch leicht mit Firewall-Regeln erreicht werden.

secure_redirects

Normalerweise akzeptiert ein Linux-System sämtliche ICMP-Redirects. Wenn Sie diese Variable aktivieren, werden nur Redirects von bekannten Default-Gateways akzeptiert. Diese Funktion erfordert zusätzlich, dass die Variable `shared_media` gesetzt ist.

send_redirects

Diese Variable weist das Linux-System an, ICMP-Redirects auszusenden, falls das System ein Router ist und eine bessere Route kennt.

shared_media

Diese Variable definiert, ob das angeschlossene Netzwerkmedium von mehreren Netzen gleichzeitig genutzt wird. Nur dann versendet der Kernel ICMP-Redirect-Nachrichten als Router und akzeptiert sie als Host.

src_valid_mark

Linux kann Pakete markieren und in Abhängigkeit von der Markierung routen. Dies wird als Policy-Routing bezeichnet. In gewissen Umgebungen kann es zu Problemen kommen, wenn die Prüfung der Rückwärtsroute eine andere Routing-Tabelle nutzen muss, als das eingehende Paket genutzt hat. Dies kommt zum Beispiel bei transparenten Proxies vor. Mit dieser Variable wird diese Unterstützung kontrolliert.

tag

Hier können Sie eine beliebige Nummer speichern. Diese Nummer wird nur von Ihnen verwendet. Der Kernel ignoriert diese Variable.

/proc/sys/net/ipv4/neigh

Diese Parameter betreffen das ARP-Protokoll. Jeden Parameter hier aufzuführen, würde dieses Kapitel sprengen, daher möchte ich Sie auf die `arp(7)`-Manpage verweisen, in der die Parameter erläutert werden.

Bitte Änderung bestätigen.

/proc/sys/net/ipv4/route

Diese Variablen verwalten den Routing-Cache und das Versenden von Fehler-Nachrichten.

23.3.10 /proc/sys/net/netfilter

Ab der Version 2.6.9 verfügt der Linux-Kernel über das Verzeichnis, in dem sich verschiedene Variablen befinden, die das Connection Tracking betreffen. Diese Variablen werden in dem entsprechenden Kapitel besprochen (siehe Kapitel 19). Dieses Verzeichnis befindet sich auf aktuellen Kerneln eine Ebene tiefer in dem `ipv4`-Verzeichnis.

23.3.11 /proc/sys/net/ipv6

Dieses Verzeichnis enthält die Variablen für das IPv6-Protokoll. Da es für das TCP- und UDP-Protokoll unerheblich ist, ob sie per IPv4 oder IPv6 übertragen werden, existieren in diesem Verzeichnis keine entsprechenden Variablen. Die Variablen im IPv4-Verzeichnis steuern auch das Verhalten der Protokolle bei Nutzung von IPv6.

Daher befinden sich in diesem Verzeichnis nur einige IPv6-spezifische Dateien.

bindv6only

IPv6 erlaubt auch die Einbettung von IPv4-Adressen innerhalb der IPv6-Adressen. Hiermit kann dieses Verhalten abgeschaltet werden.

ip6frag_high_thresh

IPv6 erlaubt auch die Fragmentierung von Paketen. Im Gegensatz zum IPv4-Protokoll werden bei IPv6 die Pakete jedoch durch den Absender und nicht durch Router während der Übertragung fragmentiert. Der Empfänger muss dann die fragmentierten Pakete wieder zusammensetzen. HIerzu benötigt er Arbeitsspeicher, dessen maximale Größe mit dieser Variablen angegeben werden kann.

ip6frag_low_thresh

Wurde der maximale Wert erreicht, so versucht der Kernel den Speicherverbrauch unterhalb dieses Wertes wieder zu drücken.

ip6frag_time

Der Kernel speichert die Fragmente maximal für diese Zeit im Arbeitspeicher zwischen.

23.3.12 /proc/sys/net/ipv6/conf/

Hier können für die einzelnen Netzwerkkarten getrennte Einstellungen vorgenommen werden.

forwarding

Hiermit wird die Weiterleitung für IPv6 an- bzw. abgeschaltet. Während bei IPv4 mit dieser Variablen bei jeder Netzwerkkarte einzeln die Funktion an- und abgeschaltet werden konnte, ist dies bei IPv6 nicht möglich. Hier muss die Weiterleitung immer für alle Karten aktiviert werden. Nur mit der Firewall können Sie dann die Netzwerkkarten einschränken!

proxy_ndp

Hiermit führen Sie für das System eine Proxy-Neighbor-Discovery durch. Dies entspricht dem ProxyARP bei IPv4.

accept_ra

Diese Datei entscheidet, ob das System Router Advertisements für die Autokonfiguration entgegennimmt. Mögliche Werte sind:

0: Verwirf Router Advertisements

1: Akzeptiere Router Advertisements, wenn das Forwarding abgeschaltet ist.

2: Akzeptiere auch dann Router Advertisements, wenn das Forwarding aktiv ist.

Default: 1

accept_ra_defrtr

Diese Variable definiert, ob der Router aus dem Router Advertisement auch als Default-Router eingetragen wird.

accept_ra_pinfo

Hiermit entscheiden Sie, ob die Präfix-Information aus dem Router Advertisement verwendet wird.

accept_ra_rt_info_max_plen

Dieser Parameter entscheidet, wie lang maximal das Präfix in dem Router Advertisement sein darf.

accept_ra_rtr_pref

Dieser Parameter definiert, ob die Präferenz aus dem Router Advertisement ausgewertet wird.

accept_redirects

Analog IPv4.

accept_source_route

IPv6 unterstützt zwei Routing Header:

`Type 0`: Dieser Routing Header entspricht dem originalen Source-Routing bei IPv4 und wurde mit dem RFC5095

`Type 2`: Dieser Routing Header wird für Mobile IPv6 benötigt.

Mit dieser Variable entscheiden Sie, ob keine Routing Header (negativer Wert) oder Routing Header vom Type 2 (positiver Wert oder 0) akzeptiert werden. Den Routing Header Type 0 können Sie nicht akzeptieren.

autoconf

Dies schaltet die Autokonfiguration mit Router Advertisements an bzw. ab.

dad_transmits

IPv6 verlangt die Durchführung der Duplicate Address Detection vor der Aktivierung einer IP-Adresse. Diese Variable definiert deren Häufigkeit.

forwarding

Dieser Parameter definiert, wenn das globale IPv6 Forwarding aktiviert wurde (siehe Abschnitt 23.3.12), wie sich die Netzwerkkarte verhalten soll. Die Variable unterstützt die folgenden drei Werte:

`0`: Forwarding abgeschaltet
Dies ist der Default. Das System verhält sich wie ein Host. Das bedeutet:

1. Das IsRouter Flag wird in Neighbour Advertisements nicht gesetzt.
2. Router Solicitations werden, wenn nötig, versandt.
3. Wenn `accept_ra` gesetzt ist, wird die Autokonfiguration durchgeführt.
4. Redirects werden akzeptiert, wenn `accept_redirects` gesetzt ist.

`1`: Forwarding angeschaltet

Das System verhält sich wie ein Router. Dies ist das Gegenteil vom ersten Fall:

1. Das IsRouter Flag wird in Neighbour Advertisements gesetzt.
2. Router Solicitations werden nicht versandt.
3. Router Advertisements werden ignoriert, außer `accept_ra` ist 2.
4. Redirects werden ignoriert.

`2`: Forwarding angeschaltet (Hybrid Mode)

Der Hybrid Mode unterscheidet sich lediglich im Punkt 2. Router Solicitations werden versandt, wenn sie benötigt werden.

Fehlt hier Text? Falls ja, bitte ergänzen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

hop_limit

Das Hop-Limit (TTL bei IPv4). Der Default ist 64.

mtu

Die Maximum Transmission Unit (MTU). Der Default beträgt 1280.

router_probe_interval

Nach RFC4191 kann ein Rechner Router testen. Dies ist das Intervall zwischen zwei Tests (Default: 60).

router_solicitation_delay

Nachdem eine Netzwerkkarte aktiviert wurde, wartet sie diesen Zeitraum, bevor Router Solicitations versandt werden (Default: 1).

router_solicitation_interval

Wurde auf eine Router Solicitation keine Antwort erhalten, wird diese in diesen Abständen wiederholt (Default: 4).

router_solicitations

Dies ist die maximale Anzahl von Router Solicitations (Default: 3).

use_tempaddr

Hiermit definieren Sie, ob das System die Privacy Extensions nach RFC3041 benutzen soll.

0: Abgeschaltet (auch negative Werte)

1: Eingeschaltet, aber normale Adressen werden den temporären Adressen vorgezogen.

2: Eingeschaltet (auch andere positive Werte)

temp_valid_lft

Valid Lifetime für temporäre Adressen. Default: 604800 (7 Tage).

temp_prefered_lft

Preferred Lifetime für temporäre Adressen. Default: 86400 (1 Tag).

max_desync_factor

Dies ist der maximale DESYNC-Faktor. Dabei handelt es sich um eine Zufallszahl, um zu verhindern, dass mehrere Clients neue Adressen gleichzeitig erzeugen (Default: 600).

regen_max_retry

Maximale Anzahl der Versuche, eine temporäre Adresse zu generieren (Default: 5).

max_addresses

Dies ist die maximale Anzahl von Adressen, die mittels Autokonfiguration je Netzwerkkarte erzeugt werden. Um einen Angriff mit zu vielen Adressen zu vermeiden, sollte der Wert nicht zu groß oder 0 (unlimitiert) gewählt werden (Default: 16).

disable_ipv6

Dieser Parameter schaltet IPv6 ab.

accept_dad

Dieser Parameter definiert das Verhalten in der Duplicate Address Detection.

- 0: Schalte DAD ab.
- 1: Schalte DAD an (Default).
- 2: Schalte DAD an, und deaktiviere IPv6, wenn eine MAC-basierte doppelte Link-Local-Adresse gefunden wurde.

force_tllao

Dies sendet die Link-Layer-Adresse des Ziels, selbst wenn eine Unicast-Neighbour Solictiation durchgeführt wird.

Das folgende Zitat aus RFC2461 erklärt die Arbeitsweise: „The option MUST be included for multicast solicitations in order to avoid infinite Neighbour Solicitation recursion when the peer node does not have a cache entry to return a Neighbour Advertisements message. When responding to unicast solicitations, the option can be omitted since the sender of the solicitation has the correct link- layer address; otherwise it would not have be able to send the unicast solicitation in the first place. However, including the link-layer address in this case adds little overhead and eliminates a potential race condition where the sender deletes the cached link-layer address prior to receiving a response to a previous solicitation.“

icmp_ratelimit

Hiermit geben Sie die Mindestabstände der ICMPv6-Pakete in Millisekunden an. 0 deaktiviert diese Funktion (Default: 1000).

23.3.13 /proc/sys/net/bridge/proc/sys/net/

bridge-nf-call-arptables

Dieser Wert definiert, ob von der Bridge weitergeleitete ARP-Pakete die FORWARD-Kette des `arptables`-Befehls durchlaufen. Default: 1 (ja).

bridge-nf-call-iptables

Hiermit entscheiden Sie, ob von der Bridge weitergeleitete IPv4-Pakete von den iptables-Ketten verarbeitet werden. Default: 1 (ja).

bridge-nf-call-ip6tables

Hiermit entscheiden Sie, ob von der Bridge weitergeleitete IPv6-Pakete von den ip6tables-Ketten verarbeitet werden. Default: 1 (ja).

bridge-nf-filter-vlan-tagged

Hiermit entscheiden Sie, ob von der Bridge weitergeleitete getaggte VLAN-Pakete von den entsprechenden (arp-, ip-, ip6tables-)Ketten verarbeitet werden. Default: 1 (ja).

bridge-nf-filter-pppoe-tagged

Hiermit entscheiden Sie, ob von der Bridge weitergeleitete getaggte PPPoE-Pakete von den entsprechenden (ip-, ip6tables-)Ketten verarbeitet werden. Default: 1 (ja).

24. Fortgeschrittene Protokollierung

Wenn Sie die Protokolle Ihrer Firewall anspruchsvoll weiterverarbeiten möchten, werden Sie möglicherweise schnell an die Grenzen des `LOG`-Targets stoßen. Dann sollten Sie sich das `ULOG`-Target ein wenig genauer ansehen. Es gibt mehrere Softwarepakete, die mit diesem Target zusammenarbeiten und die Protokolle schreiben können. Zwei davon möchte ich Ihnen vorstellen: `ulogd` und `specter`. Die Auswertung kann mit einigen der Werkzeuge erfolgen, die ich im Kapitel 11, „Protokollanalyse“, vorgestellt habe.

24.1 Das ULOG-Target

Dieses Ziel erlaubt es Ihnen, Pakete zur Protokollierung an eine beliebige Userspace-Applikation weiterzuleiten. Diese Applikation muss den `netlink`-Socket betrachten. Am einfachsten verwenden Sie `ulogd` oder `specter` für die Protokollierung. Diese nehmen die Parameter und Pakete entgegen und können die Protokollierung in einer Datei oder einer Datenbank vornehmen (siehe Kapitel 11). Damit der Protokolldienst zwischen den verschiedenen Protokolldateien oder Datenbanken unterscheiden kann, können Sie die Protokollierung mit den folgenden Optionen genauer bestimmen:

» `--ulog-nlgroup <nlgroup>`: `ULOG` unterstützt 32 verschiedene Netlink-Gruppen (1–32). Sie können in dem `ulogd` für jede Gruppe zum Beispiel eine andere Datenbank definieren.

» `--ulog-prefix „Zeichenkette“`: Ähnlich dem `LOG`-Target können Sie auch hier eine Zeichenkette (maximal 32 Zeichen) definieren, die jeder Meldung vorangestellt wird.

» `--ulog-cprange <paketgröße>`: Für die spätere Analyse ist es nützlich, das Paket oder zumindest Teile des Pakets untersuchen zu können. Hiermit können Sie angeben, wie viele Bytes des Pakets an das Userspace-Programm übergeben und damit möglicherweise protokolliert werden sollen. Der Default-Wert 0 kopiert das ganze Paket.

» `--ulog-qthreshold <queuegröße>`: Da die Übergabe des Pakets von dem Kernel in den Userspace mit aufwendigen Überprüfungen verbunden ist, können Sie hier definieren, wie viele Pakete der Kernel vor der Übergabe sammeln (1–50) und gemeinsam übergeben soll. Der Default-Wert ist aus Gründen der Rückwärtskompatibilität 1. Diesen Wert können Sie für jede Regel einzeln definieren.

```
iptables -A FORWARD -p icmp -m length --length 200: -j ULOG --ulog- ↵
        nlgroup 2 --ulog-prefix "Großes ICMP: " --ulog-qthreshold 10
```

24.2 Das ipt_ULOG-Kernelmodul

Beim Laden des `ipt_ULOG`-Kernelmoduls können Sie einige zusätzliche Parameter definieren, die das Verhalten und die Geschwindigkeit beeinflussen können. Diese Optionen möchte ich Ihnen hier vorstellen. Wenn nicht alle hier aufgeführten Optionen von Ihrem Modul unterstützt werden, setzen Sie wahrscheinlich eine ältere Kernel-Version ein. Um diese Optionen zu nutzen, müssen Sie vor der ersten Regel, die das `ULOG`-Target verwendet, das Modul manuell mit `modprobe` laden.

» `nlbufsiz=<zahl>`: Hiermit geben Sie die Größe des Netlink-Puffers an. Der Puffer darf maximal eine Größe von 128 Kbyte haben. Größere Puffer erhöhen die Geschwindigkeit, verzögern aber möglicherweise die Protokollierung leicht (Default: 4096).

» `flushtimeout:int`: Dieser Wert definiert, in welchen Intervallen spätestens der Puffer geleert werden muss, auch wenn der Puffer nicht voll ist. Damit kann die mögliche Verzögerung durch einen großen Puffer beschränkt werden. Die Zeiteinheit ist 10 ms auf x86-Systemen. Der Default ist 10 (100 ms).

» `nflog:int`: Diese Option definiert, ob dieses Modul auch für die Filterung von internen Netfilter-Meldungen verwendet werden soll. Dies sind zum Beispiel Meldungen über ungültige Pakete. Der Default ist 1 (Ja).

24.3 Der Ulogd-Daemon

Der `ulogd`-Daemon wurde von Harald Welte geschrieben und existiert im Moment in zwei Versionen. Auf `http://gnumonks.org/projects` finden Sie den Ulogd-Daemon in der Version 1 für die Verwendung mit dem `ULOG`-Target. Wenn Sie bereits den Kernel 2.6.14 mit dem `NFLOG`-Target einsetzen, können Sie den Ulogd-Daemon in der Version 2 verwenden, der immer noch nur als Beta-Version verfügbar ist (`http://www.netfilter.org/projects/ulogd/`). Moderne Distributionen bringen diesen Daemon bereits als Paket mit (z. B. Fedora 14: ulogd-1.24-13). Die aktuellste Version kann von `http://www.netfilter.org/projects/ulogd/downloads.html` heruntergeladen werden. Diese Version benötigt jedoch zusätzliche Bibliotheken:

» libnfnetlink

» libnetfilter_conntrack

» libnetfilter_log

```
$ wget http://www.netfilter.org/projects/ulogd/files/ulogd-2.0.0beta4.tar ↵
       .bz2
$ tar xjf ulogd-2.0.0beta4.tar.bz2
$ cd ulogd-2.0.0beta4/
$ ./configure
$ make
$ sudo make install
```

Der Ulogd verwendet Plug-Ins, um die verschiedenen Funktionen zur Verfügung zu stellen. Diese Plug-Ins werden in einer Konfigurationsdatei beschrieben, in der Sie festlegen, welche Meldungen wie protokolliert werden sollen. Eine Beispieldatei ist im Folgenden abgedruckt:

```
# Example configuration for ulogd
# $Id: ulogd.conf.in 714 2005-02-19 21:33:43Z laforge $

[global]
##########################################################################
# GLOBAL OPTIONS
##########################################################################

# netlink multicast group (the same as the iptables --ulog-nlgroup param)
nlgroup=1

# logfile for status messages
logfile="/var/log/ulogd/ulogd.log"

# loglevel: debug(1), info(3), notice(5), error(7) or fatal(8)
loglevel=5

# socket receive buffer size (should be at least the size of the in- ↵
        kernel buffer (ipt_ULOG.o 'nlbufsiz' parameter)
rmem=131071

# libipulog/ulogd receive buffer size, should be > rmem
bufsize=150000

##########################################################################
#PLUGIN OPTIONS
##########################################################################

# We have to configure and load all the plugins we want to use
# general rules:
# 1. load the plugins _first_ from the global section
# 2. options for each plugin in seperate section below

# ulogd_BASE.so - interpreter plugin for basic IPv4 header fields you  ↵
        will always need this
plugin="/usr/lib/ulogd/ulogd_BASE.so"

# output plugins.
plugin="/usr/lib/ulogd/ulogd_LOGEMU.so"
#plugin="/usr/lib/ulogd/ulogd_OPRINT.so"
```

```
#plugin="/usr/lib/ulogd/ulogd_MYSQL.so"
#plugin="/usr/lib/ulogd/ulogd_PGSQL.so"
#plugin="/usr/lib/ulogd/ulogd_SQLITE3.so"
#plugin="/usr/lib/ulogd/ulogd_PCAP.so"

[LOGEMU]
file="/var/log/ulogd/ulogd.syslogemu"
sync=1

[OPRINT]
file="/var/log/ulogd/ulogd.pktlog"

[MYSQL]
table="ulog"
pass="g3h31m"
user="ulog"
db="ulogd"
host="localhost"

[PGSQL]
table="ulog"
schema="public"
pass="g3h31m"
user="ulog"
db="ulogd"
host="localhost"

[SQLITE3]
table="ulog"
db="/path/to/sqlite/db"
buffer=200

[PCAP]
file="/var/log/ulogd/ulogd.pcap"
sync=1
```

Der Ulogd-Daemon kennt Input-, Filter- und Output-Plug-Ins. Während die Input-Plug-Ins als Datenquelle fungieren, interpretieren und filtern die Filter-Plug-Ins die Daten, die mit den Output-Plug-Ins geschrieben werden.

Diese Plug-Ins werden in Plug-In-Stacks zusammengefasst. Ein Plug-In-Stack ist eine Serie von Plug-Ins, die mit einem Input-Plug-In beginnt, ein, kein oder mehrere Filter-Plug-Ins aufweist und ein Output-Plug-In besitzt.

Der Ulogd-Daemon unterstützt aktuell drei Input-Plug-Ins:

» Aus Kompatibilitätsgründen unterstützt er noch das alte `ulogd_inppkt_ULOG.so`-Plug-In, das IPv4-Pakete lesen und analysieren kann. Dieses emuliert auch das Verhalten des Ulogd 1.x.

» `ulogd_inppkt_NFLOG.so` für die Verarbeitung von Paketen

» `ulogd_inpflow_NFCT.so` für die Verarbeitung von Flows

Als Filter stehen zur Verfügung:

» `ulogd_raw2packet_BASE.so`
Dieses Plug-In interpretiert Adressen, Header etc.

» `ulogd_filter_PWSNIFF.so`
Dieses Plug-In interpretiert und protokolliert Klartext-Kennwörter in FTP, Telnet etc.

» `ulogd_filter_IFINDEX.so`
Dieses Plug-In interpretiert den Namen der Netzwerkkarte.

» `ulogd_LOCAL.so`
Dieses Plug-In stellt die lokale Zeit und Rechnernamen zur Verfügung.

» `ulogd_filter_HWHDR.so`
Dieses Plug-In interpretiert die MAC-Adresse eines Paketes.

» `ulogd_filter_IP2BIN.so`
Dieses Plug-In interpretiert die IP-Adressen in binärer Form für die Verwendung in Datenbanken.

» `ulogd_filter_IP2STR.so`
Dieses Plug-In konvertiert die IP-Adressen in Strings.

» `ulogd_filter_PRINTFLOW.so`
Dieses Plug-In konvertiert die Flowdaten in eine für Menschen lesbare Form.

» `ulogd_filter_PRINTPKT.so`
Dieses Plug-In konvertiert die Paketdaten in eine für Menschen lesbare Form.

» `ulogd_filter_MARK.so`
Dieses Plug-In filtert nur die Pakete mit einer bestimmten Markierung.

Genauso stehen viele verschiedene Output-Plug-Ins zur Verfügung, mit denen Sie die Pakete in verschiedenen Formaten protokollieren können. Sie können zwischen den folgenden Output-Plug-Ins wählen:

» MySQL

» PGSQL

» PCAP

» SQLITE3

» LOGEMU (funktioniert ähnlich wie die Syslog-Protokollierung)

» OPRINT (einfache Protokollierung in einer Datei für das Debugging)

» SYSLOG (protokolliert via Syslogd)

Sie müssen mindestens ein Input- und ein Output-Plug-In wählen. Außerdem müssen Sie in der Konfigurationsdatei die folgenden Parameter setzen:

» `nlgroup=<gruppe>`: Hiermit wählen Sie die Gruppe, an die sich dieser Ulogd-Prozess binden soll. Jeder Ulogd-Prozess kann sich nur an eine Gruppe binden. Diese Zahl muss mit der Angabe in der Option `--ulog-nlgroup` des `ULOG`-Targets übereinstimmen. Wenn Sie verschiedene Gruppen in Ihren Regeln verwenden, benötigen Sie für jede Gruppe einen eigenen Ulogd-Prozess.

» `logfile=<datei>`: Dies ist die Protokolldatei des Ulogd-Daemons. Hier protokolliert der Daemon Fehler und Warnungen. Durch Angabe von `syslog` oder `stdout` protokolliert der Ulogd diese Informationen über den Syslogd oder auf der Standardausgabe.

» `loglevel=<level>`: Hiermit stellen Sie den Detailgrad der Ulogd-eigenen Protokollierung in dem `logfile` ein. Die Level sind in der Beispielkonfigurationsdatei angegeben.

» `plugin=<pfad/plugin>`: Mit diesem Parameter laden Sie die Plug-Ins, die Sie verwenden möchten. Sie können diesen Parameter mehrfach verwenden. Die einzelnen Plug-Ins benötigen teilweise zusätzliche Konfigurationsparameter (z. B. MySQL).

» `rmem=<größe>`: Mit diesem Parameter stellen Sie die Größe des Ulogd-Netlink-Socket-Empfangspuffers ein. Dieser muss mindestens der Größe des Netlink-Puffers des `ipt_ULOG`-Moduls entsprechen. Wenn Sie diesen mit der Option `nlbufsiz=größe` gesetzt haben, müssen Sie auch diesen Wert anpassen.

» `bufsize=<größe>`: Dies ist der eigentliche Ulogd-Empfangspuffer. Er muss mindestens die Größe des `rmem`-Puffers haben.

Einzelne Plug-Ins verfügen über zusätzliche Konfigurationsparameter. Bevor diese in eigenen Abschnitten besprochen werden, möchte ich kurz die Optionen erläutern, die Sie bei dem Start des Ulogd verwenden können:

» `-d, --daemon`: Daemon-Modus. Hiermit wechselt der Prozess in den Hintergrund.

» `-V, --version`: Anzeige der Version.

» `-h, --help`: Anzeige der Hilfe.

» `-c, --config <datei>`: Hiermit wählen Sie eine alternative Konfigurationsdatei aus. Die Default-Datei ist `/etc/ulogd.conf`. Diese Option ist wichtig, wenn Sie mehrere Instanzen des Ulogd auf Ihrem System betreiben möchten. Das ist immer dann der Fall, wenn Sie in Ihren Regeln das `ULOG`-Target mit verschiedenen Netlink-Gruppen verwenden, um die Ausgabe in unterschiedliche Protokolle zu schreiben.

24.3.1 [OPRINT]

Dies ist ein sehr einfaches Ausgabemodul, das nur für Debugzwecke eingesetzt werden kann. Das Modul benötigt als einzigen Parameter den Dateinamen für die Ausgabe:

```
[OPRINT]
dumpfile=<datei>
```

24.3.2 [LOGEMU]

Dieses Modul protokolliert in eine Datei und emuliert dabei das Format der Syslog-Protokollierung ähnlich dem `LOG`-Target. Die Protokollierung erfolgt aber ohne Umweg über den Syslog direkt in die Datei. Sie müssen den Dateinamen angeben und können mit dem Parameter `sync` definieren, ob das Protokoll synchron (1) geschrieben werden soll (Default 0, asynchron). Dann tauchen neue Meldungen in dem Protokoll sofort auf, gehen aber zulasten der Geschwindigkeit.

```
[LOGEMU]
file=<datei>
sync=0|1
```

24.3.3 [MYSQL]

Hiermit protokollieren Sie in eine MySQL-Datenbank. Damit Sie dieses Plug-In nutzen können, muss der Ulogd mit MySQL-Unterstützung übersetzt werden (`./configure --with-mysql`). Welche Informationen von diesem Plug-In protokolliert werden, hängt von Ihrer MySQL-Tabelle ab. Dieses Plug-In liest die verfügbaren Spalten der MySQL-Tabelle und vergleicht deren Namen mit den zur Verfügung stehenden Informationen. Als Beispiel für eine derartige Tabelle und die zu verwendenden Spalten-Typen können Sie auf die Datei `doc/mysql.table` zurückgreifen.

Wenn Sie an bestimmten Informationen nicht interessiert sind, können Sie durch Entfernen der Spalte dafür sorgen, dass dieses Plug-In die Daten auch nicht protokolliert. Dadurch wird Ihre Datenbank schlanker. Wenn Sie die IP-Adresse als Zeichenkette in der Datenbank ablegen möchten, können Sie auch die Tabelle aus der Datei `doc/mysql.table.ipaddr-as-string` verwenden. Dann müssen Sie aber auch Ulogd mit der Option `--with-mysql-log-ip-as-string` bei dem `./configure`-Aufruf übersetzen.

Das Modul benötigt die folgenden zusätzlichen Angaben:

- `table=<tabelle>`: den Namen der SQL-Tabelle
- `ldb=<datenbank>`: den Namen der SQL-Datenbank
- `host=<mysql_rechner>`: den Rechner, auf dem der MySQL-Server läuft. Damit können Sie die Datenbank auf einem anderen Rechner unterbringen. Denken Sie nur daran, dass Ihre Firewall den Zugriff auf diesen Rechner erlauben muss.
- `port=<mysql_port>`: den Port der MySQL-Datenbank
- `user=<mysql_user>`: den Benutzer, der für die Anmeldung an der Datenbank verwendet wird. Verwenden Sie aus Sicherheitsgründen bitte nicht das root-Konto.
- `pass=<mysql_password>`: das Kennwort für die Anmeldung an der Datenbank

24.3.4 [PGSQL]

Dieses Output-Plug-In arbeitet ähnlich wie das MySQL-Plug-In. Damit das Plug-In zur Verfügung steht, müssen Sie Ulogd mit der Option `--with-pgsql` bei der Übersetzung konfiguriert

haben. Es verfügt über die gleichen Parameter, und Sie finden in der Ulogd-Distribution auch eine Beispiel-Tabelle `doc/pgsql.table`. Je nach PostgreSQL-Version müssen Sie jedoch einen zusätzlichen Parameter angeben: `schema=<schema>`. PostgreSQL ab Version 7.3 unterstützt diese Schemas. Damit können Sie datenbankübergreifende Views erzeugen. (Für Hintergrundinformationen über Schemas lesen Sie bitte auf `http://sql-info.de/postgresql/schemas.html` nach.) Wenn Sie PostgreSQL 7.3 oder eine neuere Version einsetzen, müssen Sie zusätzlich ein Schema angeben. PostgreSQL stellt für diese Zwecke das Default-Schema `public` zur Verfügung.

```
[PGSQL]
table="ulog"
schema="public"
pass="g3h31m"
user="ulog"
db="ulogd"
host="localhost"
```

24.3.5 [PCAP]

Dieses Output-Plug-In protokolliert das Paket im Libpcap-Format. Damit können Sie später die Protokolldatei in Ethereal oder Tcpdump öffnen und analysieren. Dies kann eine spätere forensische Analyse des Angriffs stark erleichtern. Dazu sollte dann aber von dem ULOG-Target auch das gesamte Paket protokolliert werden.

Das Plug-In benötigt mindestens die Angabe der Pcap-Datei. Zusätzlich können Sie mit dem Parameter `sync` angeben, ob die Datei synchron (1) oder asynchron (0, Default) geschrieben werden soll.

```
[PCAP]
file="/var/log/ulogd/ulogd.pcap"
sync=1
```

24.3.6 [SQLITE3]

SQLite (`http://www.sqlite.org/`) ist eine kleine C-Bibliothek, die eine komplette, in andere Programme einbettbare SQL-Datenbank ohne zusätzliche Konfiguration darstellt. Sie benötigen keinen Dienst und müssen keinen Client oder Server konfigurieren. In vielen Umgebungen ist SQLite schneller als ein komplettes Datenbank-Managementsystem (DBMS). Um SQLite zu verwenden, müssen Sie Ulogd mit der Option `--with-sqlite` bei der Übersetzung konfiguriert haben.

Für die Erzeugung der Datenbank finden Sie in dem `doc/`-Verzeichnis der Distribution auch eine Datei `sqlite3.table`. Die einzigen Parameter, die Sie angeben müssen, sind der Pfad der Datenbank, der Name der Tabelle und die Größe des SQLite-Puffers.

```
[SQLITE3]
table="ulog"
db="/path/to/sqlite/db"
buffer=200
```

24.3.7 [SYSLOG]

Das Syslog-Plug-In protokolliert über den Syslog-Dienst. Damit sind die Protokollmeldungen identisch zu den Meldungen, die normalerweise das `LOG`-Target erzeugt. Sie müssen hier lediglich die Priorität der Meldung (Level, Priority) und die protokollierende Quelle (Facility) angeben.

```
[SYSLOG]
facility=LOG_KERN
level=LOG_INFO
```

24.4 Der Specter-Daemon

Der Specter-Daemon (`http://joker.linuxstuff.pl/specter/`) von Michal Kwiatkowski basiert auf dem Ulogd, wurde aber um einige Funktionen erweitert. Er wird genauso wie Ulogd unter der GPL-Lizenz vertrieben.

Zu den wesentlichen Unterschieden zum Ulogd zählen:

- Die Konfigurationsdatei hat ein anderes Format. Das Skript `ulogd2specter.pl` wandelt die Konfigurationsdatei `ulogd.conf` in die Datei `specter.conf` um.
- Die Gruppierung mehrerer Netlink-Gruppen ist möglich. Sie müssen nicht für jede Netlink-Gruppe einen eigenen Prozess starten.
- Eine Gruppierung der Meldungen ist auch in Abhängigkeit von der Netfilter-Firewall-Markierung möglich.
- Zwei weitere Output-Plug-Ins, `EXEC` und `HTTP`, können externe Befehle aufrufen und HTTP-Verkehr parsen.
- Die Output-Plug-Ins `MYSQL` und `PGSQL` unterstützen SSL für die Datenbankverbindung.
- Die Output-Plug-Ins `SYSLOG` und `LOGEMU` unterstützen die Protokollierung der IP- und TCP-Optionen, der TCP-Sequenznummern und des MAC-Headers.

Specter wird leider nicht mehr weiterentwickelt. Die aktuellste Version 1.4 wurde im Juli 2005 veröffentlicht. Da es ansonsten in der Anwendung dem Ulogd gleicht und die Dokumentation sehr ausführlich ist, zeige ich hier nur eine Bespielkonfiguration mit einer kurzen Beschreibung:

```
plugins {
          BASE /lib/specter/specter_BASE.so
          MYSQL /lib/specter/specter_MYSQL.so
        }

13 {
     :BASE
     :MYSQL
     db mydb
     host localhost
     user username
     pass password
     table ext_attacks
   }

15 { :BASE
     :MYSQL
     db mydb
     host localhost
     user username
     pass password
     table int_attacks
   }
```

In dem Block `plugins` definieren Sie die zu ladenden Plug-Ins. Diese werden dann weiter unten konfiguriert. Die weiteren Blöcke beginnen mit der Netlink-Gruppe. Anschließend definieren Sie die zu verwendenden Plug-Ins und geben zusätzliche Parameter an.

Wenn Sie nicht die Netlink-Gruppe, sondern die Firewall-Markierung für die Aufteilung der Meldungen verwenden möchten, müssen Sie zuvor noch einen Block `global` definieren:

```
global {
         grouping nfmark
         nlgroup 1
       }
```

Nun nimmt Specter die Meldungen der Netlink-Gruppe 1 an und kann diese entsprechend der Firewall-Markierung aufteilen. Die Nummern vor jedem Block entsprechen nun nicht mehr der Netlink-Gruppe, sondern dieser Firewall-Markierung.

25. Conntrack-Tools

Mit den Conntrack-Tools, die inzwischen Bestandteil der meisten Linux-Distributionen sind, können Sie die Connection-Tracking-Tabelle direkt verwalten und modifizieren. Bestandteil des Paketes sind zwei Binärprogramme:

» `conntrack`: Hiermit können Sie die Connection-Tracking-Tabelle anzeigen, durchsuchen, einzelne Einträge hinzufügen, löschen oder aktualisieren und auch die gesamte Tabelle löschen.

» `conntrackd`: Dieser Dienst kann die Connection-Tracking-Tabelle mit einem zweiten System synchronisieren. Dies wird für eine hochverfügbare Firewall benötigt und daher in einem eigenen Kapitel (siehe Kapitel 27) besprochen. Dieses Kapitel beschränkt sich auf den Befehl `conntrack`.

INFO

Debian-Distributionen teilen das Paket auf zwei Pakete auf: `conntrack` und `conntrackd`. Alle anderen mir bekannten Distributionen unterstützen ein Paket `conntrack-tools`.

Der Befehl `conntrack` kann zwei Zustandstabellen bearbeiten:

» `conntrack`: Dies ist die Default-Tabelle. Sie enthält sämtliche dem System bekannten und erlaubten Verbindungen.

» `expect`: Diese Tabelle enthält die Verbindungen, die von den Connection-Tracking-Hilfsmodulen erkannt wurden. Protokolle wie FTP, SIP und IRC können zusätzliche Verbindungen dynamisch aushandeln, die von dem Connection-Tracking-System als `RELATED` erkannt werden. Diese werden in dieser Tabelle verwaltet.

Der Befehl `conntrack` unterstützt die folgenden Kommandos:

» `-L`, `--dump`: Hiermit zeigen Sie die aktuellen Verbindungen an. Dies entspricht einem `cat /proc/net/nf_conntrack`.

» `-G`, `--get`: Hiermit suchen Sie nach einem spezifischen Eintrag in der Tabelle:

```
# conntrack -G -s 192.168.255.105 -d 87.106.54.221 -p tcp -sport 53882
        -dport 993
tcp 6 326249 ESTABLISHED src=192.168.255.105 dst=87.106.54.221 sport ↵
        =53882 dport=993 packets=952 bytes=108235 src=87.106.54.221  ↵
        dst=192.168.255.105
sport=993 dport=53882 packets=910 bytes=212588 [ASSURED] mark=0  ↵
        secmark=0 use=2
conntrack v0.9.14 (conntrack-tools): 1 flow entries have been shown.
```

» `-D`, `--delete`: Hiermit löschen Sie einen Eintrag in der Tabelle:

```
# conntrack -D -s 192.168.255.105 -d 122.176.194.40 -p tcp -sport 35968
        -dport 38256
tcp 6 326114 ESTABLISHED src=192.168.255.105 dst=122.176.194.40 sport ↵
        =35968 dport=38256 packets=50 bytes=4223 src=122.176.194.40  ↵
        dst=192.168.255.105 sport=38256 dport=35968 packets=38 bytes ↵
        =3337 [ASSURED] mark=0 secmark=0 use=2
conntrack v0.9.14 (conntrack-tools): 1 flow entries have been deleted.
```

» `-I`, `--create`: Hiermit erzeugen Sie einen neuen Eintrag. Beispiel:

```
# conntrack -I -s 192.168.255.105 -d 122.176.194.40 -p tcp -sport 35968
        -dport 35968 -timeout 6000 -state ESTABLISHED
conntrack v0.9.14 (conntrack-tools): 1 flow entries have been created.
```

» `-U`, `--update`: Hiermit können Sie einen Eintrag modifizieren.

```
# conntrack -U -s 192.168.255.105 -d 122.176.194.40 -p tcp -sport 35968
        -dport 35968 -timeout 60 -state CLOSE
tcp 6 60 CLOSE src=192.168.255.105 dst=122.176.194.40 sport=35968  ↵
        dport=35968 packets=0 bytes=0 [UNREPLIED] src=122.176.194.40  ↵
        dst=192.168.255.105 sport=35968 dport=35968 packets=0 bytes=0  ↵
        mark=0 secmark=0 use=2
conntrack v0.9.14 (conntrack-tools): 1 flow entries have been updated
```

» `-E`, `--event`: Hiermit können Sie sich in Echtzeit die Änderungen an den Tabellen anzeigen lassen:

```
# conntrack -E
[DESTROY] tcp 6 src=192.168.255.105 dst=77.87.229.48 sport=53127 dport ↵
        =80 packets=170 bytes=7303 src=77.87.229.48 dst ↵
        =192.168.255.105 sport=80 dport=53127 packets=214 bytes=304568 ↵
         [ASSURED]
[DESTROY] tcp 6 src=192.168.255.105 dst=122.176.194.40 sport=35968  ↵
        dport=35968 packets=0 bytes=0 [UNREPLIED] src=122.176.194.40  ↵
        dst=192.168.255.105 sport=35968 dport=35968 packets=0 bytes=0
[NEW] icmp 1 30 src=192.168.255.105 dst=192.168.0.5 type=8 code=0 id ↵
        =1353 [UNREPLIED] src=192.168.0.5 dst=192.168.255.105 type=0  ↵
        code=0 id=1353
[NEW] icmp 1 30 src=192.168.255.105 dst=192.168.255.1 type=8 code=0 id ↵
        =1355 [UNREPLIED] src=192.168.255.1 dst=192.168.255.105 type=0 ↵
         code=0 id=1355
[UPDATE] icmp 1 30 src=192.168.255.105 dst=192.168.255.1 type=8 code=0 ↵
         id=1355 src=192.168.255.1 dst=192.168.255.105 type=0 code=0  ↵
        id=1355
```

» `-F`, `--flush`: Hiermit löschen Sie die gesamte Tabelle.

```
# conntrack -F
conntrack v0.9.14 (conntrack-tools): connection tracking table has ↵
        been emptied.
```

» `-C`, `--count`: Dieser Befehl zählt die Einträge in der Tabelle.

» `-S`, `--stats`: Hiermit zeigen Sie weitere statistische Informationen an:

```
# conntrack -S
entries          11
searched     164780
found      15252009
new           76278
invalid       14000
ignore        19110
delete        72477
delete_list   49938
insert        53738
insert_failed     0
drop              0
early_drop        0
icmp_error     9931
expect_new        0
expect_create     0
expect_delete     0
search_restart    0
```

Bei einzelnen Befehlen können Sie weitere Parameter angeben. Mit der Option `-z` (`--zero`) löschen Sie die Zähler nach der Ausgabe der Tabelle mit `-L`. Mit der Option `-o` (`--output`) können Sie das Ausgabeformat bestimmen. Möglich sind `extended` (Format von `nf_conntrack`), `xml`, `timestamp` und `id`. Als Default wird das Format von `ip_conntrack` genutzt. Das Ausgabeformat `timestamp` ist bei der Echtzeit-Ereignisanzeige sinnvoll. Mit der Option `-e` (`--event-mask`) können Sie in der Echtzeitausgabe der Tabellenereignisse die Ereignisse filtern: `ALL`, `NEW`, `UPDATES`, `DESTROY` etc.

Um einzelne Einträge anzuzeigen, zu suchen, zu löschen oder hinzuzufügen, können diese mit weiteren Filterparametern spezifiziert werden.

Diese Parameter können zum Beispiel mit dem Befehl `-L` genutzt werden, um nur entsprechende Verbindungen anzuzeigen. Für die komplette Liste aller Filterparameter möchte ich Sie auf die Manpage verweisen. Im Folgenden stelle ich nur einige Beispiele vor:

» Um alle IPv6-Verbindungen anzuzeigen, nutzen Sie:

```
# conntrack -L -f ipv6
[sudo] password for spenneb:
tcp 6 431990 ESTABLISHED src=::1 dst=::1 sport=47978 dport=22 packets ↵
        =8 bytes=1621 src=::1 dst=::1 sport=22 dport=47978 packets=7  ↵
        bytes=2189 [ASSURED] mark=0 secmark=0 use=2
conntrack v0.9.14 (conntrack-tools): 1 flow entries have been shown.
```

» Um alle SNAT-Verbindungen anzuzeigen, können Sie den folgenden Ausdruck verwenden:

```
# conntrack -L -src-nat
tcp 6 12 TIME_WAIT src=192.168.255.202 dst=91.190.225.72 sport=4577  ↵
        dport=5667 packets=6 bytes=1040 src=91.190.225.72 dst ↵
        =192.168.253.2 sport=5667 dport=4577 packets=4 bytes=348 [ ↵
        ASSURED] mark=0 secmark=0 use=1
tcp 6 60 TIME_WAIT src=192.168.255.202 dst=91.190.225.72 sport=4585  ↵
        dport=5667 packets=6 bytes=1040 src=91.190.225.72 dst ↵
        =192.168.253.2 sport=5667 dport=4585 packets=4 bytes=348 [ ↵
        ASSURED] mark=0 secmark=0 use=1
```

» Um alle Verbindungen einer IP-Adresse zu löschen, nutzen Sie:

```
# conntrack -D -s 192.168.255.202
```

26. Ipset

Das Kommando `ipset` löst den alten Befehl `ippool` ab. Viele Distributionen bringen diesen Befehl in einem eigenen Paket `xtables-addons` mit. Ipset erlaubt es Ihnen, Gruppen von IP-Adressen, Port-Nummern oder anderen Informationen anzulegen, mit Werten zu füllen und in Ihren Regeln für einen Test zu nutzen. Dabei ist besonders interessant, dass Sie diese Gruppen dynamisch in Ihren Regeln verändern können. Die Gruppen können auch untereinander verknüpft werden.

Da diese Funktion mächtige Möglichkeiten bietet und in einem eigenen Befehl implementiert wurde, widme ich ihr ein eigenes Kapitel.

Der Befehl `ipset` existiert in zwei Versionen. Die Version 4.x unterstützt nur IPv4-Adressen und -Netze, während ab der Version 5 (aktuell war im März 2011 die Version 6.2) auch IPv6-Adressen unterstützt werden. Da die meisten Distributionen aktuell noch die Version 4 ausliefern, betrachte ich zunächst diese Variante.

Welche Version Ihre Distribution nutzt, erkennen Sie mit:

```
# ipset -v
ipset v4.4, protocol version 4.
Kernel module protocol version 4.
```

26.1 ipset Version 4 und iptables

Wenn Ihre Distribution den Befehl `ipset` unterstützt, stehen Ihnen für die Verwendung des `iptables`-Befehls ein neues Target `SET` mit `--add-set|--del-set` und ein neuer Match `set` (`-m set --set <set> src|dest`) zur Verfügung. Dieser Test erlaubt es Ihnen, die Mitgliedschaft des Absenders oder des Ziels des Pakets in einer Gruppe zu testen. Mit dem Target `SET` können Sie einen Absender oder ein Ziel zu einer Gruppe hinzufügen. Welche Informationen des Absenders oder des Ziels zu einer Gruppe hinzugefügt werden, hängt von der Gruppenart ab. Die verschiedenen Gruppen werden im nächsten Abschnitt genauer besprochen.

Der Befehl `ipset` gibt Ihnen die Möglichkeit, diese Gruppen unabhängig von dem Iptables-Kommando zu definieren und Mitglieder hinzuzufügen oder zu entfernen.

Die Verwendung wird am einfachsten anhand eines Beispiels deutlich. Stellen Sie sich vor, dass Sie in Ihrem Netzwerk fünf Systeme besitzen, die als Administrationsrechner per Secure-Shell auf die Firewall zugreifen können sollen. Anstatt für jeden dieser Rechner eine eigene

Regel zu definieren, können Sie folgendes Konstrukt verwenden:

```
ipset -N admin ipmap --network 192.168.0.0/24
ipset -A admin 192.168.0.5
ipset -A admin 192.168.0.31
ipset -A admin 192.168.0.57
ipset -A admin 192.168.0.91
ipset -A admin 192.168.0.211

iptables -A INPUT -p tcp --dport 22 -m set --set admin src -m state -- ↵
        state NEW -j ACCEPT
```

In der ersten Zeile wird zunächst eine neue Gruppe vom Typ `ipmap` erzeugt. Dies ist die einfachste Gruppe, die von dem Befehl Ipset zur Verfügung gestellt wird. Hier wird eine Gruppe erzeugt, in der anschließend die IP-Adressen für das Netzwerk 192.168.0.0/24 verwaltet werden können. Anschließend fügen die nächsten fünf Zeilen fünf verschiedene IP-Adressen der Gruppe hinzu. Die `iptables`-Zeile prüft schließlich, ob der Zugriff erlaubt ist. Hierzu überprüft diese Zeile, ob der Absender des Pakets (`src`) in der Gruppe `admin` ist.

Eine besondere, sehr interessante zusätzliche Funktion erlaubt auch die Verknüpfung einzelner Gruppen untereinander. Vielleicht möchten Sie auch, dass die Admin-Rechner auf alle Ports der Firewall zugreifen dürfen. Lediglich ein bestimmter Rechner (192.168.0.211) darf nur auf die beiden Ports 22 und 443 zugreifen. Auch diese Aufgabe können Sie mithilfe von Sets lösen. Hierzu erzeugen Sie zunächst eine zweite Gruppe:

```
ipset -N adminports portmap --from 1 --to 1023
ipset -A adminports 22
ipset -A adminports 443
```

Nun binden Sie diese Gruppe an den Eintrag für die IP-Adresse 192.168.0.211:

```
ipset -B admin 192.168.0.211 -b adminports
```

Jetzt müssen Sie nur noch die Iptables-Regel ändern:

```
iptables -A INPUT -p tcp -m set --set admin src,dst -m state --state NEW ↵
        -j ACCEPT
```

Diese Regel prüft nun, ob der Absender des Pakets in der Gruppe `admin` vorhanden ist. Falls eine Bindung an die IP-Adresse erfolgt ist, prüft die Regel zusätzlich, ob die gebundenen Informationen mit dem Ziel (`dst`) übereinstimmen. Ist keine Bindung vorhanden, wird keine zusätzliche Prüfung durchgeführt.

Diese Informationen sollen zunächst als Einführung genügen. Die nächsten Abschnitte stellen die Gruppen und die genaue Syntax der Befehle vor.

26.1.1 Die Ipset-4-Typen

Mit dem Ipset-Befehl können Sie viele verschiedene Arten von Gruppen verwalten. Die Art der Gruppe bestimmt die in der Gruppe gespeicherten Informationen.

Ipset unterstützt die folgenden Gruppen:

- `ipmap`: Speichert IP-Adressen oder Netzwerke identischer Größe.
- `portmap`: Speichert Portnummern.
- `macipmap`: Speichert MAC/IP-Adresspaare.
- `iphash`: Speichert IP-Adressen oder Netzwerke identischer Größe.
- `nethash`: Speichert Netzwerke unterschiedlicher Größe.
- `ipporthash`: Speichert IP/Port-Paare.
- `ipportiphash`: Speichert IP/Port/IP-Tripel.
- `ipportnethash`: Speichert IP/Port/Netzwerk-Tripel.
- `iptree`: Speichert IP-Adressen mit einer begrenzten Lebensdauer.
- `iptreemap`: Speichert IP-Adressen und Netzwerke mit einer begrenzten Lebensdauer als Bitmap.
- `setlist`: Speichert mehrere Ipset-Gruppen in einer Liste.

Diese verschiedenen Gruppen werden im Folgenden genauer besprochen und mit Beispielen vorgestellt.

26.1.2 ipmap

Die `ipmap`-Gruppe ist eine Bitmap, bei der jede IP-Adresse durch ein einzelnes Bit repräsentiert wird. Sie eignet sich daher sehr gut zur Speicherung nahe beieinander liegender IP-Adressen. Die maximale Größe der Bitmap beträgt 65.536 Adressen und entspricht damit einem Klasse-B-Netzwerk. Eine Bitmap dieser Größe belegt nur 8 Kbyte Arbeitsspeicher, da pro Adresse ein Bit benötigt wird. Die `ipmap`-Gruppe ist daher sehr speicherschonend und gleichzeitig sehr schnell, da direkt auf die Information, ob eine Adresse in der Gruppe vorhanden ist, zugegriffen werden kann.

Bei der Erzeugung einer neuen `ipmap`-Gruppe müssen Sie entweder das zu verwaltendene Netzwerk angeben (`--network <net/mask>`) oder die Start- und die Ziel-IP-Adresse definieren (`--from <ip> --to <ip>`).

Wenn Sie nicht IP-Adressen, sondern Netzwerke testen möchten, können Sie das auch mit dieser Gruppe bewerkstelligen. Dafür müssen die Netzwerke lediglich eine identische Größe aufweisen. Diese Größe geben Sie gleichzeitig mit der Option `--netmask <CIDR>` an. Um zum Beispiel eine Gruppe zu erzeugen, in der Sie alle IP-Adressen als Klasse-A-Netzwerke verwalten können, können Sie die folgende Zeile verwenden:

```
ipset -N badnetworks ipmap --network 0/0 --netmask 8
```

Die entstehende Bitmap ist 256 Netze groß. Sie können nun einzelne Netze hinzufügen oder entfernen. Dies erleichtert zum Beispiel eine Prüfung auf die noch nicht zugewiesenen und damit nicht erlaubten Netze sehr. Die IANA pflegt eine Liste sämtlicher IP-Adressen (`http://www.iana.org/assignments/ipv4-address-space`). Sie könnten alle hier als *Reserved* aufgeführten Netze dieser Liste hinzufügen und in der `raw`-Table bereits alle diese Pakete verwerfen. Hiermit schützen Sie sich vor Spoofing-Angriffen, bei denen der Angreifer diese IP-Adressen nutzt. Seit Anfang des Jahres 2011 sind sämtliche Netze zugewiesen worden, sodass mit dieser Methode nur noch die Class-D- und -E-Netze (224.0.0.0–255.0.0.0) erkannt und abgelehnt werden können.

portmap

Diese Gruppe ist in ihrem internen Aufbau mit der `ipmap` identisch. Es handelt sich ebenfalls um eine Bitmap mit einer maximalen Größe von 65.536 Ports. Daher können Sie sämtliche möglichen Ports in einer Bitmap speichern. Genau wie die `ipmap`-Gruppe ist auch diese Gruppe sehr schnell und benötigt nur wenig Speicherplatz. Für die tatsächliche Anwendung können Sie auch den Speicherverbrauch durch die Angabe der Größe weiter einschränken. Sie können den ersten und den letzten in der Gruppe zu verwaltenden Port mit `--from <port>` und `--to <port>` angeben.

Diese Gruppe ist besonders interessant, da Sie diese Gruppe wie alle anderen für Verknüpfungen der Gruppen untereinander nutzen können und so einzelne IP-Adressen aus der `ipmap`-Gruppe auf bestimmte Ports beschränken können.

macipmap

Diese Gruppe speichert nicht nur wie die Gruppe `ipmap` IP-Adressen, sondern die Kombination aus einer IP-Adresse und der MAC-Adresse. Die maximale Größe dieser Gruppe beträgt wie bei der `ipmap`-Gruppe 65.536 Adressen. Da jedoch zusätzlich die MAC-Adresse gespeichert werden muss, erhöht sich der Speicheraufwand pro IP-Adresse von 1 Bit auf 8 Byte. Bei 65.536 Adressen benötigen Sie also 512 Kbyte Arbeitsspeicher. Daher ist es sinnvoll, die Größe der Gruppe sinnvoll zu beschränken. Hierfür geben Sie entweder die Start- und End-IP-Adresse mit `--from <ip>` und `--to <ip>` an, oder Sie definieren das Netzwerk mit `--network <net/mask>`.

Sie können hiermit eine Datenbank mit allen IP-Adressen und den dazugehörigen MAC-Adressen in Ihrem Netzwerk aufbauen. Anschließend verwerfen Sie in der `raw`-Tabelle alle Pakete, deren Absender nicht in der Tabelle aufgeführt wird. So unterdrücken Sie wirkungsvoll jedes IP- und ARP-Spoofing in Ihrem Netzwerk. Speziell für diesen Zweck besitzt diese Gruppe auch noch die Option `--matchunset`, die Sie bei der Erzeugung der Gruppe angeben können. Damit wird jede IP-Adresse als Treffer bewertet, die nicht in der Gruppe gespeichert wurde, aber aufgrund des Bereichs in der Gruppe gespeichert werden könnte. Wenn Sie diese Pakete verwerfen, kommen nur noch Pakete durch Ihre Firewall, deren IP/MAC-Adresskombinationen Sie zuvor in der Gruppe definiert haben. Dies ist in großen Netzen oder in mit DHCP verwalteten Netzen jedoch leider nur schlecht umsetzbar.

Die MAC-Adresse ist nur sinnvoll als Absenderadresse auswertbar. Daher wird die MAC-Adresse auch immer als Source-Adresse von dem `set`-Match und dem `SET`-Target genutzt. Die Verwendung als Ziel-MAC-Adresse ist nicht möglich.

Um nun eine IP-Adresse der Gruppe hinzuzufügen, verwenden Sie die folgende Syntax:

```
ipset -N valid_ip_macs macipmap --network 192.168.0.0/24 --matchunset
ipset -A valid_ip_macs 192.168.0.100%00:50:56:C0:00:03
```

iphash

Es gibt insgesamt drei verschiedene Möglichkeiten, IP-Adressen in einer Gruppe zu speichern: `ipmap`, `iphash` und `iptree`. Während `ipmap` nahe beieinanderliegende IP-Adressen speicherschonend und schnell verwalten kann, ist `iphash` optimiert, um beliebige, zufällige IP-Adressen aus dem gesamten Adressraum in einer Gruppe zusammenzufassen. Das ist mit der `ipmap`-Gruppe nicht möglich. Die `ipmap`-Gruppe kann maximal einen Bereich von 65.536 benachbarten IP-Adressen verwalten.

Die `iphash`-Gruppe besitzt einige Optionen, mit denen Sie bei der Erzeugung der Gruppe ihr Verhalten beeinflussen können. Mit der Option `--hashsize <größe>` setzen Sie die initiale Größe des Hashs (1024). Wenn Sie einen neuen Eintrag hinzufügen, definieren Sie mit `--probes <versuche>`, wie viele Versuche für die Einfügung der neuen IP-Adresse durchgeführt werden sollen, bevor der Hash vergrößert wird (8). Mit `--resize <prozent>` geben Sie an, wie der Hash vergrößert werden soll (50 %). Wenn Sie keine Vergrößerung wünschen, definieren Sie hier 0.

Die Anzahl der Versuche (`--probes`) hat direkte Auswirkungen auf die Geschwindigkeit und die Größe des Hashs. Während kleine Werte (1–3) einen schnellen, aber großen Hash erzeugen, optimieren große Werte (6–10) die Größe des Hashs, erzeugen dabei aber einen langsameren Hash. Die Geschwindigkeit hängt linear von der Anzahl der Versuche ab.

Ähnlich wie bei der `ipmap`-Gruppe können Sie in dem `iphash` auch Netzwerke identischer Größe speichern. Dazu definieren Sie bei der Erzeugung des Hashs deren Größe mit `--netmask <CIDR>`.

nethash

Sie können Netzwerke in mehreren Gruppen verwalten. Sowohl die Gruppe `ipmap` als auch die Gruppe `iphash` bietet Ihnen die Möglichkeit, auch Netzwerke identischer Größe zu verwalten. Wenn Sie aber Netze unterschiedlicher Größe in einer gemeinsamen Gruppe verwalten möchten, müssen Sie den Typ `nethash` verwenden.

Dieser Typ verfügt über die identischen Optionen wie der `iphash`: `--hashsize <größe>`, `--probes <versuche>` und `--resize <prozent>`. Diese Optionen haben auch die gleiche Bedeutung wie bei dem `iphash`. Wenn Sie Netze der Gruppe hinzufügen, müssen Sie diese als `IP/CIDR-Mask` angeben.

Die Geschwindigkeit des Hashs ist linear abhängig von der Anzahl der Versuche und der Anzahl der verschiedenen Netzwerkmasken, die von den gespeicherten Netzen genutzt werden.

ipporthash

Der `ipporthash` ist eine besondere Variante des `ipmap`-Typs. Diese Gruppe speichert zusätzlich zu der IP-Adresse auch einen Port in der Syntax `IP%Port` ab. Insgesamt können Sie für 65.536 IP-Adressen beliebige Portnummern in dieser Gruppe speichern. Den IP-Adressbereich müssen Sie bei der Erzeugung der Gruppe angeben. Hierfür verwenden Sie wieder `--from <ip>` und `--to <ip>` oder `--network <ip/mask>`. Die anderen Parameter sind in ihrer Verwendung und Funktion mit den Parametern des `iphash` identisch.

iptree

Dieser Typ erlaubt die Generierung einer weiteren dritten Gruppe, in der Sie IP-Adressen speichern können. Im Gegensatz zu `ipmap` und `iphash` können Sie aber für die IP-Adressen eine Lebensdauer angeben (`--timeout <sekunden>`). Jede neu hinzugefügte IP-Adresse wird nur für diese Zeit in der Gruppe gespeichert. Entweder geben Sie die Lebensdauer mit der genannten Option bei der Erzeugung der Gruppe an, oder Sie geben für jede IP-Adresse beim Hinzufügen eine spezifische Lebensdauer an (`IP%<Lebensdauer>`).

Diese Funktion können Sie hervorragend nutzen, um Clients für eine bestimmte Zeit, zum Beispiel nach Bezahlung eines bestimmten Betrags, Zugriff auf ein Netz zu geben. Programmieren Sie nur ein Web-Interface, das die Bezahlung entgegennimmt und die IP-Adresse des Clients für die bezahlte Zeit der Gruppe hinzufügt.

Auch wenn Sie bestimmte Zugänge erst durch ein Web-Interface für eine bestimmte Zeit freischalten möchten, können Sie das hiermit erreichen. Das Web-Interface fügt einfach die entsprechende IP-Adresse zur Gruppe hinzu. Die Gruppe entfernt nach Ablauf der Lebensdauer die IP-Adresse selbstständig.

26.1.3 Das Kommando ipset in der Version 4

Mit dem Kommando `ipset` können Sie die Gruppen erzeugen, Einträge hinzufügen, entfernen und die Gruppen auch zerstören. Hierfür verwendet der `ipset`-Befehl ähnliche Optionen wie `iptables`:

» `-N <set> <type>`: Hiermit erzeugen Sie eine neue Gruppe. Sie müssen dabei mindestens den Namen und den Typ der Gruppe angeben. Bei einigen Typen müssen Sie zusätzlich weitere Optionen definieren.

```
ipset -N badnetworks ipmap --network 0/0 --netmask 8
```

» `-X [<set>]`: Hiermit löschen Sie eine oder alle Gruppen. Bevor die Gruppe jedoch gelöscht wird, werden alle Referenzen (Bindungen) von dieser Gruppe auf andere Gruppen gelöscht. Falls diese Gruppe selbst von einer anderen Gruppe referenziert wird, kann der Löschvorgang nicht durchgeführt werden.

- `-F [<set>]`: Hiermit löschen Sie alle Einträge in einer oder allen Gruppen.
- `-E <oldname> <newname>`: Hiermit benennen Sie eine Gruppe um.
- `-W <set> <set>`: Dies tauscht zwei Gruppen im Kernel aus. Damit können Sie sehr einfach eine Gruppe offline vorbereiten und für die Verwendung atomar austauschen. Es entsteht kein Zeitfenster, in dem während des Aufbaus der Gruppe diese nicht vollständig definiert ist.
- `-L [<set>]`: Diese Option zeigt den Inhalt einer oder aller Gruppen an. Mit der Option `-n` erzeugen Sie eine numerische Ausgabe. Die Option `-s` sortiert die Einträge.
- `-S`: Diese Option sichert die Einträge einer oder aller Gruppen auf der Standardausgabe. Zum Speichern können Sie die Ausgabe in eine Datei umleiten und später mit der Option `-R` wieder einlesen.
- `-R`: Hiermit stellen Sie die gespeicherten Informationen wieder her. Diese Option liest Ihre Befehle über die Standardeingabe. Wenn Sie die Datei nicht mit der Option `-S` erzeugen möchten, sondern dies manuell oder über ein anderes Skript erreichen, achten Sie bitte darauf, dass die Reihenfolge der Befehle wichtig ist. Als letzter Befehl ist auch jedes Mal der `COMMIT` anzugeben! Dieser Befehl beendet die Liste der Einträge. Zur Verdeutlichung speichern Sie einfach mit der Option `-S` die Einträge in einer Datei und nehmen diese als Beispiel.
- `-A <set> <IP>`: Diese Option fügt einen Eintrag einer Gruppe hinzu.
- `-D <set> <IP>`: Diese Option löscht einen Eintrag von einer Gruppe.
- `-T <set> <IP>`: Hiermit können Sie testen, ob ein Eintrag in einer Gruppe enthalten ist.
- `-B <set> <IP> --binding <set>`: Hiermit können Sie einen Eintrag in einer Gruppe zusätzlich an eine andere Gruppe binden. Damit dieser Eintrag später zutrifft, muss auch die gebundene Gruppe zutreffen.
- `-U <set> <IP>`: Hiermit lösen Sie die Bindung eines Eintrags.
- `-H`: Dies zeigt Ihnen die Hilfe an.

Immer wenn Sie eine administrative Änderung der Gruppen durchführen müssen, die eine Erzeugung einer neuen leeren Gruppe verlangt, die anschließend aufgefüllt wird, bietet es sich an, die Gruppe zu erzeugen und anschließend auszutauschen.

26.2 ipset Version 6 und ip6tables

Wenn Ihre Distribution den Befehl `ipset` unterstützt, stehen Ihnen für die Verwendung des `ip6tables`-Befehls ein neues Target `-j SET --add-set|--del-set` und ein neuer Match `set` (`-m set --set <set> src|dest`) zur Verfügung. Dieser Test erlaubt es Ihnen, die Mitgliedschaft des Absenders oder des Ziels des Pakets in einer Gruppe zu testen. Mit dem Target `SET` können Sie einen Absender oder ein Ziel zu einer Gruppe hinzufügen. Welche Informationen des Absenders oder des Ziels zu einer Gruppe hinzugefügt werden, hängt von der Gruppenart ab. Die verschiedenen Gruppen werden im übernächsten Abschnitt genauer besprochen.

Leider unterscheidet sich der Befehl `ipset` in der Version 6 von der älteren Version. Er verwendet eine komplett andere Syntax, die hier zunächst besprochen werden soll, bevor die neuen Gruppen erläutert werden.

Der neue Befehl ist jedoch abwärtskompatibel. Die alte Syntax wird auch noch unterstützt.

26.2.1 ipset Version 6: Syntax

Die allgemeine Syntax des `ipset`-Befehls lautet:

```
ipset [ OPTIONS ] COMMANDS [ COMMAND OPTIONS ]
```

Hierbei verwendet der Befehl als Kommando nicht wie die ältere Version `-A`, `-N`, etc., sondern komplette Worte:

create oder n Erzeugt eine neue Gruppe. Hierbei muss der Typ angegeben werden. Die möglichen Typen werden weiter unten aufgeführt.

add Dieser Befehl fügt einen Eintrag zu einer Gruppe hinzu.

del Dieser Befehl entfernt einen Eintrag aus einer Gruppe.

test Dieser Befehl prüft, ob ein Eintrag in einer Gruppe existiert.

destroy oder x Dieser Befehl entfernt eine oder alle Gruppen, wenn sie nicht mehr referenziert werden.

list Dieser Befehl zeigt eine oder alle Gruppen an.

save Dieser Befehl schreibt eine oder alle Gruppen auf die Standardausgabe. Das Format kann von `restore` wieder eingelesen werden.

restore Hiermit werden mittels `save` speicherte Gruppen wiederhergestellt.

flush Dieser Befehl löscht alle Einträge einer oder aller Gruppen.

rename oder e Hiermit benennen Sie eine Gruppe um.

swap oder w Hiermit tauschen Sie den Inhalt zweier Gruppen aus.

help Dies gibt eine allgemeine Hilfe aus.

version Hiermit zeigen Sie die Version an.

- Dieser Befehl wechselt in einen interaktiven Modus. Sie können die obigen Befehle dann über die Standardeingabe einlesen. Sie verlassen den Modus mit `quit`.

26.2.2 Die Ipset-6-Typen

Mit dem Ipset-Befehl in der Version 6 können Sie ebenfalls viele verschiedene Arten von Gruppen verwalten. Die Art der Gruppe bestimmt die in der Gruppe gespeicherten Informationen. Hierbei werden die Gruppentypen zusammengesetzt:

```
» TYPENAME := method:datatype[,datatype[,datatype]]
```

Der Ipset-Befehl in der Version 6 unterstützt nun die Methoden `bitmap`, `hash` und `list`. Als Datentypen stehen zur Verfügung: `ip`, `net`, `set`, `mac` und `port`.

Hieraus ergeben sich die folgenden Gruppen:

- `bitmap:ip`
- `bitmap:ip,mac`
- `bitmap:port`
- `hash:ip`
- `hash:net`
- `hash:ip,port`
- `hash:net,port`
- `hash:ip,port,ip`
- `hash:ip,port,net`
- `list:set`

Diese Gruppen entsprechen im Wesentlichen den schon früher verfügbaren Gruppen (siehe Abschnitt 26.1.1). Einige Besonderheiten und Unterschiede existieren jedoch. Diese will ich nun erläutern.

bitmap:ip

Diese Gruppe kann nur IPv4-Adressen oder -Netzwerke in einer Bitmap speichern. Sie entspricht daher der alten Gruppe `ipmap`. Ein Beispiel:

```
ipset create admin bitmap:ip range 192.168.0.0/24
ipset add admin 192.168.0.5
ipset add admin 192.168.0.6
```

bitmap:ip,mac

Auch diese Gruppe kann nur IPv4-Adressen in Kombination mit ihren MAC-Adressen speichern. Ein Beispiel:

```
ipset create admin bitmap:ip,mac range 192.168.0.0/24
ipset add admin 192.168.0.5,00:21:70:B4:03:17
```

bitmap:port

Diese Gruppe speichert Portnummern. Auch hier soll ein Beispiel die Verwendung verdeutlichen:

```
ipset create webports bitmap:port range 0-65535
ipset add webports 80
ipset add webports 443
ipset add webports 3128
ipset add webports 8080
```

hash:ip

Die Hash-Gruppentypen können auch IPv6-Adressen und -Netzwerke speichern. Hierzu müssen Sie sich jedoch entscheiden, ob Sie IPv4- oder IPv6-Adressen in der Gruppe nutzen möchten. Bei der Erzeugung der Gruppe geben Sie die Adressfamilie mit der Option `family` an. Mit `inet` erzeugen Sie eine IPv4-Gruppe, während `inet6` eine IPv6-Gruppe erzeugt. Unterbleibt die Angabe, wird per Default eine IPv4-Gruppe erzeugt.

Ein Beispiel:

```
# ipset create localip-v6 hash:ip family inet6
# ipset add localip-v6 fe80::221:70ff:feb4:317
# ipset list localip-v6
Name: localip-v6
Type: hash:ip
Header: family inet6 hashsize 1024 maxelem 65536
Size in memory: 9348
References: 0
Members:
fe80::221:70ff:feb4:317
```

Alle Hash-Gruppentypen verfügen über die folgenden Optionen:

» `hashsize`: Dies ist die Größe des Hashes. Der Defaultwert beträgt 1024. Es muss sich um eine Potenz von 2 handeln.

» `maxelem`: Dies ist die maximale Zahl von Elementen, die in der Gruppe gespeichert werden. Der Default beträgt 65536.

hash:net

Analog zum `hash:ip` speichert diese Gruppe ganze Netze. Auch hier können IPv6-Netze gespeichert werden.

hash:ip,port

Diese Gruppe verwaltet IP-Adressen und Port-Paare. Während in der älteren Gruppe `ipporthash` die Angabe eines Protokolls unnötig war, verlangt diese Gruppe zwingend die Angabe des Transportprotokolls: `tcp`, `udp`, `sctp` oder `udplite`. Wird kein Protokoll angegeben, verwendet der Hash per Default TCP! Alternativ können auch andere Protokolle verwendet werden. Da diese jedoch keine Ports unterstützen, verwenden Sie hier die Portnummer 0. Um auch ICMP-Informationen speichern zu können, können Sie bei Verwendung der Protokolle `icmp` oder `icmpv6` den ICMP-Typ und -Code angeben.

```
ipset create ping hash:ip,port family inet6
ipset add ping fe80::221:70ff:feb4:317,icmpv6:echo-request
```

hash:net,port

Diese Gruppe verwaltet Netzwerke und Port-Paare. Während in der älteren Gruppe `ipport-hash` die Angabe eines Protokolls unnötig war, verlangt diese Gruppe zwingend die Angabe des Transportprotokolls: `tcp`, `udp`, `sctp` oder `udplite`. Alternativ können auch andere Protokolle verwendet werden. Da diese jedoch keine Ports unterstützen, verwenden Sie hier die Portnummer 0. Um auch ICMP-Informationen speichern zu können, können Sie bei Verwendung der Protokolle `icmp` oder `icmpv6` den ICMP-Typ und -Code angeben.

```
ipset create routing hash:net,port
ipset add routing 192.168.0.0/24,vrrp:0
ipset add routing 192.168.0.0/24,igp:0
```

hash:ip,port,ip und hash:ip,port,net

Diese beiden Gruppen speichern das Triplett aus IP-Adresse, Port und IP-Adresse oder Netzwerk. Hierbei müssen Sie ebenfalls für den Port zwingend das Protokoll angeben. Beide Gruppen können sowohl IPv4- als auch IPv6-Adressen speichern.

Allgemeine Optionen

Alle Gruppen verfügen über die zusätzliche Option `timeout`, mit der Sie bei dem Hinzufügen eines Eintrages seine Lebensdauer definieren können. Sie können diesen Parameter auch bei der Erzeugung einer Gruppe angeben. Dann gilt die Angabe als Default für alle Einträge.

27. Hochverfügbare Firewalls

Wenn Sie verschiedene Firewalls geplant und implementiert haben, werden Sie irgendwann an den Punkt geraten, wo Ihre erste Firewall hochverfügbar sein soll. Das bedeutet, dass die Internetverbindung trotz eines Ausfalls des Firewall-Systems weiter gewährleistet sein muss. Dieses Kapitel zeigt Ihnen verschiedene Wege, um dieses Ziel zu erreichen. Dabei betrachte ich auch die Zustandssynchronisation über den Conntrack-Daemon. Die ursprünglich geplante Synchronisation über einen Kernelthread mit `ct_sync` wurde nicht weiterentwickelt.

27.1 Was ist Hochverfügbarkeit, und wo ist das Problem?

Hochverfügbarkeit ist eine heute übliche Anforderung an geschäftskritische Systeme. In vielen Unternehmen zählt auch die Firewall hierzu. Ein Ausfall der Firewall oder der Internetverbindung verursacht hohe Kosten. Potenzielle Kunden können nicht auf die Website gelangen, und wichtige E-Mails erreichen nicht ihren Empfänger. Wenn Sie Ihre Firewall hochverfügbar auslegen, bedeutet das, dass ein Ausfall der Funktion sehr selten ist.

Die Verfügbarkeit wird in Prozent gemessen. Wenn Ihre Firewall eine 99 %ige Verfügbarkeit aufweisen soll, bedeutet das, dass sie nur 1 % der Zeit im Jahr ausfallen darf. Bei 365 Tagen sind das 3 Tage, 15 Stunden und 36 Minuten. Dies können Sie vielleicht noch mit einem einfachen System gewährleisten. Bei einem Ausfall pro Jahr haben Sie sicherlich innerhalb von 1 bis 2 Tagen Ersatz geschaffen. Meist wird aber eine Verfügbarkeit von 99,9 % oder sogar 99,99 % gefordert. Das bedeutet, dass das System nur 0,1 % (8 Stunden, 46 Minuten) oder gar 0,01 % (52 Minuten) im Jahr ausfallen darf. Ein einfacher Stromausfall, ein Upgrade des Betriebssystems oder ein Reboot bereitet da unter Umständen schon Kopfschmerzen.

Falls Sie eine derartige Verfügbarkeit garantieren müssen, können Sie das nur leisten, wenn im Falle des Ausfalls des primären Systems die Funktion durch ein zweites Backup-System aufrechterhalten wird. Daher werden derartige Lösungen meist als Cluster aus zwei Knoten implementiert, die sich gegenseitig ersetzen können.

Bei einem einfachen Hochverfügbarkeits-Cluster ist einer der beiden Knoten aktiv (Master) und stellt die Funktion zur Verfügung, während der zweite Knoten passiv als Hot-Standby nur im Fehlerfall die Funktion des Masters übernimmt (Slave, Hot-Standby). Dies nennt man auch einen Hot-Standby-Cluster. Fortgeschrittene Cluster können die Hochverfügbarkeit auch mit zwei aktiven Knoten realisieren. Solange beide Knoten zur Verfügung stehen, wird die Funk-

tion über beide Knoten realisiert. Sobald ein Knoten ausfällt, übernimmt der zweite Knoten die komplette Funktion. In dieser Konstellation ist die Backup-Hardware nicht die meiste Zeit ungenutzt, sondern wird sinnvoll eingesetzt. Diese Cluster werden als Active-Active-Cluster bezeichnet.

Das wesentliche Problem bei einem Cluster ist die gemeinsame Nutzung der identischen Daten für die Bereitstellung der Funktion. Ein Datenbank- oder Webserver-Cluster muss sowohl auf dem Master als auch auf dem Hot-Standby-Knoten über identische Webseiten oder Datenbanken verfügen. Handelt es sich um statische Daten, so können diese einfach einmalig synchronisiert werden. Werden diese Daten jedoch bei der Verwendung des Systems dynamisch modifiziert, weil zum Beispiel neue Einträge in der Datenbank vorgenommen werden oder vorhandene Einträge geändert werden, so müssen diese Änderungen auch sofort auf den Hot-Standby übertragen (synchronisiert) werden. Ansonsten sind die Informationen bei einem anschließenden Ausfall des Masters verloren. Bei Datenbanken und Webservern wird dies häufig durch ein gemeinsam genutztes Dateisystem (Shared Storage) zum Beispiel in einem Storage-Area-Network (SAN) realisiert.

Noch komplizierter wird die Realisierung eines Active-Active-Clusters, bei dem theoretisch auf beiden Knoten gleichzeitig ein Schreibzugriff auf das identische Feld in einer Datenbank erfolgen kann. Hier sind sehr intelligente Mechanismen erforderlich, um die Datenkonsistenz zu gewährleisten.

Wo ist nun das Problem bei Firewalls? Eine Firewall verfügt doch über einen statischen Regelsatz, der sehr einfach über alle Knoten in einem Cluster synchronisiert werden kann und nur selten verändert wird. Da die Modifikationen des Regelwerks durch einen geschulten Administrator vorgenommen werden, ist es sehr unwahrscheinlich, dass dieser die Regeln gleichzeitig auf zwei Systemen verändern wird. Ein Konflikt kann daher nicht entstehen.

Dennoch tauchen Probleme auf, denn moderne Firewalls wie Iptables/Netfilter sind zustandsorientierte Firewalls. Diese Firewalls werten nicht nur ihre statischen Regeln aus, sondern auch eine Zustandstabelle, die sich in Abhängigkeit von den bereits betrachteten Paketen ändert. Die statischen Firewall-Regeln entscheiden nicht allein, ob ein Paket die Firewall passieren darf oder verworfen wird. Damit bei dem Ausfall des Masters der Hot-Standby die Funktion komplett übernehmen kann, müssten auch die Zustandsinformationen zwischen den Knoten synchronisiert werden. Ansonsten verwirft der neue Master möglicherweise die Pakete der bereits im Vorfeld aufgebauten Verbindungen.

27.2 Einfache Hochverfügbarkeit

Eine einfache Hochverfügbarkeit kann gewährleistet werden, wenn die Firewall die Zustände der Verbindungen nicht überwacht. Das bedeutet, dass Sie sowohl Regeln für eingehende Pakete als auch für ausgehende Pakete definieren müssen und kein NAT oder Masquerading verwenden dürfen. Sobald Sie diese Voraussetzungen gewährleisten können, ist es recht einfach, eine hochverfügbare Firewall aufzubauen. Sie müssen dann nur mit einer der verfügbaren Clusterlösungen die Ausfallsicherheit der Systeme herstellen und bei einem Ausfall die

Übernahme der IP-Adressen von dem ausgefallenen System auf das Ersatzsystem veranlassen. Wenn auf beiden Systemen identische Regelsätze zum Einsatz kommen, ist die Funktionalität sofort wiederhergestellt. Da keine Zustandsinformationen gespeichert werden, müssen diese auch nicht zwischen den Firewalls synchronisiert werden. Sämtliche Verbindungen laufen ohne Unterbrechung weiter.

Nun ist dieses Buch aber kein Buch über `ipchains` (eine zustandslose Firewall), sondern betrachtet mit Netfilter/Iptables eine zustandsorientierte Firewall. Sicherlich kann man auch mit Iptables zustandslose Firewalls konfigurieren, allerdings ist das unter sicherheitstechnischen Gesichtspunkten nicht sinnvoll. Daher werde ich diesen Punkt hier nicht weiter ausführen und direkt zur Hochverfügbarkeit bei einer zustandsorientierten Firewall übergehen.

27.3 Hochverfügbarkeit bei zustandsorientierten Firewalls

Natürlich können Sie auch bei einer zustandsorientierten Firewall denselben Ansatz wählen, wie ich ihn gerade bei der zustandslosen Firewall beschrieben habe. Sie wählen eine Cluster-Software Ihrer Wahl (zum Beispiel Heartbeat oder KeepAlived) und konfigurieren diese so, dass sie beim Ausfall des Master-Nodes automatisch ein Fail-Over zum Hot-Standby durchführt. Bei diesem Fail-Over werden automatisch von der Cluster-Software auch die IP-Adressen übertragen. Verfügt nun der Hot-Standby-Node über ein identisches Regelwerk, ist eine gewisse Funktionalität sofort wiederhergestellt. Sie können sofort wieder neue Verbindungen über die Firewall aufbauen. Leider sind alle zum Zeitpunkt des Fail-Overs aufgebauten Verbindungen nun tot. Die Information, welche Verbindungen von der ausgefallenen Firewall zugelassen wurden, befand sich in deren Zustandstabelle. Deren Inhalt ist verloren und steht der neuen Firewall nicht zur Verfügung. Daher kennt diese nicht die vor dem Fail-Over existenten Verbindungen und wird diese nicht erlauben.

Wenn Sie den Verlust der aufgebauten Verbindungen bei dem Fail-Over akzeptieren können, ist die Realisierung dieser Hochverfügbarkeitslösung die einfachste und eine sehr unproblematische Variante.

27.4 Automatische Erkennung aufgebauter Verbindungen nach dem Fail-Over

Sie können mit dieser Lösung, wenn Sie kein NAT einsetzen, sogar in vielen Fällen echte Hochverfügbarkeit garantieren. Dafür müssen Sie jedoch Ihre Regeln unter Umständen anpassen.

Die Wiedererkennung aufgebauter Verbindungen nach einem Fail-Over ist insbesondere für Langzeit-TCP-Verbindungen wichtig. Das bedeutet, dass der Hot-Standby nach einem Fail-Over besonders diese Verbindungen erkennen und wieder in seine Zustandstabelle aufnehmen muss. Der Connection-Tracking-Code von Netfilter ist hierzu in der Lage. Bei dem Pro-

tokoll TCP werden die folgenden zwei Pakete als neue Pakete (State: NEW) akzeptiert:

» TCP-SYN-Pakete: Diese Pakete bauen normalerweise eine Verbindung auf.

» TCP-ACK-Pakete: Diese Pakete werden nach dem Aufbau einer Verbindung ausgetauscht. Dieses Paket überführt eine unbekannte Verbindung sofort in den Zustand ESTABLISHED.

Wenn Sie folgende Regeln verwenden, kann Ihre Firewall nach einem Fail-Over vorher vorhandene Verbindungen erkennen und automatisch wieder erlauben:

```
INTDEV=eth0
EXTDEV=eth1
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp --dport 22 -m state -- ↵
        state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state ESTABLISHED,RELATED -j ACCEPT
```

Wenn vor dem Fail-Over nun ein interner Client eine SSH-Verbindung zu einem Server im Internet aufbaut, wird er den kompletten TCP-Handshake durchlaufen, und die Firewall wird die Verbindung in Ihrer Zustandstabelle eintragen. Alle weiteren Pakete, die der Client nun mit dem Server austauscht, tragen bis zur Beendigung der Verbindung lediglich das TCP-ACK-Bit. Kommt es nun zum Fail-Over, so verfügt der Hot-Standby über den identischen Regelsatz, aber nicht über die Zustandstabelle. Die Verbindung ist daher unbekannt. Wenn nun der Server ein weiteres TCP-ACK-Paket an den Client sendet, wird die Firewall dieses Paket nicht zulassen, da sie die Verbindung nicht kennt. Sendet der Client jedoch ein TCP-ACK-Paket an den Server, so akzeptiert die Firewall auch dieses Paket als Verbindungsaufbau und trägt die Verbindung sofort wieder in die Zustandstabelle mit dem Zustand ESTABLISHED ein. Alle weiteren Pakete dieser Verbindung, des Clients und des Servers, dürfen dann auch den Hot-Standby passieren.

Diese Funktionalität ist nicht gegeben, wenn Sie in den Regeln zusätzlich die Option --syn angeben:

```
$INTDEV=eth0
$EXTDEV=eth1
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp --dport 22 --syn -m ↵
        state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state ESTABLISHED,RELATED -j ACCEPT
```

Nun wird nur dann ein Paket als Verbindungsaufbau akzeptiert, wenn es sich tatsächlich um ein TCP-SYN-Paket handelt. Eine Wiederaufnahme einer alten Verbindung nach einem Fail-Over oder auch nach einem Neustart der Firewall ist nicht mehr möglich. Die Verbindungen „hängen" und müssen komplett neu aufgebaut werden.

Auch wenn Sie den Parameter /proc/sys/net/netfilter/nf_conntrack_tcp_loose auf null setzen, funktioniert dies nicht mehr (siehe Abschnitt 19.3.11).

Außerdem funktioniert die Wiederaufnahme von alten Verbindungen nicht richtig, wenn die Firewall zusätzlich noch ein Masquerading oder ein N:1-NAT durchführt.

27.4.1 Praktische Implementierung mit KeepAlived

Um nun eine derartige hochverfügbare Firewall aus zwei Knoten zu implementieren, benötigen Sie eine Software, mit der Sie die Verfügbarkeit prüfen und einen Fail-Over durchführen können. Ein Kandidat für diese Aufgabe ist KeepAlived[1]. Diese Software von dem Linux-Virtual-Server-Projekt (LVS) ist ein Userspace-Daemon, der die Gesundheit der Cluster-Knoten überwacht und einen Fail-Over durchführen kann. Sie können die KeepAlived-Software von der Homepage `http://www.keepalived.org` herunterladen. Die KeepAlived-Software ist aber auch Bestandteil von vielen Distributionen. Prüfen Sie zunächst, ob Ihre Distribution vielleicht ein passendes Paket zur Verfügung stellt.

Nach der Installation müssen Sie KeepAlived konfigurieren. KeepAlived soll eine hochverfügbare Firewall realisieren. Wenn wir für einen Moment die Firewall-Regeln vernachlässigen, handelt es sich bei der Funktion, die wir hochverfügbar anbieten möchten, um die Routing-Funktionalität der Firewall. Dass die Firewall nebenbei auch noch filtert, ist ja nur ein zusätzliches Bonbon. Speziell für diesen Zweck implementiert KeepAlived das Virtual Router Redundancy Protocol (VRRP, RFC2338).

STOP

Das VRRP-Protokoll ist möglicherweise durch Patente geschützt. Sowohl Cisco (`http://www.ietf.org/ietf/IPR//VRRP-CISCO`) als auch IBM (`http://www.ietf.org/ietf/IPR/NAT-VRRP-IBM`) behaupten, dass das VRRP-Protokoll ihre Patente verletzen würde. Deshalb hat das OpenBSD-Projekt mit dem Common Address Redundant Protocol (CARP) ein patentfreies und sicheres Protokoll geschaffen. Mit UCARP, einem Userspace-Daemon (`http://www.ucarp.org`), kann dieses Protokoll auf Linux und BSD benutzt werden. Ich verwende hier den KeepAlived mit VRRP, da dies auch die bevorzugte Variante des Netfilter-Teams ist.

Sie können mit KeepAlived zwei oder mehr redundante Router zusammenfassen, sodass sie sich als ein virtueller Router dem lokalen Netz präsentieren. Hierzu beobachtet KeepAlived mit dem VRRP-Protokoll die redundanten Router und deren Gesundheit und wählt einen Master-Router, der zusätzlich zu seiner realen IP-Adresse die virtuelle Adresse des virtuellen Routers erhält. Alle weiteren Router werden als Slave bezeichnet. Diese kontrollieren die Gesundheit des Masters und übernehmen bei dessen Ausfall die virtuelle Router-IP-Adresse. Ist der Master später wieder verfügbar, so übergibt der Slave die virtuelle Router-IP-Adresse wieder an den Master.

Das VRRP-Protokoll ist von der Internet Engineering Task Force (IETF) spezifiziert worden und wird von vielen kommerziellen Routern ebenfalls unterstützt.

Die Konfiguration des KeepAlived-Daemons erfolgt in der Datei `/etc/keepalived/keepalived.conf`. Eine Beispieldatei für einen Master könnte folgendermaßen aussehen:

```
vrrp_sync_group VG1 {
   group {
     eth0 eth1}
}
```

1 Alternativen stellen Heartbeat bzw. Corosync/Pacemaker dar.

```
! Interne virtuelle IP-Adresse
vrrp_instance VI_1 {
   state MASTER
   interface eth0
   virtual_router_id 1
   priority 100
   authentication {
      auth_type PASS
      auth_pass password
   }
   virtual_ipaddress {
      192.168.1.1/24 brd 192.168.1.255 dev eth0
   }
}
! Externe virtuelle IP-Adresse
vrrp_instance VE_1 {
   state MASTER
   interface eth1
   lvs_sync_daemon_interface eth0
   virtual_router_id 2
   priority 100
   authentication {
      auth_type PASS
      auth_pass password
   }
   virtual_ipaddress {
      192.168.2.1/24 brd 192.168.1.255 dev eth1
   }
}
```

Die entsprechende Datei für den Slave hat folgenden Inhalt:

```
vrrp_sync_group VG1 {
   group {
      eth0 eth1}
}
vrrp_instance VI_1 {
   state BACKUP
   interface eth0
   virtual_router_id 1
   priority 50
   authentication {
      auth_type PASS
      auth_pass password}
```

```
    virtual_ipaddress {
       192.168.1.1/24 brd 192.168.1.255 dev eth0
    }
}
vrrp_instance VE_1 {
    state BACKUP
    interface eth1
    lvs_sync_daemon_interface eth0
    virtual_router_id 2
    priority 50
    authentication {
       auth_type PASS
       auth_pass password
    }
    virtual_ipaddress {
       192.168.2.1/24 brd 192.168.1.255 dev eth1
    }
}
```

Wichtig ist, dass sowohl der Slave als auch der Master die identische `virtual_router_id` für die jeweilige Instanz besitzen. Jede Instanz ist für eine virtuelle IP-Adresse des Routers verantwortlich. Damit diese IP-Adressen bei dem Ausfall einer IP-Adresse auch beide gemeinsam übertragen werden, werden sie in einer `vrrp_sync_group` zusammengefasst. Die vergebene Priorität entscheidet, welches der beiden Systeme der Master ist. Bei mehreren Slaves variieren Sie die Priorität leicht zwischen den verschiedenen Slaves (50, 49, 48 etc.). Mit dem Parameter `state` können Sie entscheiden, in welchem Zustand der KeepAlived-Daemon auf diesem System starten soll: `MASTER` oder `BACKUP`. Die `virtual_ipaddress` ist einmal die interne und einmal die externe IP-Adresse des virtuellen Routers. Die PASS-Authentifizierung tauscht die Kennwörter im Klartext aus. Alternativ können Sie die AH-Authentifizierung wählen. Der Parameter `lvs_sync_daemon_interface` definiert die Netzwerkkarte, über die die Kommunikation zwischen den KeepAlived-Daemons für diese Instanz erfolgen soll. Auf einer Firewall ist es sinnvoll, diese Informationen in dem internen Netz zu halten oder sogar auf ein eigenes getrenntes Netz auszuweichen. Dann stellt die Klartext-Authentifzierung auch kein Problem mehr dar.

Wenn Sie nun den KeepAlived-Daemon starten, wird das erste System sich als Master konfigurieren und die virtuelle IP-Adresse übernehmen. Sobald der Master ausfällt, wird der Slave die Funktion übernehmen. Sie müssen nun nur allen internen Clients als Standard-Gateway die virtuelle IP-Adresse des Routers zuweisen: 192.168.1.1/24. Wenn Sie nun noch Ihre Firewallregeln auf beiden Systemen synchronisieren und laden, können Sie Ihre neue hochverfügbare Firewall nutzen!

27.4.2 Hochverfügbarkeit und Masquerading/NAT

Sobald Sie SNAT oder Masquerading nutzen, wird Ihr hochverfügbarer Firewall-Cluster nicht mehr richtig funktionieren. Um das zu verstehen, müssen Sie sich vor Augen führen, dass eine Firewall bei einem Masquerading oder N:1-SNAT[2] häufig nicht nur die IP-Adresse, sondern auch den TCP- oder UDP-Port austauscht. Stellen Sie sich vor, ein Client1 baut durch Ihre Firewall eine Verbindung zu einem Webserver in dem Internet auf. Der Client1 verwendet den Quellport 1027. Dieser Port ist jedoch auf der Firewall bereits durch eine andere genattete Verbindung belegt. Dann wird die Firewall bei dem NAT nicht nur die IP-Adresse, sondern auch den Quellport modifizieren. Hierfür wählt sie den nächsten geeigneten freien Port aus. Welcher das ist, hängt natürlich sehr von den anderen gerade aktiven und in der Vergangenheit aktiven Verbindungen ab. In unserem Beispiel (siehe Abbildung 27.1) ist es der Port 1275. Für den Webserver wird die Verbindung nun von 3.0.0.1:1275 aufgebaut.

Nach dem Fail-Over wird der zweite Knoten des Firewall-Clusters die Verbindung wieder aufnehmen. Da sich dieser aber nicht in demselben Zustand befindet wie der alte Master vor dem Fail-Over, wird er bei der Wahl des NAT-Ports entweder den Port gar nicht verändern, da dieser Port 1027 noch frei ist, oder sicherlich einen anderen Port als der erste Master wählen. Der Webserver wird die Pakete aber nicht zu der Verbindung zählen, da sie über einen falschen Quellport verfügen und die Pakete verwerfen. Die Verbindung hängt bis zum Timeout.

Sobald Sie NAT einsetzen, ist es daher zwingend erforderlich, dass die Zustandstabelle von dem Master auf alle Slaves in dem Cluster synchronisiert wird, da die hier gespeicherten Informationen (NAT) für die Modifikation der Pakete bei dem NAT der Verbindung zwingend erforderlich sind. Diese Zustandssynchronisation kann mit dem Conntrack-Daemon durchgeführt werden.

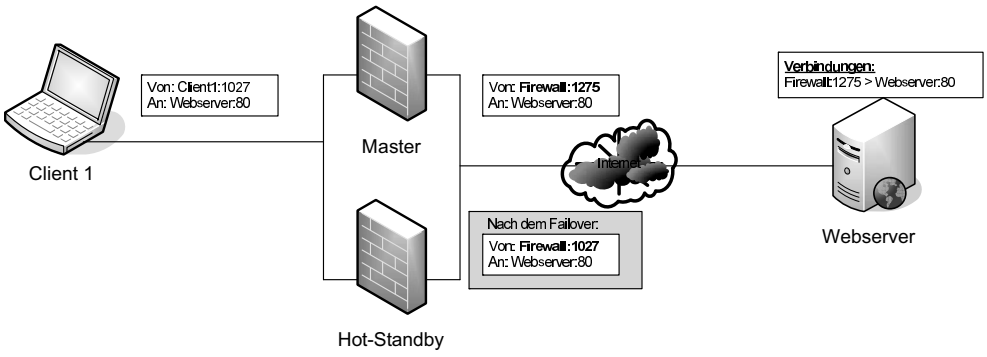


Abbildung 27.1: **NAPT (Network Address and Port Translation) macht bei einem einfachen Firewall-Cluster CE! einen Strich durch die Rechnung.**

2 Ein N:1-SNAT ist ein Source-NAT, bei dem viele verschiedene Absender-IP-Adressen gegen eine IP-Adresse ausgetauscht werden. Dies ist der häufigste Fall des SNAT. Masquerading bezeichnet unter Linux dann den Umstand, dass diese IP-Adresse dynamisch zugewiesen wird.

CE! dem Benutzer?

TS^m Bitte Jahreszahl ergänzen.

Editor's or typesetter's annotations (will be removed before the final TeX run)

27.5 Zustandssynchronisation mit conntrackd

Viele kommerzielle Firewalls verfügen seit mehreren Jahren über eine Synchronisation der Zustandstabelle in einem Firewall-Cluster. Selbst OpenBSD hat mit `pfsync` eine seit Jahren stabile Implementierung dieser Funktion. Für den Linux-Kernel existierte 19xx TS[m], zum Zeitpunkt der ersten Auflage dieses Buches, eine rudimentäre Lösung, die aber fehlerbehaftet war. Dieser Code war für den Linux-Kernel 2.4.26 und den Linux-Kernel 2.6.10 verfügbar. Er implementierte mit `ct_sync` ein Connection-Tracking-Synchronisationsframework, das den Aufbau eines Hot-Standby-Clusters ermöglicht. Diese Lösung wurde jedoch nicht weiterentwickelt.

Stattdessen haben die Netfilter-Entwickler mit dem Conntrack-Daemon eine Userspace-Applikation geschaffen, die sich um die Synchronisation der Zustandstabelle kümmert. Dieser Daemon baut auf den Conntrack-Tools auf, die eine Modifikation der Zustandstabelle zur Laufzeit erlauben.

27.5.1 Synchronisationsprotokolle

Der Conntrackd verwendet zwei Caches: einen internen und einen externen Cache. Der interne Cache erhält seine Informationen von dem lokalen Kernel und enthält eine Kopie der aktuellen lokalen Connection-Tracking-Tabelle. Der externe Cache enthält eine Kopie der Tabelle des entfernten Systems. Abbildung 27.2 zeigt die Funktion.

Der Conntrackd unterstützt drei verschiedene Synchronisationsprotokolle:

» `notrack`: Dieses Protokoll ist das einfachste der drei Protokolle. Es prüft nicht die erfolgreiche Synchronisation. Daher können Synchronisationsnachrichten verloren gehen. Durch die Verwendung von TCP als Transportprotokoll für dieses Synchronisationsprotokoll kann jedoch trotzdem eine verlässliche Übertragung gesichert werden.

» `ft-fw`: Dieses Protokoll führt eine eigene Prüfung durch. Daher kann es auch über UDP eine sichere Übertragung mit einer Fehlererkennung und -korrektur garantieren.

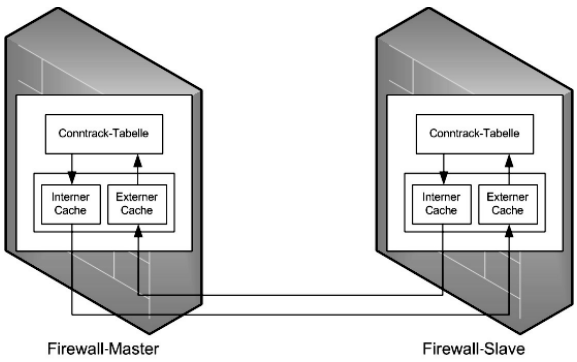


Abbildung 27.2: **Der Conntrackd-Daemon verwendet zwei Caches, um den Kernel zu entlasten.**

» `alarm`: Dieses Protokoll versendet in regelmäßigen Abständen sämtliche Einträge aus der Zustandstabelle erneut, selbst wenn es keine Änderungen gegeben hat. Damit kann es am schnellsten Synchronisationsprobleme beheben. Es benötigt aber auch die meiste Bandbreite.

Für jedes dieser Protokolle gibt es eine Beispielkonfigurationsdatei im Dokumentationsverzeichnis des Conntrackd-Daemons. Diese befinden sich in `sync/<protokoll>/conntrackd.conf`.

Für einen ersten Test ist es sinnvoll, das `notrack`-Protokoll zu nutzen. Später kann auf das `ft-fw`- oder `alarm`-Protokoll umgestellt werden.

Das notrack-Protokoll

Das `notrack`-Protokoll wurde ähnlich dem PFsync-Protokoll von OpenBSD entwickelt. Es besitzt keine eingebauten Prüfungen und erwartet eine verlässliche Übertragung. Dies kann zum Beispiel durch ein Crossover-Kabel zwischen den beiden Firewallsystemen gewährleistet werden. Hier kommt es üblicherweise nicht zu einem Paketverlust. Bei einer Übertragung über mehrere Systeme kann es ansonsten zu einem Paketverlust und damit zu einer fehlerhaften Synchronisation kommen (siehe Abbildung 27.3).

Das Protokoll sollte daher nur über verlässliche Medien oder mit einem verlässlichen Transportprotokoll wie TCP eingesetzt werden.

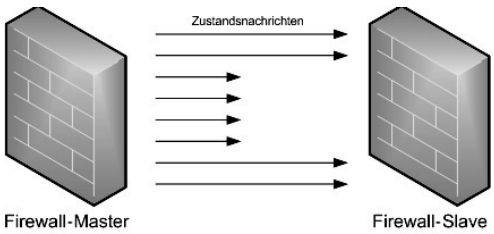


Abbildung 27.3: **Das notrack-Protokoll erkennt nicht den Verlust einzelner Pakete. Die Synchronisation ist dann nicht vollständig.**

Das ft-fw-Protokoll

Dieses Protokoll verwendet Sequenznummern für die einzelnen Synchronisationsnachrichten. Diese müssen von dem Empfänger bestätigt werden. Sobald einzelne Nachrichten nicht bestätigt wurden, werden diese erneut versandt. Damit ist die verlässliche Synchronisation auch gesichert, wenn es möglicherweise zu Paketverlusten kommt (siehe Abbildung 27.4).

In der Konfiguration dieses Protokolls kann die Größe der Queue, in der die möglicherweise neu zu sendenden Nachrichten gespeichert werden, mit der Option `ResendQueueSize` angegeben werden. Der Default beträgt 131.072 Nachrichten. Auch die Häufigkeit der Bestätigung kann mit `AckWindowSize` angegeben werden. Per Default werden alle 300 Nachrichten Bestätigungen versandt.

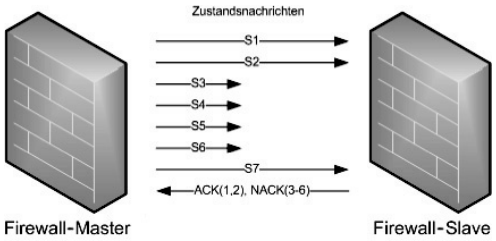


Abbildung 27.4: **Beim Verlust einzelner Pakete sendet der Absender bei fehlender Bestätigung die entsprechenden Synchronisationsnachrichten erneut. Die vollständige Synchronisation wird durch das Protokoll garantiert.**

Das alarm-Protokoll

Das Alarm-Protokoll nutzt einen anderen Weg. Um Synchronisationprobleme auszuschließen, versendet es alle Objekte in regelmäßigen Abständen neu. Dies betrifft auch Objekte, die sich nicht ändern. Die beiden anderen Protokolle versenden nur geänderte Objekte. Kommt es zu Fehlern in der Synchronisation, so werden diese automatisch sehr schnell durch dieses Protokoll behoben. Hierfür benötigt es jedoch eine wesentliche größere Bandbreite. Scherzhaft wird daher auch von Spamming gesprochen (siehe Abbildung 27.5).

Auch dieses Protokoll kennt einige Parameter, mit denen Sie es konfigurieren können. Im Wesentlichen ist das der Parameter `Refreshtime`, mit dem Sie angeben, in welchen Abständen auch nicht geänderte Objekte erneut synchronisiert werden sollen (Default: 15 Sekunden). Für den Empfänger gibt der `CacheTimeout`-Parameter an, nach welcher Zeit (in Sekunden) Objekte im Cache gelöscht werden sollen, wenn sie nicht erneut synchronisiert wurden (Default: 180 Sekunden).

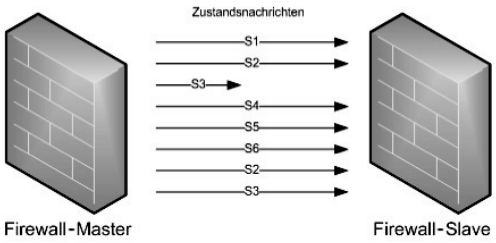


Abbildung 27.5: **Der Verlust einzelner Nachrichten hat nur eine geringe Auswirkung, da die gesamte Tabelle in regelmäßigen Abständen neu versandt wird.**

27.5.2 Aufbau eines Active/Backup-Conntrackd-Clusters

Zunächst benötigen Sie zwei identisch konfigurierten Rechner, die Sie mithilfe einer Heartbeat-Software wie KeepAlived oder Linux-HA-Heartbeat (`http://www.linux-ha.org`) hochverfügbar konfigurieren. Die eingesetzte Software muss lediglich in der Lage sein, die Gesundheit des Masters zu überwachen und bei einem Ausfall auf dem Slave ein Skript aufzurufen. Dieses Skript aktiviert den Slave und macht ihn dadurch zum Master und den Master zum Slave. Beispielskripte sind in dem Conntrack-Tools Paket enthalten. Hier ist auch eine Beispielkonfi-

guration für den KeepAlived enthalten (`doc/sync/keepalived.conf`). Diese müssen Sie nur entsprechend Ihrer IP-Adressen anpassen. Am Ende dieses Abschnitts stelle ich eine passende Konfiguration vor.

Bauen Sie zunächst Ihren Firewall-Cluster wie in Abbildung 27.6 auf. Achten Sie darauf, dass Ihre Cluster-Knoten über eine dedizierte Netzwerkkarte miteinander verbunden sind. Aktivieren Sie nun die Netzwerkkarte zur Synchronisation, und weisen Sie ihr eindeutige Adressen zum Beispiel in dem Netzwerk 10.0.0.0/24 zu. Sie sollten nun die anderen Knoten in dem Synchronisationsnetzwerk pingen können.

Weisen Sie außerdem den externen und internen Netzwerkkarten der beiden Firewall-Systeme eindeutige IP-Adressen zu. Nutzen Sie zum Beispiel extern 192.168.1.0/24 und intern 192.168.2.0/24. Dann erhält zum Beispiel die linke Firewall jeweils die IP-Adressen 192.168.1.101 und 192.168.2.101 und die rechte Firewall die IP-Adressen 192.168.1.102 und 192.168.2.102. Der KeepAlived kümmert sich dann um die Verteilung der virtuellen IP-Adressen 192.168.1.1 und 192.168.2.1.

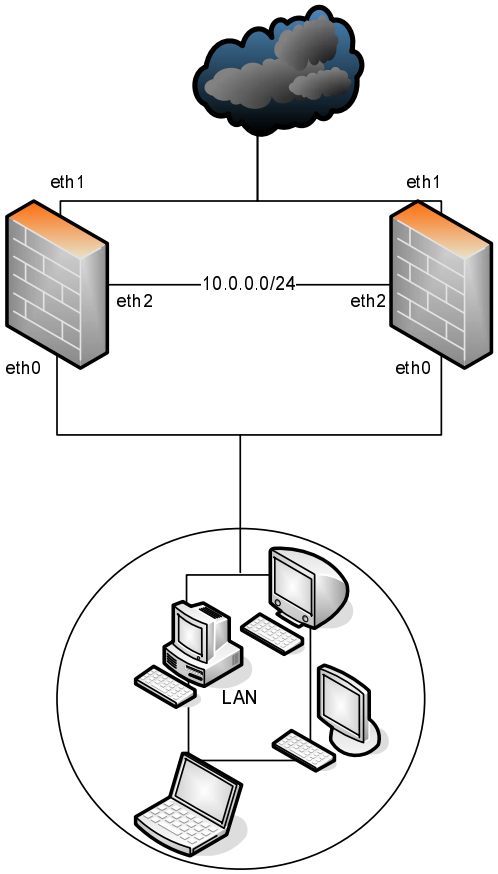


Abbildung 27.6: **Der Firewall-Cluster besteht aus zwei Knoten, die sich gegenseitig ersetzen.**

STOP

Für den seltenen Fall, dass Sie noch einen Kernel älter als 2.6.22 einsetzen sollten, müssen Sie das TCP-Window-Tracking (siehe Abschnitt 19.1.1) deaktivieren. Dieses wird hier noch nicht unterstützt. Setzen Sie hierzu den folgenden Parameter in der Datei `/etc/sysctl.conf`*:*

```
net.netfilter.nf_conntrack_tcp_be_liberal=1
```

Nun müssen Sie auf beiden Systemen die für Ihr gewähltes Protokoll richtige Konfigurationsdatei in das Conntrackd-Konfigurationsverzeichnis kopieren:

```
cp doc/sync/notrack/conntrackd.conf /etc/conntrackd
```

Prüfen Sie diese Datei, und passen Sie die notwendigen Parameter an. In unserem Beispiel sind dies:

```
Sync {
    Multicast {
        IPv4_interface 10.0.0.X
        Interface eth2
    }
}
General {
    Filter From Userspace {
        Protocol Accept {
            TCP
        }
        Address Ignore {
          IPv4_address <Tragen Sie hier die IP-Adressen der Firewall ein>
        }
    }
}
```

Es ist sinnvoll, sämtliche Verbindungen, die von der Firewall selbst initiiert werden oder von ihr entgegengenommen werden, nicht zu synchronisieren. Hierzu tragen Sie diese als zu ignorierende IP-Adressen ein. Wenn Sie einen aktuellen Kernel (2.6.29 oder neuer) verwenden, können Sie die Filterung in den Kernel verschieben. Dies ist performanter. Dann verwenden Sie als Schlüsselwort statt `Userspace` `Kernelspace` in der Zeile `Filter From`. Kurzlebige Verbindungen müssen nicht synchronisiert werden. Daher beschränken wir die Synchronisation nur auf TCP-Verbindungen. Protokolle, die UDP verwenden, bauen üblicherweise keine langlebigen Verbindungen auf. Auch ICMP kennt keine langlebigen Verbindungen.

Die komplette Conntrackd-Konfigurationsdatei hat, nach dem Entfernen der überflüssigen Kommentare, den folgenden Inhalt:

```
Sync {
    Mode NOTRACK {
    }
    Multicast {
        IPv4_address 225.0.0.50
        Group 3780
        IPv4_interface 10.0.0.101
        Interface eth2
        SndSocketBuffer 1249280
        RcvSocketBuffer 1249280
        Checksum on
        }
    General {
        Nice -20
        HashSize 32768
        HashLimit 131072
        LogFile on LockFile /var/lock/conntrack.lock
        UNIX {
            Path /var/run/conntrackd.ctl Backlog 20
            }
        NetlinkBufferSize 2097152
        NetlinkBufferSizeMaxGrowth 8388608
        Filter From Userspace {
            Protocol Accept {
                TCP
                # ICMP
                # This requires a Linux kernel >= 2.6.31
            }
            Address Ignore {
                IPv4_address 127.0.0.1 # loopback
                IPv4_address 192.168.1.1 # virtual IP 1
                IPv4_address 192.168.2.1 # virtual IP 2
                IPv4_address 192.168.1.101
                IPv4_address 192.168.2.101
                IPv4_address 10.0.0.101 # dedicated link ip
                }
            }
        }
    }
```

Zusätzlich benötigen Sie das Skript, das aus die Standby-Firewall in die Master-Funktion versetzt. Auch dieses Skript ist enthalten:

```
cp doc/sync/primary-backup.sh /etc/conntrackd chmod 755 /etc/conntrackd/ ↵
        primary-backup.sh
```

Nun benötigen Sie auch noch Firewallregeln. Für einen ersten Test mögen die folgenden Regeln genügen:

```
#!/bin/bash

INTDEV=eth1
EXTDEV=eth0
SYNCDEV=eth2

VIRTEXTIP=192.168.1.1

IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
MODPROBE=/sbin/modprobe

$MODPROBE nf_conntrack_ftp
$MODPROBE nf_nat_ftp

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

$IPTABLES -F
$IPTABLES -F -t nat
$IPTABLES -F -t mangle

$IPTABLES -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT -p vrrp -j ACCEPT
$IPTABLES -A INPUT -i $SYNCDEV -j ACCEPT
$IPTABLES -A INPUT -i $INTDEV -j ACCEPT
$IPTABLES -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $SYNCDEV -j ACCEPT
$IPTABLES -A INPUT -i lo -j ACCEPT

$IPTABLES -A OUTPUT -o lo -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
$IPTABLES -A FORWARD -i $INTDEV -m state --state NEW -j ACCEPT

$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -j SNAT --to $VIRTEXTIP

$SYSCTL net.ipv4.ip_forward=1
```

Nun müssen Sie den Conntrack-Daemon auf beiden Systemen starten. Auf Debian-Distributionen ist hierfür ein Start-Skript vorbereitet.

Die Conntrackd-Dämonen werden nun bereits Daten austauschen. Sie können dies verfolgen, indem Sie den Befehl `conntrackd -s` aufrufen:

```
# conntrackd -s
cache internal:
current active connections:   1
connections created:         13   failed: 0
connections updated:         28   failed: 0
connections destroyed:       12   failed: 0

cache external:
current active connections:   1
connections created:        311   failed: 0
connections updated:         13   failed: 0
connections destroyed:      310   failed: 0

traffic processed: 107770 Bytes 246 Pckts

multicast traffic (active device=eth2):
70656 Bytes sent   73404 Bytes recv
 5979 Pckts sent    5884 Pckts recv
    1 Error send       0 Error recv

message sequence tracking:
   0 Msgs mfrm   9 Msgs lost
```

Um den Inhalt des internen oder externen Cache anzeigen zu lassen, nutzen Sie ebenfalls den Befehl `conntrackd`. Die Option `-i` zeigt den internen, während die Option `-e` den externen Cache zeigt.

```
# conntrackd -i
tcp 6 ESTABLISHED src=192.168.2.10 dst=192.168.255.1 sport=52385 dport=22 ↵
         src=192.168.255.1 dst=192.168.1.102 sport=22 dport=52385 [ ↵
        ASSURED] mark=0  secmark=0 [active since 573s]
```

Um das Verhalten des Conntrackd zu verfolgen, können Sie auch dessen Protokolldatei mitlesen:

```
# tail -f /var/log/conntrackd.log
[Wed Apr 6 13:39:46 2011] (pid=1992) [notice] resync with master table
[Wed Apr 6 13:39:46 2011] (pid=1992) [notice] sending bulk update
[Wed Apr 6 13:54:54 2011] (pid=1992) [notice] flushing conntrack table in ↵
         60 secs
[Wed Apr 6 13:54:54 2011] (pid=1992) [notice] request resync
[Wed Apr 6 13:54:59 2011] (pid=1992) [notice] committing external cache
[Wed Apr 6 13:54:59 2011] (pid=1992) [notice] Committed 2 new entries
[Wed Apr 6 13:54:59 2011] (pid=1992) [notice] commit has taken 0.008336  ↵
        seconds
[Wed Apr 6 13:54:59 2011] (pid=1992) [notice] flushing caches
[Wed Apr 6 13:54:59 2011] (pid=1992) [notice] resync with master table
[Wed Apr 6 13:54:59 2011] (pid=1992) [notice] sending bulk update
```

Zusätzlich müssen Sie auch den KeepAlived starten. Dessen Konfigurationsdatei `keepalived.conf` könnte auf dem Master folgendermaßen aussehen:

```
vrrp_sync_group G1 {
    # must be before vrrp_instance declaration
    group {
        VI_1
        VI_2
        }
    notify_master "/etc/conntrackd/primary-backup.sh primary"
    notify_backup "/etc/conntrackd/primary-backup.sh backup"
    notify_fault "/etc/conntrackd/primary-backup.sh fault"
}

vrrp_instance VI_1 {
    interface eth1
    state SLAVE
    virtual_router_id 61
    lvs_sync_daemon_interface eth2
    priority 100
    advert_int 3
    authentication {
        auth_type PASS
        auth_pass papas_con_tomate
        }
    virtual_ipaddress {
        192.168.2.1/24
        }
    }
```

```
vrrp_instance VI_2 {
    interface eth0
    state SLAVE
    virtual_router_id 62
    lvs_sync_daemon_interface eth2
    priority 100
    advert_int 3
    authentication {
        auth_type PASS
        auth_pass papas_con_tomate
        }
    virtual_ipaddress {
        192.168.1.1/24
        }
    }
```

Auf dem Slave ändert sich nur die Priorität in beiden Instanzen. Setzen Sie hier zum Beispiel als Priorität 50.

Auch der KeepAlived protokolliert. Seine Nachrichten tauchen auf einem Debian-System in der Datei `/var/log/syslog` auf.

```
Apr 6 14:22:58 squeezeFW1 Keepalived: Starting Keepalived v1.1.20 ↵
        (08/18,2010)
Apr 6 14:22:58 squeezeFW1 Keepalived_healthcheckers: Initializing ipvs ↵
        2.6
Apr 6 14:22:58 squeezeFW1 Keepalived: Starting Healthcheck child process, ↵
         pid=3833
Apr 6 14:22:58 squeezeFW1 Keepalived_healthcheckers: Registering Kernel ↵
        netlink reflector
Apr 6 14:22:58 squeezeFW1 Keepalived_healthcheckers: Registering Kernel ↵
        netlink command channel
Apr 6 14:22:58 squeezeFW1 Keepalived_vrrp: Registering Kernel netlink ↵
        reflector
Apr 6 14:22:58 squeezeFW1 Keepalived: Starting VRRP child process, pid ↵
        =3835
Apr 6 14:22:58 squeezeFW1 Keepalived_vrrp: Registering Kernel netlink ↵
        command channel
Apr 6 14:22:58 squeezeFW1 Keepalived_vrrp: Registering gratutious ARP ↵
        shared channel
Apr 6 14:22:58 squeezeFW1  Keepalived_vrrp: Initializing ipvs 2.6
Apr 6 14:22:58 squeezeFW1 Keepalived_healthcheckers: Opening file '/etc/ ↵
        keepalived/keepalived.conf'.
Apr 6 14:22:58 squeezeFW1 Keepalived_healthcheckers: Configuration is ↵
        using : 4958 Bytes
```

```
Apr 6 14:22:58 squeezeFW1 Keepalived_healthcheckers: Using LinkWatch ↵
        kernel netlink reflector...
Apr 6 14:22:58 squeezeFW1 Keepalived_vrrp: Opening file '/etc/keepalived/ ↵
        keepalived.conf'.
Apr 6 14:22:58 squeezeFW1 Keepalived_vrrp: Configuration is using : 67143 ↵
         Bytes
Apr 6 14:22:58 squeezeFW1 Keepalived_vrrp: Using LinkWatch kernel netlink ↵
         reflector...
Apr 6 14:22:58 squeezeFW1 Keepalived_vrrp: VRRP_Instance(VI_1) Entering ↵
        BACKUP STATE
Apr 6 14:22:58 squeezeFW1 Keepalived_vrrp: Opening script file /etc/ ↵
        conntrackd/primary-backup.sh
Apr 6 14:22:58 squeezeFW1 Keepalived_vrrp: VRRP_Instance(VI_2) Entering ↵
        BACKUP STATE
Apr 6 14:22:58 squeezeFW1 kernel: [ 8042.984614] IPVS: sync thread ↵
        started: state = BACKUP, mcast_ifn = eth2, syncid = 61
Apr 6 14:23:23 squeezeFW1 Keepalived_vrrp: VRRP_Instance(VI_1) forcing a ↵
        new MASTER election
Apr 6 14:23:23 squeezeFW1 Keepalived_vrrp: VRRP_Group(G1) Transition to ↵
        MASTER state
Apr 6 14:23:23 squeezeFW1 Keepalived_vrrp: VRRP_Instance(VI_2) forcing a ↵
        new MASTER election
Apr 6 14:23:29 squeezeFW1 Keepalived_vrrp: VRRP_Instance(VI_2) Transition ↵
         to MASTER STATE
Apr 6 14:23:29 squeezeFW1 Keepalived_vrrp: VRRP_Group(G1) Syncing ↵
        instances to MASTER state
Apr 6 14:23:29 squeezeFW1 Keepalived_vrrp: VRRP_Instance(VI_1) Transition ↵
         to MASTER STATE
Apr 6 14:23:29 squeezeFW1 Keepalived_vrrp: Opening script file /etc/ ↵
        conntrackd/primary-backup.sh
Apr 6 14:23:33 squeezeFW1 Keepalived_vrrp: VRRP_Instance(VI_1) Entering ↵
        MASTER STATE
Apr 6 14:23:33 squeezeFW1 kernel: [ 8077.612103] IPVS: stopping backup ↵
        sync thread 3843 ...
Apr 6 14:23:33 squeezeFW1 kernel: [ 8077.624266] IPVS: sync thread ↵
        started: state = MASTER, mcast_ifn = eth2, syncid = 61
Apr 6 14:23:39 squeezeFW1 Keepalived_vrrp: VRRP_Instance(VI_2) Entering ↵
        MASTER STATE
```

INFO

Um Ihnen den Einstieg zu vereinfachen, habe ich auf der CD in dem Verzeichnis `HA-Firewall` virtuelle 64-Bit-Debian-Squeeze-Systeme für die KVM-Virtualisierung vorbereitet, mit denen Sie dieses Setup testen können. Weitere Informationen finden Sie in der Datei `README` in dem Verzeichnis.

28. Transparente Proxies

Ein transparenter Proxy ist ein für den Client und Server unsichtbarer Proxy. Um eine derartige Unsichtbarkeit zu erzeugen, müssen die Verbindungen zwischen dem Client und Server ohne deren Unterstützung über den Proxy geleitet werden.

Linux Netfilter bietet mit den Zielen `REDIRECT` und `TPROXY` die entsprechenden Funktionen.

Während in älterer Dokumentation bereits von einem transparenten Proxy gesprochen wird, wenn die Verbindung lediglich für den Client transparent über einen Proxy geleitet wird und dies meistens auch nur vorgestellt wird, werden wir uns auch mit einem echten transparenten Proxy beschäftigen, der auch für den Server transparent ist.

Ein Proxy, der für den Client transparent ist, kommuniziert mit dem Client unter Verwendung der tatsächlichen IP-Adresse des Servers, zu dem der Client die Verbindung aufbauen wollte. Ist der Proxy zusätzlich auch für den Server transparent, so verwendet er gegenüber dem Server die echte IP-Adresse des Clients.

Üblicherweise werden die transparenten Proxies für das HTTP-Protokoll genutzt, um weitere Analysen des Protokolls zu ermöglichen, ohne die Clients konfigurieren zu müssen. Im Weiteren werde ich daher auch nur diese Verwendung in Kombination mit dem Squid beschreiben.

28.1 Client-transparenter Proxy

Um einen einfachen Proxy aufzubauen, der für den Client transparent arbeitet, genügt eine einfache `iptables`-Regel:

```
$IPT -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port ↵
        3128
```

Nun müssen Sie nur dafür sorgen, dass die Clients erfolgreich eine Namensauflösung durchführen können. Sobald ein Client dann versucht, die Verbindung zum tatsächlichen Server im Internet auf Port 80/tcp aufzubauen, wird die Verbindung auf den lokalen Port 3128/tcp umgeleitet. Hier müssen Sie einen Squid-Proxy starten, der in der Lage ist, wie ein normaler Webserver zu reagieren.

Damit der Squid-Proxy ermitteln kann, mit welcher Website der Client sich tatsächlich verbinden wollte, muss der Client das Protokoll HTTP/1.1 verwenden. Nur dieses enthält den Host-Header, in dem der gewünschte DNS-Name übertragen wird. Damit der Squid-Proxy

diesen[CE] auswertet, müssen Sie ihn entsprechend konfigurieren. Die genaue Syntax hängt von der Squid-Version ab.

Bei Versionen älter als 2.6 verwenden Sie:

```
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

Ab der Version 2.6 verwenden Sie:

```
http_port 3128 transparent
```

28.2 Voll-transparenter Proxy

Ab der Version 3.1 kann der Squid auch als voll-transparenter Proxy eingesetzt werden. Hierzu muss er jedoch bei der Übersetzung mit der Option `./configure --enable-linux-netfilter` konfiguriert werden.

Der Kernel muss die `socket`-Erweiterung und das `TPROXY`-Target unterstützen.

Damit die Pakete im Kernel richtig gehandhabt werden, müssen zusätzliche Routen gesetzt werden. Diese dürfen nur auf später von `iptables` markierte Pakete angewendet werden. Um diese Regeln zu erzeugen, nutzen Sie den Befehl `ip`. Hierbei sind die Routen für die von TPROXY zu verwendenen Netzwerkkarten zu setzen.

```
ip rule add fwmark 1 lookup 100
ip -f inet route add local 0.0.0.0/0 dev lo table 100
ip -f inet route add local 0.0.0.0/0 dev eth0 table 100
```

Da die Pakete nun mit falschen IP-Adressen auf den Netzwerkkarten ankommen, ist es wichtig, die Reverse-Path-Filterfunktion abzuschalten (siehe auch Abschnitt 23.3.9) und das Forwarding zu aktivieren.

```
sysctl -w net.ipv4.conf.lo.rp_filter=0
sysctl -w net.ipv4.ip_forward=1
```

Nun müssen wir die neuen Verbindungen von aufgebauten Verbindungen unterscheiden. Die erste Regel erkennt die existierenden Verbindungen und verhindert, dass diese ein zweites Mal von dem `TPROXY`-Target bearbeitet werden.

```
iptables -t mangle -A PREROUITNG -p tcp -m socket -j DIVERT
iptables -t mangle -A PREROUTING -p tcp --dport 80 -j TPROXY --tproxy- ↵
        mark 0x1/0x1 --on-port 3129
```


[TS] Oder ist hier „die Clients dann versuchen“ gemeint?

[CE] „diesen“ wegen „Namensraum“?

Die benutzerdefinierte Kette `DIVERT` hat folgenden Inhalt:

```
iptables -t mangle -N DIVERT
iptables -t mangle -A DIVERT -j MARK --set-mark 1
iptables -t mangle -A DIVERT -j ACCEPT
```

Nun benötigen wir noch den Squid-Proxy. Dessen Konfiguration ist recht einfach. Für den transparenten Proxy ist ein eigener Port erforderlich. Soll der Squid noch als normaler Proxy eingesetzt werden, müssen zwei Ports vorgesehen werden:

```
http_port 3128
http_port 3129 tproxy
```

Weitere Hinweise und auch Tipps zur Fehlersuche finden Sie im Squid-Wiki unter `http://wiki.squid-cache.org/Features/Tproxy4`.

Teil VI

Transparente Firewalls

Transparente Firewalls filtern auf der Schicht 2 der Netzwerkprotokolle. Sie sind daher auf der Schicht 3 unsichtbar und können ohne eine Modifikation der IP-Adressen in einem Netzwerk eingeführt werden.

29. ProxyARP

Die Verwendung von ProxyARP ist eine der einfachsten Varianten für die transparente Filterung von Paketen in einem Netzwerk. Hierbei wird ein Router in ein vorhandenes Netzwerk eingebracht, ohne dass das Netzwerk neu konfiguriert wird. Auf beiden Seiten des Routers befinden sich Systeme aus demselben Netzwerk. Um dennoch eine Kommunikation zwischen den Systemen zu ermöglichen und ARP-Auflösungen durchzuführen, unterstützt der Router ProxyARP. Eine Filterung ist dann ganz normal mit Iptables möglich.

29.1 Wie funktioniert ProxyARP?

Beim ProxyARP antwortet ein Rechner auf eine ARP-Anfrage anstelle (als Proxy) eines anderen Rechners. Eigentlich sehr einfach, aber wann braucht man das und wofür?

Eine häufige Anwendung ist die Einwahl eines Rechners in ein Netzwerk. In der Abbildung 29.1 sehen Sie ein typisches Beispiel.

Der Client wählt sich per Modem in ein Netzwerk ein und verwendet das PPP-Protokoll für die Anmeldung. Über das PPP-Protokoll erhält er eine IP-Adresse, ein Default-Gateway, einen DNS-Server und einen WINS-Server mitgeteilt. In vielen Fällen handelt es sich wie hier bei der IP-Adresse um eine IP-Adresse aus dem internen Netzwerk. Wenn nun der Client ein Paket an einen Rechner in dem internen Netz schicken möchte, ist dies zunächst kein Problem, da er das Paket nur an den Remote-Access-Service-(RAS-)Server schicken kann. Wenn jedoch der Rechner A aus dem internen Netz ein Paket an den Client schicken möchte, entsteht ein Problem. Der Rechner A in dem internen Netz prüft über seine Netzmaske und die

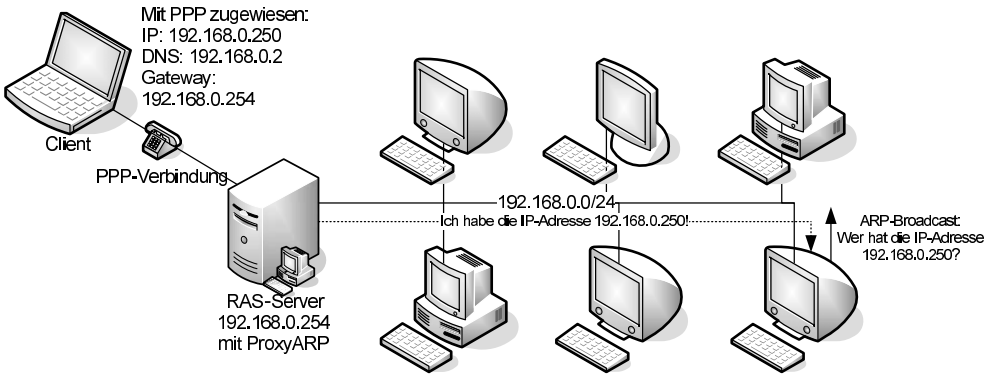


Abbildung 29.1: **Bei der Einwahl erhält der Client eine IP-Adresse aus dem internen Netzwerk.**

IP-Adresse des Clients, ob sich der Client in seinem eigenen Netz befindet. Da beide Rechner sich in dem Netzwerk 192.168.0.0/24 befinden, sendet er eine ARP-Anfrage für die IP-Adresse 192.168.0.250 aus. ARP-Anfragen werden nur in dem lokalen Netz weitergeleitet. Router wie der RAS-Server arbeiten auf der Schicht (Layer) 3 und leiten diese Pakete der Schicht 2 nicht weiter! Die ARP-Anfrage kann nie den Client erreichen, und der Client kann diese ARP-Anfrage nie beantworten. Der Rechner A erhält keine Antwort auf seine ARP-Anfrage und gibt eine Fehlermeldung aus.

ProxyARP behebt dieses Problem. Hierbei wird auf dem RAS-Server ProxyARP angeschaltet. Dann beantwortet der RAS-Server die ARP-Anfrage für den Client. Der Rechner A sendet das Paket an den RAS-Server, der nach Betrachtung der Ziel-IP-Adresse das Paket weiter an den Client sendet. Die Verbindung kommt zustande.

Mit dieser Methode können Sie an beliebigen Stellen in einem Netzwerk ohne eine Änderung der Konfiguration (IP-Adressen, Netzmaske und Gateway) Router einfügen. Diese Router können anschließend auch die gerouteten Pakete mit Iptables filtern. Im Folgenden erkläre ich zunächst die Konfiguration des ProxyARP und anschließend die Filterung der Pakete.

29.2 ProxyARP-Konfiguration

Die Konfiguration des ProxyARP beschränkt sich auf den aktuellen Kernel 2.6. In den älteren Kerneln wurde die ProxyARP-Funktionalität unterschiedlich gehandhabt (siehe `http://www.tldp.org/HOWTO/Proxy-ARP-Subnet/`).

Die Konfiguration des ProxyARP unter Linux ist sehr einfach. Beginnen Sie damit, die Netzwerkkarten in Ihrem Router zu konfigurieren. Hierbei ist es durchaus möglich, dass beide Netzwerkkarten identische IP-Adressen erhalten. Sie können aber auch die Netzwerkkarten mit unterschiedlichen IP-Adressen ausstatten:

```
# ip addr ip add 192.168.0.2 dev eth0
# ip addr ip add 192.168.0.2 dev eth1
# ip link set eth0 up
# ip link set eth1 up
```

Nun müssen Sie in der Routing-Tabelle die entsprechenden Routen eintragen. Wenn sich auf der einen Seite des ProxyARP-Routers nur ein Rechner befindet und auf der anderen Seite der Rest des Netzwerks (siehe Abbildung 29.2), dann können Sie die Routen folgendermaßen setzen:

```
# ip route add 192.168.0.3/32 dev eth1
# ip route add 192.168.0.0/24 dev eth0
```

Nun müssen Sie lediglich für jede Netzwerkkarte, die ProxyARP unterstützen soll, dieses zunächst anschalten. Das tun Sie im `/proc`-Verzeichnis. Dort existiert für jede Netzwerkkarte

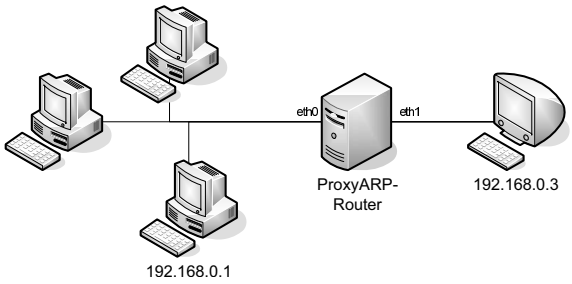


Abbildung 29.2: **Ein einfaches Beispiel, in dem sich links und rechts von einem ProxyARP-Router zwei Rechner, 192.168.0.1 und 192.168.0.3, befinden.**

ein Eintrag `/proc/sys/net/ipv4/conf/ethX/proxy_arp`. Tragen Sie hier eine Eins ein, um die Funktion anzuschalten:

```
# echo "1" > /proc/sys/net/ipv4/conf/eth0/proxy_arp
# echo "1" > /proc/sys/net/ipv4/conf/eth1/proxy_arp
```

Wenn Sie eine Option für alle Netzwerkkarten aktivieren möchten, genügt es auch, die Option in `/proc/sys/net/ipv4/conf/all` anzuschalten.

Anschließend müssen Sie noch die IP-Weiterleitung aktivieren. Eine Kommunikation sollte nun zwischen 192.168.0.1 und 192.168.0.3 möglich sein.

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

ProxyARP ist eine sehr einfache Möglichkeit, um in einem Netzwerk nachträglich ohne Änderung der Konfiguration einen Router einzuführen. Ganz transparent ist jedoch dieser Router nicht. Er wird, wie jeder andere Router, den TTL-Wert der weitergeleiteten Pakete um eins heruntersetzen. Ansonsten ist aber auf der Schicht 3 (IP) keine weitere Modifikation der Pakete erkennbar.

29.3 Filterung mit Iptables

Die Filterung mit Iptables ist beim Einsatz von ProxyARP sehr einfach. Da es sich bei dem ProxyARP-System um einen Router handelt, können Sie ganz normal die Pakete filtern. Die Pakete durchlaufen, wie auf einem normalen Router, die Ketten PREROUTING (Tabellen `Mangle` und `NAT`), FORWARD (Tabellen `Mangle` und `Filter`) und POSTROUTING (Tabellen `Mangle` und `NAT`). Sie müssen nur aufpassen, wenn Sie in Ihren Regeln IP-Adressen angeben: Achten Sie darauf, dass auf beiden Seiten der Firewall IP-Adressen aus demselben Netzwerk vorhanden sind.

Handelt es sich bei dem Rechner 192.168.0.3 aus Abbildung 29.2 um einen MySQL-Datenbankserver, den Sie zusätzlich mit einer Firewall vor dem lokalen Netz 192.168.0.0/24 schützen möchten, dann könnten Sie die folgenden Regeln verwenden:

```
MYSQL_SRV=192.168.0.3
DIENSTE="mysql,ssh"
LANDEV=eth0 # An dieser Schnittstelle ist das restliche LAN angeschlossen
MYSQLDEV=eth1 # Hier ist der MySQL-Server angeschlossen

$IPTABLES -P FORWARD DROP

$IPTABLES -A FORWARD -i $LANDEV -o $MYSQLDEV -d $MYSQL_SRV -p tcp -m ↵
        multiport --dport $DIENSTE -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

29.4 Fazit

So können Sie mit einfachen Mitteln auch den Verkehr in einem Netz filtern, ohne Änderungen an den Adressen, Netzmasken und Default-Gateways vornehmen zu müssen.

Um jedoch tatsächlich die Firewall ohne IP-Adressen zu betreiben, benötigen Sie den Bridge-Modus. Das nächste Kapitel erläutert diesen und die Anwendung von Iptables.

30. Iptables auf einer Bridge

Eine Bridge ist eine Software, die ähnliche Funktionen übernimmt wie ein Switch, der in Hardware implementiert wurde. Die Bridge verfügt häufig über zusätzliche Möglichkeiten, wie zum Beispiel die Änderung des Mediums. So gibt es Bridges, die Token-Ring- und Ethernet-Netze miteinander verbinden können. Linux kann derartige Bridges betreiben und auch mit Iptables die von der Bridge weitergeleiteten Pakete filtern.

30.1 Wie funktioniert die Bridge?

Wie funktioniert nun eine Bridge? Eine Bridge entspricht einem Switch in Software. Nach Aktivierung der Bridge lernt die Bridge ähnlich wie ein Switch die MAC-Adressen der Rechner, die an der Bridge angeschlossen sind. Dazu betrachtet die Bridge bei jedem Paket die Absender-MAC-Adresse und speichert diese in einer Tabelle ab (siehe Abbildung 30.1).

Solange die Bridge die Ziel-MAC-Adresse eines Pakets noch nicht kennt, kopiert die Bridge dieses Paket in alle angeschlossenen Netze, sodass auf jeden Fall eine Kommunikation möglich ist. Sobald aber die Bridge die Ziel-MAC-Adresse bereits kennt, ermittelt sie mit ihrer Tabelle die Netzwerkkarte, an der der entsprechende Rechner angeschlossen ist, und leitet das Paket auf dieser Netzwerkkarte weiter. So unterdrückt die Bridge Broadcasts und ver-

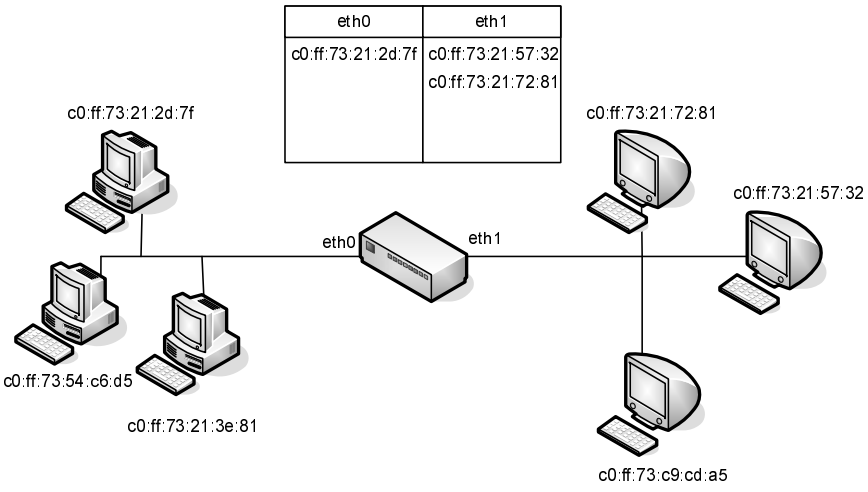


Abbildung 30.1: **Eine Bridge lernt selbstständig die MAC-Adressen der angeschlossenen Geräte.**

meidet Kollisionen in dem Netzwerk durch gleichzeitig gesendete Pakete. Damit erhöht die Bridge die Leistung des Ethernet-Netzwerks, das bei einer Kollision die Pakete erneut senden muss.

30.2 Bau einer Bridge mit Linux

Um mit einem Linux-System eine Bridge zu bauen, benötigen Sie lediglich einen Kernel 2.4 oder 2.6. Für die älteren Kernel benötigen Sie zusätzlich den Bridge-Patch von `http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge`. Für die Konfiguration der Bridge müssen Sie die Bridge-Utilities installieren. Diese sind in den meisten Distributionen enthalten. Das Paket heißt üblicherweise `bridge-utils`.

Der wesentliche Befehl der Bridge-Utilities ist `brctl`. Hiermit steuern Sie die gesamte Bridge:

```
[root@bibo ~]# brctl
commands:
     addbr           <bridge>                add bridge
     delbr           <bridge>                delete bridge
     addif           <bridge> <device>       add interface to bridge
     delif           <bridge> <device>       delete interface from bridge
     setageing       <bridge> <time>         set ageing time
     setbridgeprio   <bridge> <prio>         set bridge priority
     setfd           <bridge> <time>         set bridge forward delay
     sethello        <bridge> <time>         set hello time
     setmaxage       <bridge> <time>         set max message age
     setpathcost     <bridge> <port> <cost>  set path cost
     setportprio     <bridge> <port> <prio>  set port priority
     show                                    show a list of bridges
     showmacs        <bridge>                show a list of mac addrs
     showstp         <bridge>                show bridge stp info
     stp             <bridge> <state>        turn stp on/off
```

Um nun eine Bridge zu erzeugen, verwenden Sie den Befehl `brctl addbr br0`. Das Gerät `br0` ist anschließend sofort verfügbar:

```
[root@bibo ~]# ip link show br0
4: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop link/ether ↩
        00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
```

Nun können Sie der Bridge Netzwerkkarten hinzufügen. Hierzu verwenden Sie den Befehl `brctl addif`. Dabei ist es wichtig, dass diese Netzwerkkarten keine IP-Adressen tragen. Dann können Sie mit dem `ip`-Befehl die Netzwerkkarten aktivieren.

```
[root@bibo ~]# brctl addif br0 eth0
[root@bibo ~]# brctl addif br0 eth1
[root@bibo ~]# ip link set eth0 up
[root@bibo ~]# ip link set eth1 up
```

```
[root@bibo ~]# ip link set br0 up
[root@bibo ~]# ip addr add dev br0 192.168.0.5/24
[root@bibo ~]# ip link show
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast
qlen 100
    link/ether 00:20:e0:6c:72:1e brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast
qlen 100
    link/ether 00:10:a4:c3:26:cb brd ff:ff:ff:ff:ff:ff
4: br0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
    link/ether 00:10:a4:c3:26:cb brd ff:ff:ff:ff:ff:ff
```

Wenn bei Ihnen der `ip link show`-Befehl nicht das `PROMISC`-Flag bei den Netzwerkkarten anzeigt, arbeiten Sie wahrscheinlich auf einem Linux-Kernel 2.6. Hier wird dieses Flag nicht mehr angezeigt.

Mit dem Befehl `brctl` können Sie sich die Einzelheiten der Bridge anzeigen lassen:

```
[root@bibo ~]# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.0010a4c326cb       yes             eth0
                                                        eth1
```

Wenn Sie die Bridge für Firewall-Zwecke einsetzen möchten, sollten Sie das Spanning-Tree-Protokoll (STP) abschalten. Dies ermöglicht den Aufbau von redundanten Bridges. Da das Protokoll aber keine Sicherheit bietet, sollte es im Zusammenhang mit Firewalls nicht genutzt werden: `brctl stp br0 off`. Dann können Sie auch die Forward-Delay-Zeit auf null setzen. Bei dem Einsatz des Spanning-Tree-Protokolls werden hiermit Schleifen zum Startzeitpunkt der Bridge ausgeschlossen: `brctl fd br0 0`.

Nun sollte bereits eine Kommunikation über die Bridge möglich sein. Dazu können Sie bereits durch die Bridge pingen. Anschließend können Sie sich die MAC-Adressen anzeigen lassen, die die Bridge bereits gelernt hat:

```
[root@bibo ~]# brctl showmacs br0
port no mac addr                is local?       ageing timer
  1     00:20:e0:6c:72:1e       yes                0.00
  2     00:20:e0:6c:72:10       yes                0.00
  1     00:20:e0:73:71:20       yes                0.00
  2     00:20:e0:73:71:21       yes                0.00
```


CEP Hier „Tabelle“ entfernt. Korrekt?

Editor's or typesetter's annotations (will be removed before the final TeX run)

30.3 Filtern auf der Bridge mit iptables

Nun können Sie wie üblich auf der Bridge filtern. Die Pakete durchlaufen nacheinander die `mangle-PREROUTING`-, CE P die `nat-PREROUTING`-, die `mangle-FORWARD`-, die `filter-FORWARD`-, die `mangle-POSTROUTING`- und die `nat-POSTROUTING`-Kette.

Auf modernen Kerneln können Sie entscheiden, ob die Firewall-Ketten durchlaufen werden sollen. Hierzu existieren in `/proc/sys/net/bridge` die folgenden Dateien:

- `bridge-nf-call-iptables`
- `bridge-nf-call-ip6tables`
- `bridge-nf-call-arptables`

Damit die Filterung erfolgt, müssen diese Dateien den Wert 1 enthalten.

STOP

Ich habe bei mir auf einigen Kerneln feststellen müssen, dass die Pakete die `Mangle-PREROUTING`-Kette teilweise dreimal durchlaufen. Dies scheint ein Artefakt zu sein.

Wenn Sie nun die Pakete filtern möchten, können Sie das ganz normal in der `FORWARD`-Kette tun. Allerdings sollten Sie darauf achten, dass die Bridge, während sie die MAC-Adressen lernt, auch noch Pakete weiterleitet, die nicht weitergeleitet werden müssen. Das bedeutet, dass Sie mit `REJECT`-Regeln sehr vorsichtig sein sollten. In Abbildung 30.2 ist ein Szenario gezeigt, in dem eine REJECT-Regel zum Abbruch gültiger Verbindungen führt. Hier baut der Rechner A zum Rechner B in demselben Netzwerk eine Verbindung auf. Die Bridge sieht ebenfalls das Paket und muss es weiterleiten, da sie die Ziel-MAC-Adresse noch nicht kennt. Es durchläuft die `FORWARD`-Kette und wird abgelehnt. Da die Ablehnung durch eine `REJECT`-Regel erfolgt, sendet die Bridge einen Fehler an Rechner A, obwohl die Verbindung sehr wohl aufgebaut werden darf.

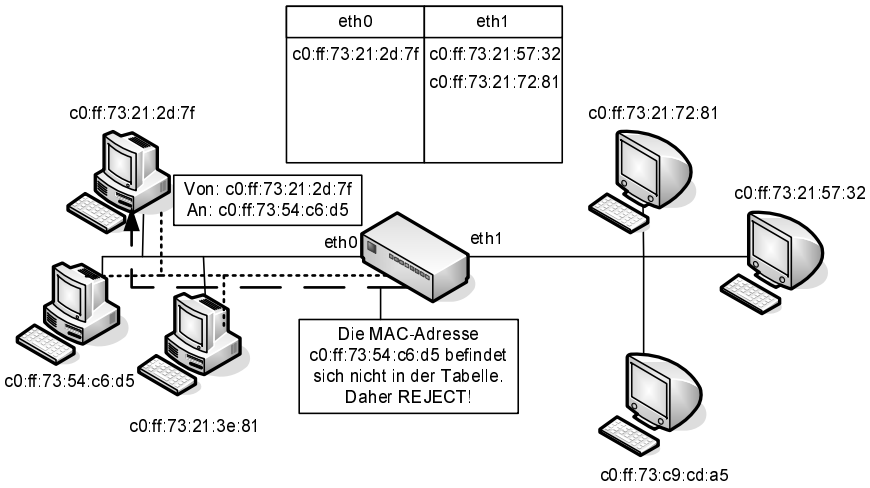


Abbildung 30.2: **Ein REJECT kann auf einer Bridge Probleme erzeugen.**

Bei der Definition der Regeln ist es häufig sinnvoll zu prüfen, über welche Netzwerkkarte der Bridge das Paket die Bridge erreicht. Hierfür verwenden Sie bei dem Linux-Kernel 2.4 wie gewohnt die Optionen `-i` bzw. `--in-interface` und `-o` bzw. `--out-interface`. Bei dem Linux-Kernel 2.6 wurden hierfür neue Optionen geschaffen, da insgesamt die Firewall-Möglichkeiten auf der Bridge erweitert wurden (siehe Kapitel 31). Diese Optionen heißen `--physdev-in` und `--physdev-out`. Um diese Optionen zu nutzen, müssen Sie die Erweiterung `physdev` in Ihren Regeln laden. Ein Beispiel zeigt die Anwendung:

```
$IPTABLES -A FORWARD -m physdev --physdev-in eth0 --physdev-out eth1 -d ↵
        192.168.0.1 -j ACCEPT
```

Die weiteren Optionen (`--physdev-is-in`, `--physdev-is-out` und `--physdev-is-bridged`) der `physdev`-Erweiterung werden nur selten benötigt und erklären sich von selbst.

Wenn Sie mit einer Bridge ein Netzwerk in zwei Kollisionsdomänen aufgeteilt haben und nun auf der Bridge filtern möchten, ist häufig die `ipset`-Erweiterung des Linux-Kernels hilfreich. Damit ist es sehr leicht möglich, mehrere nicht zusammenhängende IP-Adressen in einer Regel zusammenzufassen. Dies reduziert den Regelaufwand und erlaubt das Erzeugen übersichtlicher Regelwerke. Sie finden die Informationen über diese Erweiterungen in Kapitel 26.

30.4 Filtern auf der Bridge mit arptables

Der Befehl `arptables` ist in den meisten modernen Distributionen enthalten. Er erlaubt die Filterung der Address-Resolution-Protocol-Pakete (ARP), die das IPv4-Protokoll benötigt, um ausgehend von einer IP-Adresse die MAC-Adresse eines Rechners zu ermitteln.

Eine Filterung der ARP-Pakete auf einem Router oder einem einzelnen Rechner ist ungewöhnlich, da dies nur dazu führt, dass ein Rechner in dem IPv4-Netzwerk scheinbar unsichtbar wird.

Wesentlich interessanter ist die Anwendung auf einer Bridge, die im Gegensatz zu einem Router ARP-Pakete weiterleitet. Hier kann durch eine sinnvolle Filterung der ARP-Pakete erreicht werden, dass die von der Firewall-Bridge geschützten Systeme auch auf der Ebene des ARP-Protokolls nur für bestimmte Clients erreichbar sind.

Der `arptables`-Befehl kann auf einem Linux-Kernel 2.6 drei Ketten in der ARP-Filter-Tabelle verwalten. Diese Ketten verwalten lediglich die ARP-Pakete. Die `IN`-Filter-Kette betrachtet alle ARP-Pakete, die an den Rechner gerichtet sind. Die `OUT`-Filter-Kette betrachtet alle lokal erzeugten ARP-Pakete, während die `FORWARD`-Filter-Kette auf einer Bridge alle von der Bridge durchgeleiteten ARP-Pakete betrachtet.

So ist es möglich, dass nur bestimmte Clients eine ARP-Anfrage für eine bestimmte IP-Adresse senden dürfen. Mit `arptables` können Sie das mit dem folgenden Befehl erreichen:

```
ARPT=/usr/sbin/arptables
$ARPT -A FORWARD -s 192.168.0.7 -d 192.168.0.25 -j ACCEPT
$ARPT -A FORWARD -d 192.168.0.25 -j DROP
```

30.5 Fazit

Dieser Abschnitt hat Ihnen gezeigt, wie Sie sehr einfach mit `iptables` auf einer Bridge die Pakete filtern können. So können Sie sehr einfach auf der IP-Ebene transparente Paketfilter aufbauen. Jedoch möchten Sie vielleicht eine noch speziellere Funktion implementieren. Es besteht die Möglichkeit, einige Pakete (zum Beispiel alle IP-Pakete) zu routen und andere Pakete (zum Beispiel NETBEUI-Pakete) über die Bridge zu senden und dort zu filtern. Hierzu benötigen Sie den Befehl `ebtables`, und die entsprechenden Funktionen müssen in Ihrem Kernel aktiviert worden sein. Das nächste Kapitel erklärt die Funktion.

31. ebtables

Der `ebtables`-Befehl ist ein zusätzlicher Befehl zur Filterung von Netzwerkpaketen auf einer Linux-Bridge. Mit diesem Befehl können Sie auch Nicht-IP-Pakete eingeschränkt filtern. Außerdem können Sie entscheiden, ob ein Paket geroutet oder gebridget werden soll. Der Aufbau eines kombinierten Routers mit Bridge-Funktionalität ist für Nicht-IP-Pakete möglich. Derartige Systeme werden als Brouter bezeichnet.

Die Anwendung des Befehls erfordert zusätzliche Patches für die 2.4er-Linux-Kernel. Diese Patches sind in den 2.6er-Kerneln bereits enthalten und bei den meisten Distributionen auch aktiviert. Viele Distributionen liefern jedoch den Befehl selbst nicht mit. Daher wird zunächst die Installation des Befehls beschrieben.

31.1 Ebtables-Installation

Die letzte Ebtables-Version 2.0.9 stammt vom Februar 2010. Falls Ihre Distribution bereits Ebtables enthält, handelt es sich wahrscheinlich um diese Version, und ein Update ist nicht erforderlich. Für den Fall, dass Ihre Distribution das Paket nicht enthält und Sie möglicherweise auch noch den Kernel 2.4 einsetzen, schildere ich im Weiteren kurz die Installation des Pakets.

31.1.1 Konfiguration des Linux-Kernels

Während der Linux-Kernel 2.6 bereits den notwendigen Code enthält, müssen Sie den Linux-Kernel 2.4 noch patchen. Unter `http://ebtables.sf.net` finden Sie für den Linux-Kernel 2.4 die notwendigen Patches. Um den Patch anzuwenden, wechseln Sie in das Quelltextverzeichnis Ihres Kernels und rufen den Befehl `zcat patch.gz | patch -p1` auf. Anschließend müssen Sie Ihren Kernel noch konfigurieren. Zusätzlich zu den üblichen Iptables-Optionen wählen Sie die Option `802.1d Ethernet Bridging` an. Dann erscheint unter dieser Option die Option `Bridge: ebtables`. Wählen Sie diese und die anschließend erscheinenden Optionen ebenfalls aus.

Den Linux-Kernel 2.6 müssen Sie nicht mit einem Patch vorbereiten. Hier müssen Sie nur darauf achten, dass die entsprechenden Optionen ausgewählt wurden. Auch hier benötigen Sie die Option `802.1d Ethernet Bridging`, und unter `Bridge: Netfilter configuration` wählen Sie die gewünschten Ebtables-Funktionalitäten.

31.1.2 Installation des Userspace-Werkzeugs

Um den Befehl `ebtables` zu installieren, laden Sie zunächst das Paket von der Homepage und packen es aus. Anschließend übersetzen Sie das Paket mit dem Befehl `make`. Bei dem anschließenden `make install` können Sie mit den Variablen `KERNEL_INCLUDES`, `LIBDIR`, `MANDIR`, `BINDIR`, `ETCDIR`, `ETHERTYPEPATH` und `DESTDIR` die Zielverzeichnisse angeben.

31.2 Die Ebtables-Tabellen

Ebtables verfügt über insgesamt 3 Tabellen. Die `broute`-Tabelle enthält die `BROUTING`-Kette. Die `filter`-Tabelle enthält die Ketten `FORWARD`, `INPUT` und `OUTPUT`. Die `nat`-Tabelle schließlich enthält die `PREROUTING`-, `OUTPUT`- und `POSTROUTING`-Ketten. Abbildung 31.1 zeigt die Ketten.

Die Ebtables-Tabellen haben keinerlei Bezug zu den Iptables-Tabellen und -Ketten. Es handelt sich um vollkommen eigenständige Tabellen und Ketten. Die `filter`- und die `nat`-`OUTPUT`-Ketten sind ebenfalls eigenständig und werden nacheinander (zuerst `nat`, dann `filter`) durchlaufen.

Mit der `BROUTING`-Kette der `broute`-Tabelle können Sie einen Brouter aufbauen. Ein Brouter ist nach Definition des Maintainers des Ebtables-Codes ein System, das zwei Netze verbindet und einige Pakete routet (zum Beispiel IP-Pakete), während es andere Frames (zum Beispiel NETBEUI-Frames) über die Bridge weiterleitet. Dabei können Sie in der `BROUTING`-Kette entscheiden, ob Sie die Frames/Pakete routen oder bridgen möchten.

Wie bereits weiter oben ausgeführt wurde, erfolgt das Bridging auf der Schicht 2 (Data-Link) des OSI-Modells. Dies ist die Schicht 1 des TCP-Modells. Das Routing erfolgt auf der Schicht 3 des OSI-Modells beziehungsweise auf der Schicht 2 des TCP-Modells. Pakete auf der Data-Link-Schicht des OSI-Modells werden üblicherweise als Rahmen (Frame) bezeichnet, während der Begriff „Paket“ für die Netzwerkschicht (Schicht 3) reserviert ist. Ich werde im Folgenden versuchen, diese beiden Begriffe korrekt zu verwenden.

Wie durchlaufen nun die Frames/Pakete die einzelnen Ketten? Um dies nachzuvollziehen, ist es sinnvoll, einige Szenarien durchzuspielen.

31.2.1 Der Rahmen erreicht das System über ein Bridge-Interface

Wenn das Paket über ein Bridge-Interface das System erreicht, wird in dem System aus Abbildung 31.2 das Paket an die `BROUTING`-Kette weitergegeben.

Hier können Sie mit einer Regel entscheiden, ob das Paket geroutet oder der Rahmen gebridget werden soll. Die Default-Aktion ist das Bridging. Bleibt der Rahmen in dem Bridging-Code, so wird er an die `PREROUTING`-Ketten weitergegeben. Hier werden nacheinander die Ebtables- und die Iptables-`PREROUTING`-Ketten durchlaufen. Anschließend erfolgt die Prüfung, ob das Paket gebridget werden muss oder ob sein Ziel das lokale System ist. Ist Letzteres der Fall, so wird der Rahmen an die Ebtables-`INPUT`-Kette weitergereicht und anschließend weiter an die Netzwerkschicht geschickt. Dort wird dann die Routen-Entscheidung getroffen.

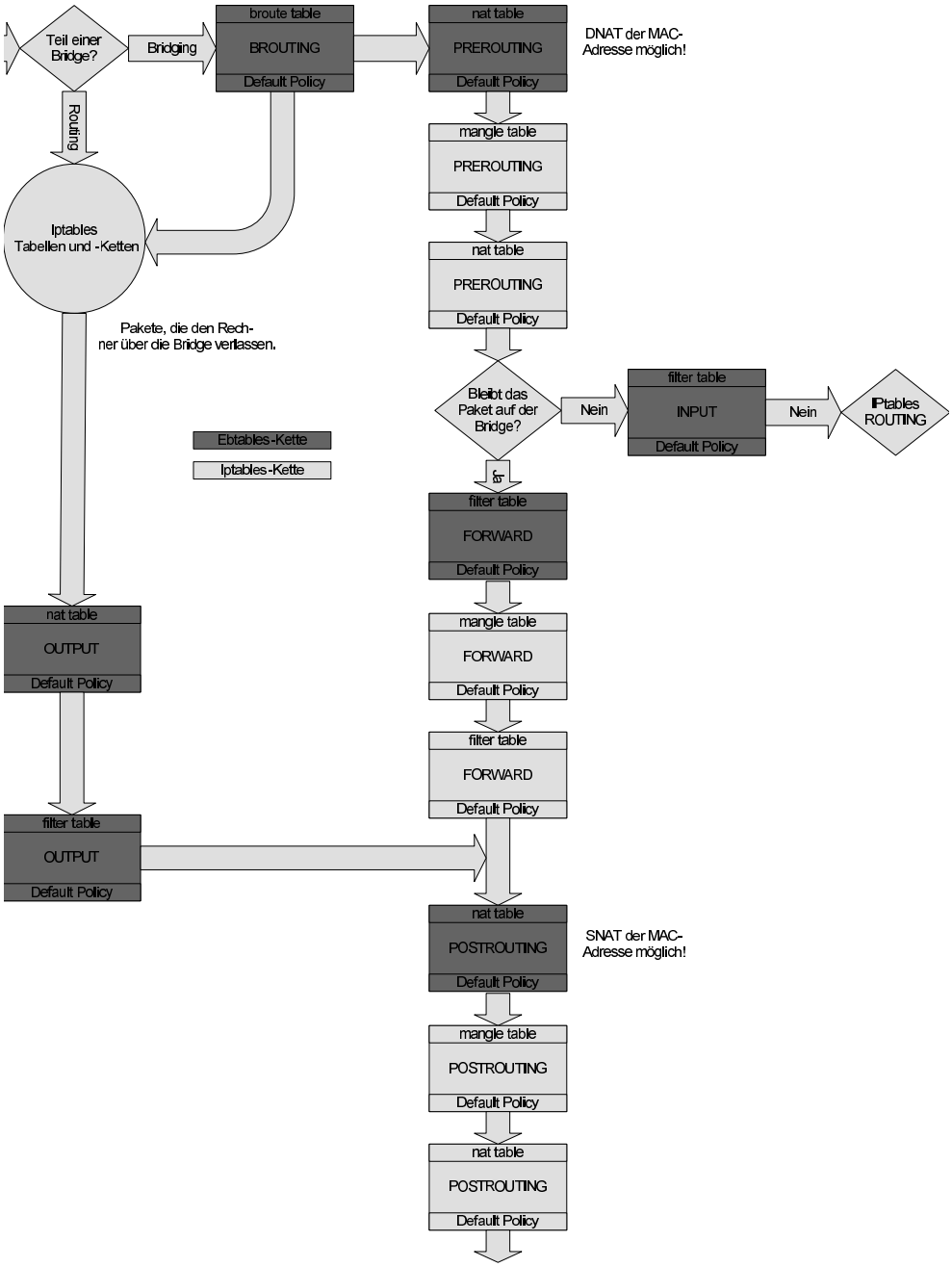


Abbildung 31.1: **Ebtables verfügt über drei eigene Tabellen.**

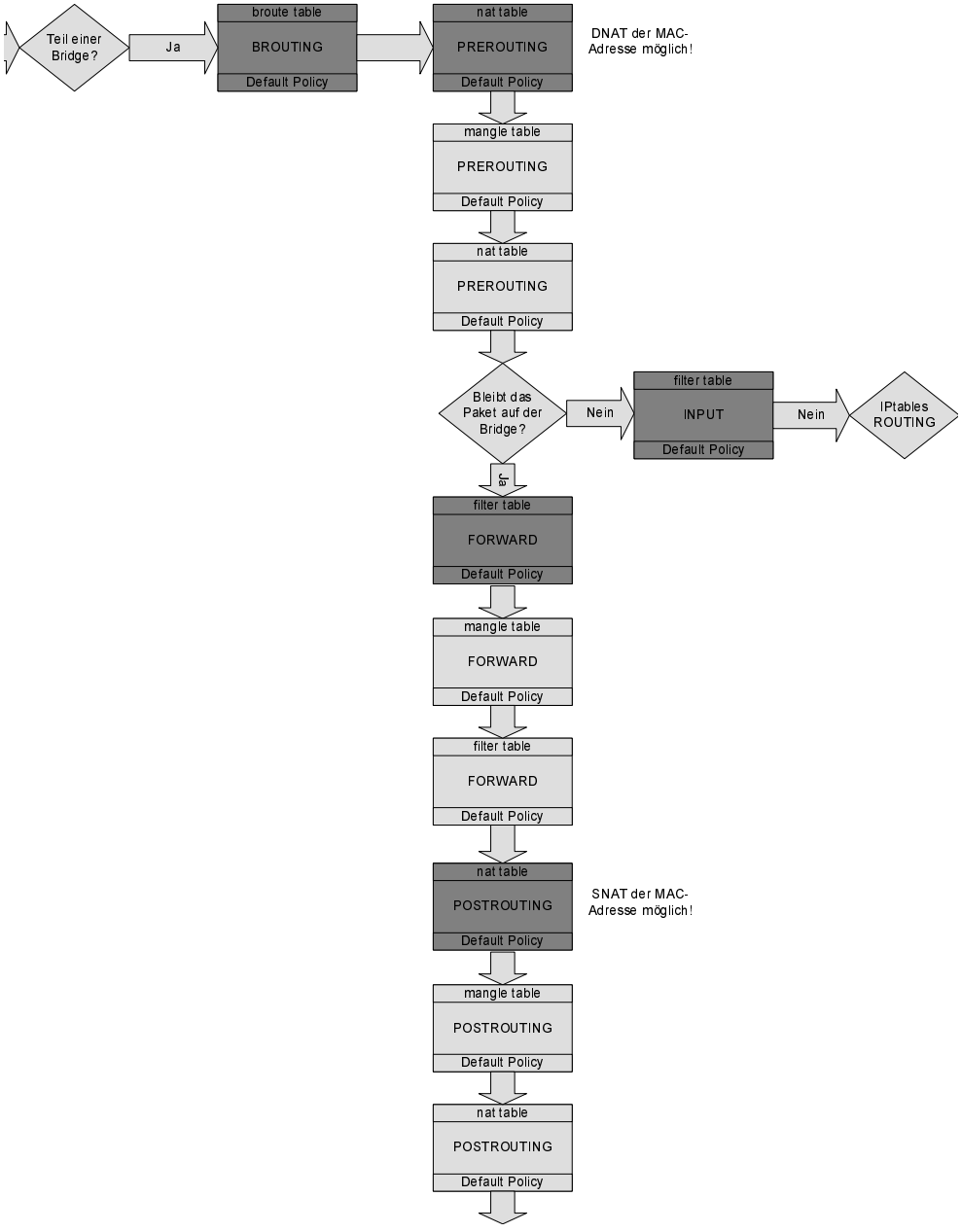


Abbildung 31.2: **Die Ebtables- und Iptables-Ketten werden nacheinander durchlaufen.**

Muss der Rahmen gebridget werden, so wird er durch die Ebtables- und Iptables-FORWARD-Ketten und die entsprechenden POSTROUTING-Ketten an das entsprechende Interface gesendet.

31.2.2 Der Rahmen erreicht das System über ein Nicht-Bridge-Interface

Falls das Paket über ein Bridge-Interface das System verlassen soll, durchläuft das Paket ganz normal die mangle- und nat-PREROUTING-Ketten. Wird das Paket geroutet, merkt der Kernel, dass das Paket über ein gebridgtes Interface den Rechner verlässt. Das Paket durchläuft die mangle- und filter-FORWARD-Ketten von Iptables und wird dann nach der Bridging-Entscheidung über die Ebtables-nat- und filter-OUTPUT-Ketten an die Ebtables-POSTROUTING- und die Iptables-POSTROUTING-Ketten gesendet.

31.2.3 Ein lokal erzeugtes Paket verlässt das System über die Bridge

Dies ist der dritte mögliche Fall. Hierbei erzeugt ein lokaler Prozess ein Paket. Dies wird zunächst von den Iptables-OUTPUT-Ketten der mangle-, nat- und filter-Tabellen geprüft. Anschließend wird es über die Ebtables-OUTPUT-Ketten wie im letzten Fall transportiert.

31.3 Die broute-Tabelle

In manchen Umgebungen kann es sinnvoll sein, eine Bridge mit einem Router auf denselben Netzwerkkarten zu kombinieren (siehe Abbildung 31.3).

Dies kann zum Beispiel dann interessant sein, wenn Sie noch einige nicht routing-fähige Protokolle in Ihrem Netzwerk einsetzen. Dies können zum Beispiel NETBEUI, Appletalk oder DEC-LAT sein. Während ein normaler IP-Router diese Protokolle nicht weiterleiten würde, kann dies eine Bridge für diese Rahmen leisten.

In der BROUTING-Kette der broute-Tabelle können Sie entscheiden, ob ein Rahmen, der normalerweise gebridget werden würde, doch geroutet werden soll. Hierfür müssen Sie hier eine Regel hinzufügen, die das entsprechende Paket verwirft (DROP). Alle Pakete, die in der BROUTING-Kette verworfen werden, verlassen die Data-Link-Schicht und werden an den Routing-

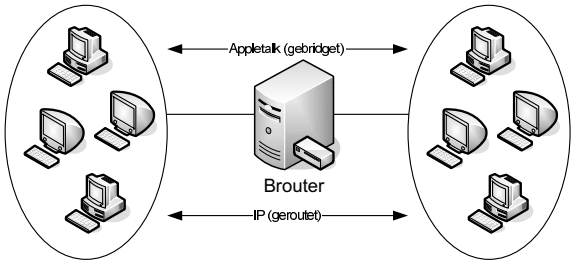


Abbildung 31.3: **Ein Brouter routet einige Pakete, während er andere Rahmen bridget.**

Prozess und die Netzwerk-Schicht weitergegeben. Um zum Beispiel IP-Pakete zu routen, ARP-Anfragen zu verwerfen und alle weiteren Rahmen über die Bridge weiterzuleiten, können Sie folgendes Skript verwenden:

```
EBTABLES=/sbin/ebtables
BRCTL=/usr/sbin/brctl
IP=/sbin/ip

$EBTABLES -t filter -F
$EBTABLES -t nat -F
$EBTABLES -t broute -F

$BRCTL addbr br0
$BRCTL addif br0 eth0
$BRCTL addif br0 eth1

$IP addr add 172.16.1.1/24 dev eth0
$IP addr add 172.16.2.1/24 dev eth1
$IP link set eth0 up
$IP link set eth1 up
$IP link set br0 up

$EBTABLES -t broute -A BROUTING -p IPv4 -j DROP
$EBTABLES -t broute -A BROUTING -p ARP -j DROP
```

31.4 Die ebtables-Syntax

Der `ebtables`-Befehl verhält sich sehr ähnlich wie der `Iptables`-Befehl. Jedoch gibt es einige wesentliche Unterschiede, die hier kurz zusammengefasst werden sollen. Außerdem gibt es bei dem `ebtables`-Befehl einige zusätzliche Matches und Targets, die ich im Weiteren erläutern werde.

Zunächst weist der Befehl `ebtables` dieselben Kommandos wie der Befehl `Iptables` auf. Es gibt im Einzelnen:

- `-A, --append`: Anhängen einer Regel
- `-D, --delete`: Löschen einer Regel
- `-P, --policy`: Ändern der Kettenrichtlinie
- `-F, --flush`: Löschen einer Tabelle oder Kette
- `-Z, --zero`: Löschen der Zähler einer Tabelle oder Kette
- `-L, --list`: Anzeige der Regeln. Hier gibt es einen Unterschied. Zunächst zeigt dieser Befehl die Regeln in einem anderen Format an:

```
# ebtables -t broute -L
Bridge table: broute

Bridge chain: BROUTING, entries: 2, policy: ACCEPT
-p IPv4 -j DROP
-p ARP -j DROP
```

Um zusätzlich auch Zeilennummern angezeigt zu bekommen, müssen Sie auch die Option `--Ln` angeben. Die Rahmen- und Byte-Zähler erhalten Sie mit `--Lc`:

```
# ebtables -t broute -L --Ln --Lc
Bridge table: broute

Bridge chain: BROUTING, entries: 2, policy: ACCEPT
1. -p IPv4 -j DROP , pcnt = 1016 -- bcnt = 320521
2. -p ARP -j DROP , pcnt = 3 -- bcnt = 138
```

Wenn Sie die Regeln so anzeigen möchten, dass Sie diese direkt in ein Skript übernehmen können, müssen Sie die Option `--Lx` bei dem Befehl spezifizieren:

```
# ebtables -t broute -L --Lx
ebtables -t broute -A BROUTING -p IPv4 -j DROP
ebtables -t broute -A BROUTING -p ARP -j DROP
```

Schließlich zeigt die Option `-Lmac2` die MAC-Adressen an.

» `-N, --new-chain`: Erzeugt eine benutzerdefinierte Kette.

» `-X, --delete-chain`: Löscht eine benutzerdefinierte Kette.

» `-E, --rename-chain`: Benennt eine benutzerdefinierte Kette um.

» `--init-table`: Initialisiert die Tabelle. Dies löscht alle Regeln und setzt die Richtlinie (Policy) auf den Default zurück.

» `--atomic-init`: Das Kommando `ebtables` kann die Regeln einer Tabelle aus einer Datei lesen. Außerdem kann jeder Befehl auch auf eine Datei angewendet werden (siehe `--atomic-file`[TS 9]). Mit diesem Befehl initialisieren Sie die Datei.

» `--atomic-save`: Dieses Kommando speichert den aktuellen Zustand einer Ebtables-Tabelle in der angegebenen Datei.

» `--atomic-commit`: Dieses Kommando lädt die Regeln aus der angegebenen Datei. Dies ist ein atomarer Vorgang. Die Regeln werden alle gleichzeitig geladen! Die Datei kann mit dem Kommando `--atomic-save` oder `--atomic-init` erzeugt werden.

Sie können für die Verwendung mit dem Ebtables-Befehl zunächst alle Regeln in einer Datei aufbauen und dann atomar in einem einzigen Schritt aktivieren. So stellen Sie sicher, dass kein Paket ungefiltert Ihre Bridge passiert. Diese Gefahr besteht, wenn Sie zunächst die Tabelle löschen und dann inkrementell die Regeln aufbauen. Das folgende Listing zeigt beispielhaft, wie Sie die Regeln aufbauen können:

[TS 9] Den Anstrich zu `--atomic-file` gibt es nicht.

```
$EBTABLES --atomic-file filter_table -t filter --atomic-init
$EBTABLES --atomic-file filter_table -A FORWARD -s 00:11:22:33:44:55 -p ↵
        IPV4 -j
          ACCEPT
# ... weitere Befehle

# Lade die Tabelle
$EBTABLES --atomic-file filter_table -t filter --atomic-commit
```

Wenn Sie nicht in jeder Zeile die Datei angeben möchten, können Sie auch die Umgebungsvariable `EBTABLES_ATOMIC_FILE` verwenden.

Um die Rahmen in den Regeln zu prüfen, verfügt Ebtables über eingebaute Tests (Matches) und über Erweiterungen (Match-Extensions und Watcher-Extensions). Diese werden im Folgenden kurz erläutert.

Zunächst können Sie das Protokoll des Rahmens mit der Option `-p, --protocol` prüfen. Hierbei handelt es sich nicht um ein Protokoll wie TCP oder UDP. Denken Sie daran, dass Sie hier einen Ethernet-Frame betrachten. Mögliche Protokolle sind IPv4, ARP und NetBEUI. Eine Liste aller möglichen Protokolle finden Sie in der Datei `/etc/ethertypes`. Das Protokoll kann sowohl mit seinem Namen als auch hexadezimal (IPv4 = 0x0800) angegeben werden. Einige Protokolle (802.2) verwenden dieses Feld als Längenangabe. Sobald der Wert in diesem Feld kleiner als 0x600 ist, handelt es sich um eine Länge. Dann verwenden Sie hier bitte `LENGTH`.

Natürlich kann Ebtables auch die Netzwerkkarten prüfen, über die das Paket die Bridge erreicht hat. Hier wird jedoch zwischen dem physikalischen Interface (z. B. `eth0`) und dem logischen Interface (z. B. `br0`) unterschieden. Die eingehenden Interfaces können Sie in den Ketten `INPUT`, `FORWARD`, `PREROUTING` und `BROUTING` testen, während Sie die ausgehenden Interfaces in den Ketten `OUTPUT`, `FORWARD` und `POSTROUTING` testen können. Die Optionen heißen:

- `-i, --in-if, --in-interface`
- `--logical-in`
- `-o, --out-if, --out-interface`
- `--logical-out`

Um die MAC-Adressen des Rahmens zu prüfen, gibt es die Optionen `-s, --src, --source` beziehungsweise `-d, --dst, --destination`. Die Adresse wird als hexadezimale Adresse mit Doppelpunkten geschrieben: `00:50:56:C0:00:03`. Beide Optionen erlauben auch die Angabe einer Netzmaske. Alternativ können Sie als Absenderadresse auch `Unicast`, `Broadcast`, `Multicast` oder `BGA` (Bridge Group Address) angeben. Um diese Typen als Zieladresse zu prüfen, existiert die Option `--pkt-type` (siehe unten).

Der Ebtables-Befehl verfügt über eine Reihe von Erweiterungen, die für die Prüfung von Rahmen genutzt werden können. Diese Erweiterungen müssen nicht wie bei `iptables` mit der Option `-m` geladen werden. Sie werden jedoch in eigenen Kernel-Modulen implementiert.

Für die Prüfung von 802.2- oder 802.3-Rahmen stehen die Optionen `--802_3-sap` und `--802_3-type` zur Verfügung.

Für das ARP- und das RARP-Protokoll können Sie die folgenden Optionen verwenden. Diese Optionen sind nur gültig, wenn Sie auch als Protokoll (`-p`) ARP oder RARP spezifiziert haben.

» `--arp-opcode`: Diese Option gibt den ARP-Operation-Code an. Insgesamt existieren neun verschiedene Opcodes:
  - Request (1)
  - Reply (2)
  - Request_Reverse (3)
  - Reply_Reverse (4)
  - DRARP_Request (5)
  - DRARP_Reply (6)
  - DRARP_Error (7)
  - InARP_Request (8)
  - ARP_NAK (9)

  Diese Opcodes können als Zeichenkette oder numerisch angegeben werden.

» `--arp-htype`: Der Hardware-Typ. Eigentlich immer Ethernet (1).

» `--arp-ptype`: Der Protokoll-Typ. Eigentlich immer IPv4 (0x0800).

» `--arp-ip-src`, `--arp-ip-dst`: Die IP-Absender- beziehungsweise IP-Zieladresse des ARP-Pakets.

» `--arp-mac-src`, `--arp-mac-dst`: Die MAC-Absender- beziehungsweise MAC-Zieladresse des ARP-Pakets.

Wenn Sie als Protokoll (`-p`) IPv4 in der Regel spezifizieren, dürfen Sie die folgenden Optionen nutzen, um rudimentär das IP-Paket zu prüfen. Diese Optionen erreichen bei Weitem nicht die Mächtigkeit des `iptables`-Kommandos.

So können Sie mit den Optionen `--ip-source`, `--ip-src` und `--ip-destination`, `--ip-dst` die Absender- und Ziel-IP-Adresse prüfen. Mit `--ip-tos` kann durch Angabe der hexadezimalen Nummer der Type-of-Service geprüft werden. Die Option `--ip-protocol`, `--ip-proto` erlaubt die Prüfung des IP-Protokolls (z. B. TCP oder UDP). Handelt es sich um ein TCP- oder UDP-Paket, können Sie mit `--ip-source-port`, `--ip-sport` und `--ip-destination-port`, `--ip-dport` die Ports testen.

Ebtables unterstützt genauso wie Iptables die Markierung von Paketen. Mit der Option `--mark` können Sie diese Markierung testen.

Die Option `--pkt-type` gibt Ihnen die Möglichkeit, den Pakettyp zu testen. Mögliche Typen sind `broadcast`, `multicast`, `host` und `otherhost`. Ist das Paket an den lokalen Rechner gerichtet, so ist der Typ `host`. Alle anderen Unicast-Pakete sind `otherhost`.

Für die Prüfung von Spanning-Tree-Protokoll-Rahmen (STP) gibt es eine Vielzahl von Optionen, die hier nur verwirren würden. Für Kenner des Protokolls sei nur erwähnt, dass sämt-

liche in der Bridge Protocol Data Unit (BPDU) übertragenen Informationen, wie zum Beispiel die Priorität der Root-Bridge, mit entsprechenden Optionen getestet werden können. Auf der Manpage finden Sie weitere Informationen.

Auch das VLAN können Sie hier prüfen. Hierzu stehen die Optionen `--vlan-id`, `--vlan-prio` und `--vlan-encap` zur Verfügung.

Um Pakete zu protokollieren, existiert die Option `--log`. Die Protokollierung ist bei Ebtables im Gegensatz zu Iptables eine Option und kein Ziel (Target). Die Option `--log` protokolliert das Paket mit den Defaultwerten für den Level, das Präfix und ohne IP- und ARP-Informationen. Wenn Sie diese Defaultwerte ändern möchten, nutzen Sie anstelle der Option `--log` die Optionen `--log-prefix`, `--log-level`, `--log-ip` und `--log-arp`.

Sobald eine Regel auf einen Rahmen zutrifft, kann Ebtables verschiedene Aktionen ausführen. Zusätzlich zu den von Iptables bekannten Aktionen `ACCEPT`, `DROP` und `RETURN` besitzt Ebtables auch noch `CONTINUE` und Erweiterungen (Target Extensions). Wurde `CONTINUE` als Ziel (Target) einer Regel angegeben, so wird bei Zutreffen dieser Regel die Kette nicht verlassen, sondern die weiteren Regeln der Kette werden abgearbeitet. Als Erweiterungen stehen `arpreply`, `dnat`, `mark`, `redirect` und `snat` zur Verfügung. Achtung, diese Ziele werden entgegen der Konvention kleingeschrieben! Bei allen diesen Zielen müssen Sie zusätzlich noch angeben, was anschließend mit dem Paket passieren soll. Hierfür gibt es bei jedem Ziel noch eine zusätzliche Option, die weiter unten erläutert wird.

- `-j arpreply`: Das `arpreply`-Ziel erlaubt es, ARP-Anfragen zu beantworten. Dieses Ziel dürfen Sie nur in der `PREROUTING`-Kette der `nat`-Tabelle verwenden. Die Regel erkennt selbstständig, ob es sich um ARP-Anfragen handelt. Mit der Option `--arpreply-mac` können Sie die MAC-Adresse angeben, die in der Antwort verwendet werden soll. Mit der Option `--arpreply-target` geben Sie an, wie Ebtables anschließend den originalen Rahmen behandeln soll. Die Default-Einstellung ist DROP. Dies hat den Nachteil, dass der lokale ARP-Cache nicht mit den Absenderinformationen aktualisiert wird. Wenn Sie das Target auf `ACCEPT` oder `CONTINUE` setzen, ist dies dennoch der Fall. Sie müssen dann nur aufpassen, dass nicht ein weiterer ARP-Reply ausgesendet wird!
- `-j dnat`: Dieses Ziel darf in der `BROUTING`-Kette und in den `PREROUTING`- und `OUTPUT`-Ketten der `nat`-Tabelle genutzt werden. Die Ziel-MAC-Adresse geben Sie mit `--to-destination`, `--to-dst` an. Anschließend wird das Paket akzeptiert. Wenn Sie dies ändern möchten, können Sie mit `--dnat-target` ein anderes Ziel angeben.
- `-j mark`: Ebtables unterstützt wie Iptables eine Markierung der Rahmen/Pakete mit `mark`. Die Markierung geben Sie mit `--set-mark` als hexadezimale unsignierte Zahl an. Anschließend wird das Paket akzeptiert. Wenn Sie weitere Modifikationen des Pakets in der gleichen Kette vornehmen wollen, können Sie das mit `--mark-target CONTINUE` erreichen.
- `-j redirect`: Dieses Ziel ist analog dem `REDIRECT`-Ziel von Iptables und erlaubt es, Pakete in der `BROUTING`- und `PREROUTING`-Kette, die eigentlich an eine andere MAC-Adresse gerichtet sind und gebridget werden müssten, auf die lokale MAC-Adresse umzuleiten. Dabei wird als neue Ziel-MAC-Adresse die Adresse der Netzwerkkarte genutzt, über die

der Rahmen das System erreicht hat. Anschließend wird das Paket akzeptiert. Das Verhalten können Sie mit `--redirect-target` ändern. Mit dieser Option können Sie auf einer Bridge Verbindungen über einen transparenten Proxy lenken!

» `-j snat`: Hiermit können Sie in der `POSTROUTING`-Kette die Absender-MAC-Adresse ändern. Die neue MAC-Adresse geben Sie mit `--to-source`, `--to-src` an. Mit `--snat-target` können Sie ein anderes Ziel als `ACCEPT` (Default) angeben.

31.5 Start einer Bridge auf Linux

Sämtliche Linux-Distributionen bieten die einfache Möglichkeit, eine Bridge mithilfe Ihrer Startskripte zu definieren. Bei OpenSUSE ist diese Konfigurationsfunktionalität sogar in Yast enthalten. Auch unter Debian oder Fedora kann einfach eine Bridge realisiert werden.

Im Folgenden zeige ich, wie dies unter Fedora oder Debian funktioniert.

31.5.1 Bridge auf Fedora

Hierzu erzeugen Sie die folgenden Dateien in dem Verzeichnis `/etc/sysconfig/network-scripts`:

» `ifcfg-br0`

```
DEVICE=br0
ONBOOT=yes
BOOTPROTO=static
IPADDR=10.1.2.3
NETMASK=255.255.255.0
TYPE=Bridge
STP=off
DELAY=0
GCINT=30
```

» `ifcfg-eth0`

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
BRIDGE=br0
```

» `ifcfg-eth1`

```
DEVICE=eth1
ONBOOT=yes
BOOTPROTO=static
BRIDGE=br0
```

» Für jede weitere Netzwerkkarte der Bridge gibt es eine entsprechende Datei.

Fedora wird nun beim Booten die entsprechende Bridge initialisieren und die physikalischen Netzwerkkarten hinzufügen. Mit dem Parameter `STP` können Sie das STP-Protokoll an- beziehungsweise abschalten.

31.5.2 Bridge auf Debian

Um diese Bridge auf einem Debian-System automatisch bei dem Boot erzeugen zu lassen, können Sie die folgende Konfiguration in der Datei `/etc/networkinterfaces` verwenden:

```
iface eth0 inet manual
    up ifconfig eth0 up
    down ifconfig eth0 down

iface eth1 inet manual
    up ifconfig eth1 up
    down ifconfig eth1 down

iface br0 inet static
    address 10.1.2.3
    netmask 255.255.255.0
    bridge_ports eth0 eth1
    bridge_fd 0
    bridge_stp off
```

Teil VII

Protokolle und Applikationen

32. Behandlung einzelner Protokolle

In diesem Kapitel zeige ich Ihnen einzelne IP-Protokolle bei ihrer Filterung durch Netfilter und gebe Ihnen wertvolle Tipps für die sichere Konfiguration Ihrer Firewall. Dabei werden zunächst immer die Möglichkeiten aufgezeigt, die Sie in einer Standard-Linux-Distribution vorfinden. Wenn es darüber hinaus besondere Patches oder Erweiterungen gibt, werden diese extra behandelt und ihre Vorteile erläutert, sodass Sie selbst entscheiden können, ob Sie diese nutzen möchten.

Hierbei bauen die Beispiele auf IPv4 und `iptables` auf. Sie können aber auch direkt auf IPv6 und `ip6tables` angewendet werden. Die Besonderheiten im Bezug auf IPv6 werden in einem gesonderten Kapitel (siehe Kapitel 36) besprochen.

32.1 DHCP

Das Dynamic Host Configuration Protocol (DHCP) erlaubt die automatische IP-Konfiguration der Komponenten in einem Netzwerk. Es benötigt hierzu einen Server, der diese Konfigurationsparameter verwaltet und verteilt. Dabei löst der DHCP-Server das Problem der Zuordnung eindeutiger IP-Adressen, da er eine IP-Adresse nicht gleichzeitig an zwei Netzwerkkomponenten verteilt.

Grundsätzlich gibt es zwei verschiedene Möglichkeiten der Verteilung der IP-Adressen, die aber auch kombiniert eingesetzt werden können:

manuelle Zuordnung: Sie ordnen in der Konfiguration des DHCP-Servers jeder MAC-Adresse eine feste IP-Adresse zu. Sobald sich ein Client mit der MAC-Adresse bei dem DHCP-Server meldet, erhält er die entsprechende IP-Adresse.

dynamische Zuordnung: Sie weisen dem Server einen bestimmten Bereich von IP-Adressen zu, die er dynamisch verteilen darf. Sobald sich ein Client bei ihm meldet, weist er diesem Client für eine bestimmte Zeit eine IP-Adresse zu. Der Client darf diese Adresse für den genannten Zeitraum (Lease) nutzen. Anschließend kann der Server die Adresse an einen weiteren Client vergeben. Um die Adresse länger benutzen zu dürfen, muss der Client sich um eine Verlängerung bemühen.

Obwohl es zwei verschiedene Varianten der Zuordnung gibt, ist das Protokoll bei beiden Varianten identisch.

32.1.1 Das DHCP-Protokoll

Das DHCP-Protokoll basiert auf dem UDP-Protokoll. Der Server bindet sich auf den Port 67 (`bootps`), während der Client den Port 68 (`bootpc`) verwendet. Die gewählten IP-Adressen hängen stark von dem Zustand des Clients ab (siehe Abbildung 32.1).

Eine DHCP-Verbindung besteht aus folgenden Paketen:

1. Der Client sendet eine DHCP-Discover-Nachricht an sämtliche DHCP-Server in dem lokalen Netzwerk. Dieses UDP-Paket hat die IP-Absenderadresse 0.0.0.0:68 und die Zieladresse 255.255.255.255:67.
2. Alle verfügbaren DHCP-Server antworten mit einer DHCP-Offer-Nachricht, in der sie eine IP-Adresse anbieten.
3. Der Client wählt eines der Angebote aus und sendet einen DHCP-Request.
4. Dieser DHCP-Request wird von dem Server mit einem DHCP-Acknowledge bestätigt.
5. Jetzt darf der Client die IP-Adresse verwenden.
6. Bei einer Verlängerung (Renewal) der Nutzungsdauer (Lease) sendet der Client einen DHCP-Request als Unicast-Paket direkt an die Adresse des DHCP-Servers und trägt seine noch gültige IP-Adresse auch als Quell-IP-Adresse ein.

DHCPv6 verhält sich genauso wie das DHCP-Protokoll. Es verwendet jedoch andere Ports: 546/udp und 547/udp.

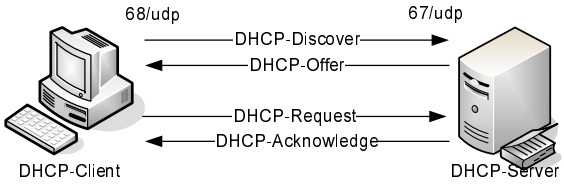


Abbildung 32.1: **Das DHCP-Protokoll**

32.1.2 Iptables-Regeln

Es ist eher unwahrscheinlich, dass Sie das DHCP-Protokoll auf einer normalen Firewall filtern möchten, da das Protokoll auf einer Broadcast-Kommunikation basiert und daher nicht über Router und damit auch nicht über Paketfilter arbeiten kann. Jedoch könnte es sein, dass Sie für die Kommunikation über den Paketfilter ein DHCP-Relay einsetzen, das genau diese Aufgabe übernimmt. Es nimmt die Broadcast-Anfrage entgegen und leitet sie per Unicast-Paket an den DHCP-Server in einem anderen Netz weiter. Dann benötigen Sie folgende Regeln:

TS Bitte bestätigen.

Editor's or typesetter's annotations (will be removed before the final TEX run)

Listing 32.1: **Regeln für die Kommunikation eines DHCP-Relays mit einem DHCP-Server**

```
DHCP_SERVER=192.168.0.2
DHCP_RELAY=192.168.1.2

$IPTABLES -A FORWARD -s $DHCP_RELAY -d $DHCP_SERVER -p udp --sport bootpc ↵
          --dport bootps -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Wenn Sie einen DHCP-Server mit einer lokalen Firewall schützen möchten, benötigen Sie die folgenden Regeln:

```
$IPTABLES -A INPUT -p udp --sport bootpc --dport bootps -m state --state ↵
        NEW -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Es gibt jedoch auch einen Fall, in dem es Sinn machen kann, das DHCP-Protokoll tatsächlich über eine Firewall zu erlauben. Dies ist der Fall, wenn Sie eine transparente Firewall im Bridge-Mode aufbauen. Dann arbeitet die Firewall nicht als Router, sondern wie eine Bridge und sollte DHCP-Pakete erlauben.

```
DHCP_SERVER=192.168.0.2
CLIENTS=192.168.0.0/24

$IPTABLES -A FORWARD -s $CLIENTS -d $DHCP_SERVER -p udp --sport bootpc -- ↵
        dport bootps -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -s 0.0.0.0 -d 255.255.255.255 -p udp --sport bootpc ↵
        --dport bootps -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.2 DNS

Das Domain Name System (DNS) ist einer der wichtigsten Dienste im Internet. Es handelt sich dabei um eine große hochverfügbare und verteilte Datenbank, die den Namensraum im Internet verwaltet. Seine wesentliche Aufgabe ist die Auflösung von Namen in IP-Adressen und umgekehrt. Zusätzlich ist es aber auch für das Mail-Routing verantwortlich und wird immer mehr für den Schlüsselaustausch von PKI-Systemen genutzt.

32.2.1 Das DNS-Protokoll

Das DNS-Protokoll unterscheidet sich von den meisten anderen Protokollen dadurch, dass es sowohl UDP als auch TCP als Transportprotokoll nutzen kann. Die Auswahl des Transportprotokolls erfolgt dabei jedoch nicht beliebig. Der Client stellt zunächst die Anfrage mit dem UDP-Protokoll. Dabei verwendet der Server den Port 53/udp und der Client einen beliebigen UDP-Port > 1023. Der Server beantwortet die Anfrage. Wenn jedoch die Antwort ein Paket

> 512 Bytes benötigt, dann wird das Paket bei 512 Bytes abgeschnitten und das TC-Flag im DNS-Header gesetzt. Dieses Flag bedeutet *truncated* (abgeschnitten). Der Client muss dann erneut die Anfrage stellen. Diese neue Anfrage wird mit dem TCP-Protokoll gestellt.

Das RFC 2671 hat zumindest die Voraussetzungen geschaffen, dass diese letzte Einschränkung fällt. Mit dem RFC Extension Mechanisms for DNS (EDNS0) ist es möglich, dass der Client die Größe seines Empfangspuffers angibt und so auch größere Pakete als 512 Bytes gesendet werden dürfen.

STOP

*Ein Betriebssystem, das jetzt bereits diese Funktion nutzt, ist Microsoft Windows 2003 Server*CE^s*. Auch der Bind-DNS-Server nutzt dies. Dies kann zu Problemen führen, wenn Firewalls DNS-UDP-Pakete > 512 Bytes verwerfen (z. B. Cisco PIX < 6.3). Diese Funktionalität kann bei Win2k3 mit dem Kommando* `dnscmd` *abgeschaltet werden:*

```
dnscmd /config /enableednsprobes 0
```

Zur Synchronisation der sekundären Nameserver mit den primären Nameservern werden Zonentransfers durchgeführt. Diese Zonentransfers werden grundsätzlich mit dem TCP-Protokoll durchgeführt.

Das DNS-Protokoll ist ein Klartextprotokoll. Eine Authentifizierung und Signatur ist bei Bedarf möglich (TSIG, DNSSEC). Zunehmend werden diese Funktionen auch genutzt. Eine flächendeckende Signatur der DNS-Informationen im Internet existiert jedoch noch nicht. Das Denic führt aktuell (März 2011) noch Tests für die DE-Zone durch.

32.2.2 Iptables-Regeln

Da häufig nur bestimmte DNS-Server für die Namensauflösung eingesetzt werden, kann es sinnvoll sein, die erlaubten DNS-Anfragen speziell auf diese Nameserver zu begrenzen.

STOP

Selbst wenn Sie die Nutzung auf spezielle Nameserver beschränken, besteht die Gefahr, dass das DNS-Protokoll zur Übertragung anderer Informationen genutzt wird.

Seit einigen Jahren existieren Anwendungen, die beliebige Protokolle über DNS tunneln können. Das bekannteste Programm für einen DNS-Tunnel ist sicher `nstx` *(*`http://thomer.com/howtos/nstx.html`*). Ein weiteres mächtigeres Programm wurde von Dan Kaminsky auf der Konferenz Blackhat 2005 in Amsterdam vorgestellt. Sein Fragile-Router-Protokoll ist in der Lage, 65 Kbit/s Daten über das DNS-Protokoll zu streamen. Dieses Protokoll wurde jedoch bis heute nicht veröffentlicht.*TS^t

NSTX wird nicht mehr weiterentwickelt. Ein würdiger Nachfolger ist Iodine (`http://code.kryo.se/iodine/`*).*

Da DNS-Server Anfragen, die sie nicht beantworten können, weiterleiten, ist die Einschränkung der Nutzung auf bestimmte Nameserver nicht ausreichend. Hier könnten Sie versuchen, die Anzahl der Anfragen je Client zu beschränken.

CE^s Gibt es kein aktuelleres Betriebssystem? Klingt so ein wenig veraltet.

TS^t Noch aktuell?

Im Folgenden stelle ich die Regeln für drei Szenarien vor.

1. Ein Netz (LAN) greift auf zwei DNS-Server zu. Dann benötigen Sie die folgenden Regeln:

```
LANDEV=eth0
EXTDEV=eth1
DNSSERVER="3.0.0.1 5.0.0.1"
for DNSSRV in $DNSSERVER
do
  $IPTABLES -A FORWARD -i $LANDEV -o $EXTDEV -p udp --dport 53 -d ↵
          $DNSSRV -m state --state NEW -j ACCEPT
  $IPTABLES -A FORWARD -i $LANDEV -o $EXTDEV -p tcp --dport 53 -d ↵
          $DNSSRV -m state --state NEW -j ACCEPT
done
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

2. Ein DNS-Server greift auf weitere DNS-Server zu:

```
LANDEV=eth0
EXTDEV=eth1
MYDNS=192.168.255.2
DNSSERVER="3.0.0.1 5.0.0.1"
for DNSSRV in $DNSSERVER
do
  $IPTABLES -A FORWARD -i $LANDEV -o $EXTDEV -p udp --dport 53 -s ↵
          $MYDNS -d $DNSSRV -m state --state NEW -j ACCEPT
  $IPTABLES -A FORWARD -i $LANDEV -o $EXTDEV -p tcp --dport 53 -s ↵
          $MYDNS -d $DNSSRV -m state --state NEW -j ACCEPT
done
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

3. Sie möchten einen DNS-Server mit einer lokalen Firewall schützen. Dieser muss sowohl auf weitere DNS-Server für die rekursive Auflösung zugreifen als auch den Zugriff der lokalen Clients erlauben. Die Regeln nutzen die Tatsache, dass der Nameserver Bind üblicherweise mit einem eingeschränkten Benutzer betrieben wird. Diese Regeln gehen von dem Benutzer *named* aus.

```
# Der DNS-Server muss auf andere DNS-Server zugreifen
DNSSERVER="3.0.0.1 5.0.0.1"
CLIENTS=192.168.0.0/24

for DNSSRV in $DNSSERVER
do
  $IPTABLES -A OUTPUT -p udp --dport 53 -d $DNSSRV -m state --state ↵
          NEW -m owner --uid-owner named --cmd-owner named -j ACCEPT
  $IPTABLES -A OUTPUT -p tcp --dport 53 -d $DNSSRV -m state --state ↵
          NEW -m owner --uid-owner named --cmd-owner named -j ACCEPT
done
```

```
# Die Clients müssen auf ihn zugreifen
$IPTABLES -A INPUT -s $CLIENTS -p udp --dport 53 -m state --state NEW ↵
        -j ACCEPT
$IPTABLES -A OUTPUT -s $CLIENTS -p tcp --dport 53 -m state --state NEW ↵
         -j ACCEPT

$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.3 HTTP/HTTPS/Proxy

Das HTTP-Protokoll ist mit Abstand das am häufigsten verwendete Protokoll im Internet. Es dient zum Zugriff auf Webserver und verwendet üblicherweise auf der Seite des Servers den TCP-Port 80. Clientseitig wird ein beliebiger TCP-Port > 1024 verwendet.

Das HTTPS-Protokoll ist mit dem HTTP-Protokoll identisch. Es verwendet nur zusätzlich Secure Socket Layer (SSL). Dies erlaubt eine Authentifizierung des Servers und des Clients mit einer anschließenden Verschlüsselung der gesamten Verbindung. Für die Authentifizierung benötigt das SSL-Protokoll öffentliche Schlüssel in Form von X.509-Zertifikaten. Der zu authentifizierende Rechner muss zusätzlich über den passenden privaten Schlüssel verfügen. Das HTTPS-Protokoll verwendet einen eigenen Port (443/tcp) und verlangt auf diesem Port den Einsatz von SSL.

Bei dem Zugriff auf einen Proxy wird ebenfalls das HTTP-Protokoll verwendet. Auch hier nutzt der Client nur einen einzigen Port für die Kommunikation mit dem Server. Dieser TCP-Port ist häufig frei wählbar. Jedoch hat auch jeder Proxy einen typischen Standardport (Squid: 3128/tcp). Wenn der Proxy mehrere Protokolle (z. B. HTTP, HTTPS und FTP) erlaubt, erfolgt der Zugriff des Clients immer mit HTTP.

Das HTTP-Protokoll ist ein zustandsloses Protokoll. Normalerweise baut der Client für jede Information, die er wünscht, eine eigene Verbindung auf und fordert die Daten an. Der Server überträgt die Daten und baut die Verbindung ab. Für den Server handelt es sich bei jeder Verbindung um einen neuen Client. Damit eine Webapplikation auf dem Server erkennen kann, dass es sich bei allen Aufrufen um denselben Benutzer handelt, wurden Cookies erfunden. Es gibt clientseitige Cookies, die von dem Server an den Browser übertragen werden und bei dem nächsten Besuch von dem Browser wieder an den Server geschickt werden. Im Gegensatz dazu werden serverseitige Cookies meist in der URL als Zeichenkette kodiert. Der Server stattet alle Verknüpfungen auf der Webseite mit derselben Zeichenkette aus und kann so den Weg des Benutzers durch die Seiten verfolgen.

Das HTTP-Protokoll erlaubt auch eine Authentifizierung des Benutzers mit Benutzernamen und Kennwort. Dabei existieren insgesamt drei Möglichkeiten der Übertragung dieser Informationen:

» BASIC: Hier werden der Benutzername und das Kennwort, durch einen Doppelpunkt getrennt, in Base64-Kodierung übertragen. Es handelt sich also um eine Übertragung in

Klartext. Mit der Anwendung Wireshark (früher Ethereal) können Sie sich diese Zeichenkette ansehen (siehe Abbildung 32.2).

Die Zeichenkette lässt sich dann auf jedem Linux-System leicht dekodieren:

```
[spenneb@bibo screenshots]$ echo dXNlcjpnZWhlaW0= | openssl base64 -d
user:geheim
```

» DIGEST: Hier wird lediglich der Benutzername übertragen. Anstelle des Kennworts wird das Ergebnis einer mathematischen Berechnung übertragen. Dazu übermittelt der Server eine Zufallszahl an den Client, der aus dieser Zahl und dem Kennwort eine Antwort errechnet und diese zurückschickt. Der Server führt dieselbe Berechnung durch und vergleicht das Ergebnis.

» NTLM/GSSAPI: Hier erfolgt die Anmeldung analog einer Anmeldung an einem Microsoft Windows-Server oder an einer Kerberos-Domäne. Die Anmeldeinformationen werden hier verschlüsselt übertragen.

Bei der Verwendung der Authentifizierung direkt durch das HTTP-Protokoll ist zu beachten, dass diese Informationen bei jedem Verbindungsaufbau erneut übertragen werden, da der Server nicht in der Lage ist, zu erkennen, dass die Verbindungen immer von demselben Benutzer aufgebaut werden! Daher ist es sinnvoll, entweder die gesamten Verbindungen mit Secure Socket Layer (SSL) zu schützen oder auf alternative Authentifizierungsverfahren umzustellen. Hierzu werden häufig Webapplikationen genutzt, die dann die Authentifizierung durchführen. Diese Phase wird mit SSL geschützt. Anschließend erzeugt die Webapplikation eine Zufallszahl, die in einem Cookie gespeichert wird, der anschließend für die Authentifi-

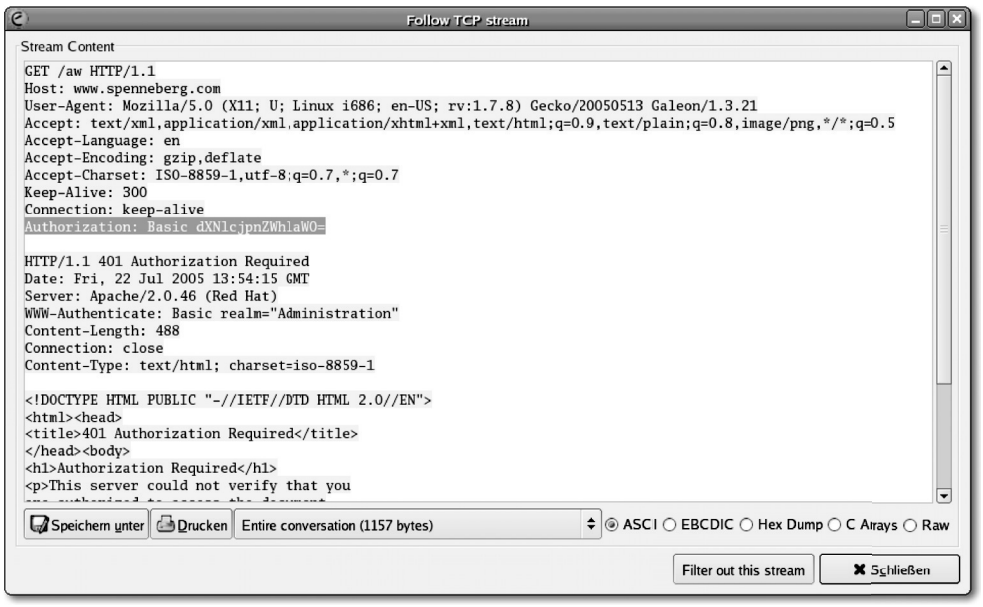


Abbildung 32.2: **Wireshark kann die BASIC-HTTP-Authentifizierung anzeigen.**

zierung genutzt wird. Die weiteren Verbindungen können dann wieder in Klartext übertragen werden, da kein Kennwort mehr übertragen wird. Es besteht nun nur noch die Gefahr, dass der Cookie geklaut wird. Daher wird häufig dieser Cookie mit einer maximalen Lebensdauer von wenigen Minuten versehen. Dennoch ist ein Cookie-Diebstahl nicht zu unterschätzen und bei vielen Applikationen einem Kennwort-Diebstahl gleichzustellen.

INFO

Es gibt Web-Applikationen, die sich für einen Cookie nicht nur den authentifizierten Benutzer, sondern auch dessen IP-Adresse und Browserversion merken. Wird der Cookie dann von einer anderen IP-Adresse oder mit einem anderen Browser benutzt, kann die Applikation den Cookie-Diebstahl erkennen und die Authentifizierung ablehnen. Dies muss jedoch von dem Entwickler der Webapplikation vorgesehen werden.

32.3.1 Iptables-Regeln

Die Iptables-Regeln für dieses Protokoll sind sehr einfach, da es lediglich zwei Ports (HTTP: 80/tcp und HTTPS: 443/tcp) verwendet. Wenn Sie keine HTTPS-Verbindungen erlauben wollen, entfernen Sie den Port 443 aus den Regeln. Für einen Proxy fügen Sie den entsprechenden Port hinzu.

Zunächst werden Beispielregeln demonstriert, sodass ein Netz (LAN) auf beliebige Webserver im Internet zugreifen darf.

Listing 32.2: **Zugriff auf Webserver im Internet**

```
LANDEV=eth0
EXTDEV=eth1

$IPTABLES -A FORWARD -i LANDEV -o EXTDEV -p tcp -m multiport --dport ↵
        80,443 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Natürlich können Sie auch einen Webserver mit einer lokalen Firewall sichern. Dann können Sie folgende Regeln verwenden:

Listing 32.3: **Eine lokale Firewall schützt den Webserver.**

```
$IPTABLES -A INPUT -p tcp -m multiport --dport 80,443 -m state --state ↵
        NEW -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.4 ELSTER

Die elektronische Steuererklärung (ELSTER) überträgt die Steuerdaten über das Internet an die Finanzämter in Deutschland. Hierbei werden die Daten verschlüsselt übertragen. Über ELSTER können die folgenden Informationen übermittelt werden:

» Einkommensteuererklärung
» Umsatzsteuer-Voranmeldung
» Antrag auf Dauerfristverlängerung
» Antrag auf Sondervorzahlung
» Umsatzsteuererklärung
» Gewerbesteuererklärung
» Lohnsteuererklärung
» Lohnsteuerbescheinigungen

Leider gibt es inzwischen verschiedene Möglichkeiten der Übertragung, die von unterschiedlichen Systemen unterschiedlich genutzt werden. Elster selbst ist eine Software, die nicht einzeln erhältlich ist, sondern in andere Programme, zum Beispiel Buchhaltungs- oder Lohn- und Gehaltssoftware eingebaut wird. Des Weiteren gibt es mit der Software ElsterFormular eine kostenlose Software, die zum Ausfüllen der Formulare und deren Übertragung genutzt werden kann. ElsterFT ist schließlich eine Software für Behörden, um verfahrensabhängige Formulare herunterzuladen, Sende- und Abholaufträge zu erstellen und Monitor- und Protokollfunktionen zu übernehmen. Eine Übertragung von Steuererklärungen oder deren Daten ist nicht möglich.

Als Verfahren können sowohl ELSTER-1 als auch ELSTER-2 verwendet werden.

» ELSTER 1 (Datenübertragung ohne Authentifizierung): 62.157.211.58, 62.157.211.59, 193.109.238.26, 193.109.238.27, 213.182.157.55 (Kommunikation über die Ports 8000 und 443)
» ELSTER 2 (Datenübertragung mit Authentifizierung): 80.146.179.2, 80.146.179.3, 193.109.238.58, 193.109.238.59 (Kommunikation über Port 80)

32.4.1 Iptables-Regeln

Im Wesentlichen benötigt Elster Zugriff auf die Server der Finanzverwaltung. Hier muss der Zugriff in Abhängigkeit vom jeweiligen Verfahren auf den TCP-Port 80, 443 und 8000 möglich sein. Folgende Regeln erlauben den Zugriff:

```
ELSTER1_SERVER="62.157.211.58 62.157.211.59  193.109.238.26  ↵
        193.109.238.27 213.182.157.55"
ELSTER2_SERVER="80.146.179.2, 80.146.179.3, 193.109.238.58,  ↵
        193.109.238.59"
for SRV in $ELSTER1_SERVER
do
  $IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -d $SRV -p tcp -m multiport  ↵
          --dport 443,8000 -m state --state NEW -j ACCEPT
done
for SRV in $ELSTER2_SERVER
do
```

```
  $IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -d $SRV -p tcp -m multiport ↵
          --dport 80 -m state --state NEW -j ACCEPT
done
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Wenn Ihr Kernel und Ihr Iptables-Befehl den Befehl `ipset` unterstützen, können Sie das auch einfacher konfigurieren:

```
IPSET=/sbin/ipset
$IPSET -N elster1 iphash
$IPSET -A elster1 62.157.211.58
$IPSET -A elster1 62.157.211.59
$IPSET -A elster1 193.109.238.26
$IPSET -A elster1 193.109.238.27
$IPSET -A elster1 213.182.157.55
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -m set --set elster1 dst -p ↵
        tcp -m multiport --dport 443,8000 -m state --state NEW -j ACCEPT
```

Diese Regeln benutzen nur eine einzige Iptables-Regel. Damit ist das Regelwerk wesentlich übersichtlicher als mit sechs Regeln.

32.5 Telnet

Das Telnet-Protokoll simuliert eine Terminal-Umgebung über ein Netzwerk (Terminal Emulation over NETwork). Es überträgt sämtliche Daten in Klartext. Dies trifft auch auf den Benutzernamen und das Kennwort zu. Daher wird eine Anmeldung als root üblicherweise auch abgelehnt. Achtung: Das Kennwort wird trotzdem übertragen. Es gibt moderne sichere Alternativen, wie Telnet über SSL, kerberosiertes Telnet und die Secure Shell (SSH). Der Telnet-Server nimmt auf dem TCP-Port 23 Verbindungen entgegen. Der Client wählt einen beliebigen Port > 1023.

32.5.1 Iptables-Regeln

Obwohl ich grundsätzlich gegen den Einsatz von Telnet in modernen Netzwerken plädiere, da es wesentliche Sicherheitslücken aufweist, wird der Einsatz dennoch häufig gefordert. Häufig werden noch ältere Netzwerkkomponenten wie Router, Switches oder Drucker per `telnet` konfiguriert. Eine Aufrüstung auf SSH ist teilweise zwar möglich, kostet aber häufig zusätzliche Lizenzgebühren.

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp --dport 23 -m state -- ↵
        state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Wenn Sie einen Telnet-Server mit einer lokalen Firewall schützen möchten, verwenden Sie die folgenden Regeln:

```
$IPTABLES -A INPUT -p tcp --dport 23 -m state --state NEW -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.6 SSH

Die Secure Shell (SSH) stellt eine sichere Alternative zu den folgenden Protokollen und Anwendungen dar: `telnet`, `rlogin`, `rcp`, `ftp` etc. Dabei überprüft die Secure Shell im ersten Schritt die Identität des Servers, zu dem die Verbindung aufgebaut wird, bevor das Kennwort über einen verschlüsselten Tunnel übertragen wird. Achtung: Das Kennwort erreicht den Serverprozess in Klartext. Lediglich die Übertragung ist verschlüsselt. Ein bösartiger Server könnte das Kennwort in Klartext abspeichern. Um auch hiervor Schutz zu bieten, bietet die SSH auch die Authentifizierung des Clients mit öffentlichen Schlüsseln an.[1] Sämtliche anschließend übertragenen Daten werden dann von der SSH verschlüsselt, und jedes einzelne Paket wird auf seine Integrität und Authentizität geprüft.

Die meisten Implementierungen der Secure Shell bieten inzwischen nicht nur den `ssh`-Befehl, sondern auch die Befehle `scp` und `sftp`. Mit diesen Befehlen können Dateien kopiert oder FTP-ähnlich transportiert werden. Unabhängig von dem gewählten Namen werden die Daten rein über die SSH-Verbindung transportiert. Ein zusätzlicher FTP-Server ist nicht erforderlich. Ab der Version 4.0 kann OpenSSH auch ganze VPN-Verbindungen mit virtuellen Netzwerkkarten aufbauen.

Der SSH-Server bindet sich auf den TCP-Port 22 und nimmt dort Verbindungen entgegen.

32.6.1 Iptables-Regeln

Die Regeln für die Filterung von SSH sind sehr einfach und mit den Regeln für einen HTTP-Server vergleichbar.

Listing 32.4: **Zugriff auf SSH-Server im Internet**

```
LANDEV=eth0
EXTDEV=eth1

$IPTABLES -A FORWARD -i LANDEV -o EXTDEV -p tcp --dport 22 -m state -- ↵
        state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Natürlich können Sie auch einen SSH-Server mit einer lokalen Firewall sichern. Dann können Sie folgende Regeln verwenden:

1 Um die öffentlichen Schlüssel einfach auf die Zielsysteme zu verteilen, bietet die SSH den Befehl `ssh-copy-id`.

Listing 32.5: **Eine lokale Firewall schützt den SSH-Server.**

```
$IPTABLES -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.7 P2P: Edonkey

Bei den Peer2Peer-Netzwerken gibt es grundsätzlich zwei Ansätze eines Firewall-Administrators: Der Heimanwender möchte diese Dienste häufig nutzen, um einfach auf bestimmte Dateien zugreifen zu können, während der Administrator in einer Firma möglichst jede Anwendung verhindern möchte, da die Gefahr besteht, dass mit diesen Protokollen Copyright-Verletzungen erfolgen, die negative Folgen für die Firma haben können. Wir werden daher beide Varianten darstellen. Wie können Sie möglichst einfach P2P-Verkehr erkennen und verhindern, und wie ermöglichen Sie diesen Verkehr?

Edonkey nutzt wie fast alle anderen P2P-Netze hohe Ports (> 1023) für die Kommunikation des Clients mit dem Server. Bei Edonkey handelt es sich um die folgenden Ports:

» 4661/tcp: Der Client verbindet sich mit diesem Port auf dem Server und meldet sich dort an.

» 4662/tcp: Der Server verbindet sich mit diesem Port auf dem Client. Die Erreichbarkeit dieses Ports durch den Server auf dem Client entscheidet auch darüber, ob der Client eine hohe (>= 16777217) oder eine niedrige ID (<= 16777216) erhält. Dieser Port kann auf dem Client geändert werden. Die ID ist ausschlaggebend für die eigene Download-Rate auf fremden Servern.

» 4665/udp: Dieser Port wird benötigt, um Quellen auf Servern zu finden, auf denen man nicht angemeldet ist.

» 4672/udp: Dieser Port wird für direkte Client-Client-Verbindungen genutzt. Auch dieser Port ist einstellbar.

32.7.1 Iptables-Regeln

Um nun mit einem Client eine Verbindung zum Edonkey-Netzwerk aufzubauen, sollte eine Verbindung zu den entsprechenden Ports erlaubt werden:

```
# Client-Zugriff auf den Server
EDONKEY_CLIENT=192.168.0.5
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -s $EDONKEY_CLIENT -p tcp -m ↵
        multiport --dport 4661,4662 -m state --state -NEW -j ACCEPT
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -s $EDONKEY_CLIENT -p udp -m ↵
        multiport --dport 4665,4672 -m state --state -NEW -j ACCEPT
```

```
# Zugriff auf den Client
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp --dport 4662 -j DNAT -- ↵
        to-destination $EDONKEY_CLIENT
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p udp --dport 4672 -j DNAT -- ↵
        to-destination $EDONKEY_CLIENT
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -d $EDONKEY_CLIENT -p tcp -- ↵
        dport 4662 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -d $EDONKEY_CLIENT -p udp -- ↵
        dport 4672 -m state --state NEW -j ACCEPT

$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Falls Sie den Edonkey-Client mit einer lokalen Firewall direkt schützen möchten, benötigen Sie folgende Regeln:

```
$IPTABLES -A OUTPUT -p tcp -m multiport --dport 4661,4662 -m state -- ↵
        state -NEW -j ACCEPT
$IPTABLES -A OUTPUT -p udp -m multiport --dport 4665,4672 -m state -- ↵
        state -NEW -j ACCEPT
$IPTABLES -A INPUT -p tcp --dport 4662 -m state --state NEW -j ACCEPT
$IPTABLES -A INPUT -p udp --dport 4672 -m state --state NEW -j ACCEPT

$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Häufiger jedoch möchten Sie den Edonkey-Verkehr unterbinden. Am einfachsten ist das, wenn Ihre Firewall grundsätzlich alles verbietet und nur bestimmte Protokolle erlaubt. Edonkey ist dann einfach nicht dabei.

Vielleicht müssen Sie aber die Firewall für eine Universität implementieren. Hier wird häufig bewusst ein anderer Ansatz gewählt. Um die freie Forschung nicht zu behindern, wird alles erlaubt, und es werden nur bestimmte Protokolle unterbunden.

Um Edonkey zu verhindern, können Sie den Zugriff auf die Ports unterbinden:

```
$IPTABLES -A FORWARD  -p tcp -m multiport --dport 4661,4662 -j REJECT
$IPTABLES -A FORWARD  -p udp -m multiport --dport 4665,4672 -j REJECT
```

Es gibt jedoch auch noch zwei weitere mögliche Lösungen: L7-Filter (`http://l7-filter.sourceforge.net/`) und das Iptables-Modul Iptables-p2p (`http://sourceforge.net/projects/iptables-p2p/`). Leider wird das Letztere seit September 2004 nicht mehr aktiv weiterentwickelt.

L7-Filter wird in einem eigenen Kapitel besprochen (siehe Kapitel 33).

32.8 P2P: KaZaA

KaZaA ist ebenfalls wie Edonkey ein P2P-Filesharing-Protokoll. Auch hier gibt es grundsätzlich zwei Zielsetzungen: erlauben oder verbieten. Während es bei Edonkey relativ leicht war, den Verkehr zu verbieten, ist dies bei KaZaA wesentlich schwieriger. KaZaA benutzt den Standardport 1214/tcp. Dieser Port kann jedoch auch frei gewählt werden. Für eine volle Funktionalität muss dieser Port auf dem Client auch von außen erreichbar sein. Verhindert die Firewall den Zugriff auf diesen Port, kann KaZaA auch den Port 80 verwenden. Dies ist sogar bei den aktuellen KaZaA-Clients die Default-Einstellung. Eine Unterdrückung dieses Ports wird jedoch in den meisten Netzwerken nicht möglich sein. KaZaA kann sogar über einen Proxy arbeiten!

32.8.1 Iptables-Regeln

Um die volle KaZaA-Funktionalität mit dem Standardport zu erhalten, können Sie die folgenden Regeln verwenden:

```
KAZAA_PORT=1214
KAZAA_CLIENT=192.168.0.5
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -s $KAZAA_CLIENT -p tcp -- ↵
        dport $KAZAA_PORT -m state --state NEW -j ACCEPT
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp --dport $KAZAA_PORT -j  ↵
        DNAT --to-destination $KAZAA_CLIENT
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -d $KAZAA_CLIENT -p tcp -- ↵
        dport $KAZAA_PORT -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Wenn Sie den KaZaA-Verkehr unterbinden möchten, benötigen Sie eine intelligente Protokollerkennung. Hier können Sie dieselben Systeme einsetzen wie bei Edonkey. Zusätzlich gibt es für KaZaA, das das Fasttrack-Protokoll verwendet, noch die FTwall von Chris Lowth (`http://www.lowth.com/p2pwall/ftwall/`). Diese wird jedoch seid 2004 nicht weiterentwickelt. Außerdem können Sie versuchen, mit dem Iptables-`string`-Modul spezielle KaZaA-Pakete zu erkennen und die Verbindungen abzubrechen:

```
$IPTABLES -A FORWARD -p tcp -m string --string "X-Kazaa-Username:" -j  ↵
        REJECT
$IPTABLES -A FORWARD -p tcp -m string --string "X-Kazaa-Network:" -j  ↵
        REJECT
$IPTABLES -A FORWARD -p tcp -m string --string "X-Kazaa-IP:" -j REJECT
$IPTABLES -A FORWARD -p tcp -m string --string "X-Kazaa-SupernodeIP:" -j  ↵
        REJECT
```

Aktuell nutzen Sie am besten den L7-Filter (siehe Kapitel 33).

32.9 P2P: BitTorrent

BitTorrent beschreibt sich selbst als ein Werkzeug zur freien Meinungsäußerung. Es handelt sich auch um ein Filesharing-System, das aber nicht mit dem Ziel geschaffen wurde, urheberrechtlich geschützte Dateien zu tauschen. BitTorrent wurde geschaffen, um Firmen und Personen, die interessante Dateien anzubieten haben, eine optimale Download-Plattform zu bieten. Wenn eine Firma eine neue Demodatei eines Spiels anbieten möchte, wird die Download-Rate rapide in die Höhe schnellen. Die Anwender müssen sich mit langsameren Downloads zufriedengeben, und die Firma muss die hohen Kosten für die Übertragung tragen. BitTorrent löst das Problem, indem die Anwender, die die Datei gerade herunterladen, wieder anderen Anwendern als Download-Plattform zur Verfügung stehen. Dadurch wird die maximale Download-Geschwindigkeit vervielfacht. Dieser verteilte gemeinsame Download ist das Geheimnis des Erfolges von BitTorrent. Diese Plattform wird von immer mehr Firmen (zum Beispiel auch Red Hat) genutzt, um große Datenmengen (zum Beispiel CD-ISO-Abbilder) schnell und preiswert zur Verfügung zu stellen.

Das BitTorrent-Protokoll verwendet die TCP-Ports 6969 und 6881–6889. Um die volle Funktionalität und hohe Download-Geschwindigkeiten zu erreichen, muss der Client auf diesen Ports auch von außen erreichbar sein. Dieses Modell kennen Sie schon von anderen P2P-Protokollen.

Heute werden über BitTorrent aber auch zu einem großen Anteil urheberrechtlich geschützte Dateien ausgetauscht.

32.9.1 Iptables-Regeln

Wenn Sie den BitTorrent-Verkehr durch Ihre Firewall für einen Client erlauben möchten, können Sie die folgenden Regeln verwenden:

```
# Client-Zugriff auf den Server
BITT_CLIENT=192.168.0.5
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -s $BITT_CLIENT -p tcp --dport ↵
         6969 -m state --state -NEW -j ACCEPT
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -s $EDONKEY_CLIENT -p tcp -- ↵
        dport 6881:6889 -m state --state -NEW -j ACCEPT

# Zugriff auf den Client
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp --dport 6881:6889 -j  ↵
        DNAT --to-destination $BITT_CLIENT
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tdp --dport 6969 -j DNAT -- ↵
        to-destination $BITT_CLIENT
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -d $BITT_CLIENT -p tcp --dport ↵
         6881:6889 -m state --state NEW -j ACCEPT
```

```
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -d $BITT_CLIENT -p tcp --dport ↵
         6969 -m state --state NEW -j ACCEPT

$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Wenn Sie den Zugriff verhindern möchten, ist es am sinnvollsten, grundsätzlich nur bestimmte Protokolle zu erlauben und BitTorrent nicht mit aufzunehmen. Sollten Sie sich in der Lage befinden, dass Sie grundsätzlich alles erlauben, aber nur BitTorrent unterbinden möchten, so können Sie die Ports 6881–6889 in Ihrer Firewall schließen. Jedoch kann dieser Portbereich beliebig eingestellt werden. Sinnvoller ist auch hier der Einsatz eines intelligenten Protokollerkennungswerkzeugs. Dies kann das L7-Filter-Modul sein. Dieses wird in Kapitel 33 besprochen.

32.10 FTP

Das File Transfer Protocol ist eines der ältesten Applikationsprotokolle, die heute noch im Einsatz sind. Es wurde 1973 entwickelt und als RFC veröffentlicht. Nur zum Vergleich: Ethernet wurde 1974 und UUCP wurde 1978 entwickelt. Das Internetprotokoll IP hat erst 1980 das Network Control Protocol (NCP) abgelöst.

Dennoch ist FTP eines der kompliziertesten Protokolle. Es verlangt zu Beginn eine Authentifizierung und überträgt sämtliche Informationen im Klartext. Auch der Benutzername und das Kennwort werden ungeschützt übertragen.

Es existieren kerberosierte Versionen, die einen Single-Sign-On ermöglichen und auch anschließend sämtliche Daten bei der Übertragung schützen können. Außerdem bietet die SSH häufig einen SFTP-Zugang an. Dies ist ein geschützter FTP-ähnlicher Zugang zum Dateitransfer.

32.10.1 Das FTP-Protokoll

Das FTP-Protokoll ist deswegen so kompliziert, weil die Anmeldeinformationen und die Befehle von dem FTP-Client zum Server über die Steuerungsverbindung übertragen werden, während alle Daten, die der Server an den Client übermittelt, über eigene separate Datenverbindungen transportiert werden.

Die Steuerungsverbindung (Control Connection) wird von dem Client zum Server auf dem Port 21/tcp aufgebaut. Der Client verwendet für den Aufbau einen beliebigen Port > 1023. Diese Verbindungen werden genutzt, um alle Anmeldeinformationen zu übertragen und die Befehle des Clients an den Server zu übertragen. Dateien werden über eigene Datenverbindungen übertragen. Dies trifft auch schon auf die Inhaltsangabe eines Verzeichnisses bei dem Befehl `ls` bzw. `dir` zu. Ebenso ist dies unabhängig von der Richtung des Transfers. Sowohl der `PUT`- als auch der `GET`-Befehl triggert eine Datenverbindung.

Für den Aufbau dieser Datenverbindungen gibt es grundsätzlich zwei Möglichkeiten. Bei dem aktiven FTP baut der Server diese Datenverbindungen auf. Dazu übermittelt der Client dem

Server in einem PORT-Kommando die IP-Adresse und den Port, auf dem er die Verbindung des Servers entgegennimmt. Der Server baut dann von dem Port 20/tcp (ftp-data) die Verbindung zu dem Port auf, den er von dem Client erhalten hat (siehe Abbildung 32.3). Für jede zu transportierende Datei wird eine neue Datenverbindung geöffnet. Die Verbindung wird nach dem Transport nicht für zukünftige Dateien offen gehalten.

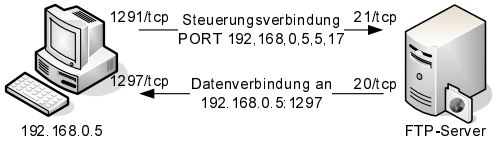


Abbildung 32.3: **Der Server baut bei dem aktiven FTP die Datenverbindung auf.**

Bei dem passiven FTP sendet der Client ein PASV-Kommando über die Steuerungsverbindung an den Server. Der Server antwortet mit einer IP-Adresse und einer Portnummer, auf der er den Aufbau der Datenverbindung erwartet. Hier verwendet der Server einen TCP-Port >1023!

Der FTP-Server kann als Antwort auf ein PASV-Kommando eine andere IP-Adresse zurückliefern, als er für die Steuerungsverbindung verwendet!

Der Client baut nun die Verbindung von einem beliebigen TCP-Port >1023 zum angegebenen Port des FTP-Servers auf (siehe Abbildung 32.4).

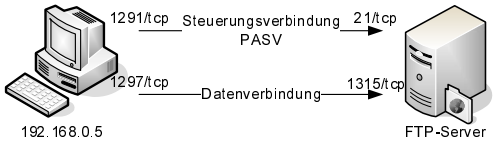


Abbildung 32.4: **Der Server baut bei dem passiven CE U FTP die Datenverbindung auf.**

Für den Administrator einer Firewall stellen sich nun mehrere Probleme. Stellen wir uns vor, Sie möchten den FTP-Client mit Ihrer Firewall schützen. Dies ist sicherlich der Standardfall.

Sie werden mit folgenden Problemen konfrontiert:

» Das FTP-Protokoll baut eine Vielzahl von Verbindungen auf.

» Einige der verwendeten Zielports und -adressen werden dynamisch während der Verbindung ausgehandelt und stehen nicht zu Beginn bereits fest! Eine Berücksichtigung in statischen Regeln ist nicht möglich.

» Bei dem aktiven FTP müssen Sie Verbindungen von dem Server zu Ihrem geschützten FTP-Client durch die Firewall erlauben. Dabei kennen Sie den Zielport der Verbindung nicht. Es handelt sich um einen beliebigen Port >1023.

» Bei dem passiven FTP ist es nicht ganz so schlimm. Dennoch müssen Sie beliebige Verbindungen von Ihrem Client zum Server unter Verwendung beliebiger Ports > 1023 erlauben. Achtung, auch KaZaA, Edonkey etc. verwenden diese Ports.

CE U Korrekt?

Editor's or typesetter's annotations (will be removed before the final TEX run)

» Da Ihr Firewall-Skript nicht in der Lage ist, einen Zusammenhang zwischen den Steuerungsverbindungen und den Datenverbindungen zu erstellen, müssen Sie in Ihrem Skript sogar die Datenverbindungen erlauben, wenn gar keine Steuerungsverbindung existiert. Schlimmer noch, Sie müssen die Datenverbindungen von und zu beliebigen Servern und Clients erlauben!

Um diese Probleme zu beseitigen, wurde die Stateful Inspection entwickelt.

32.10.2 Stateful Inspection des FTP-Protokolls

Die Stateful Inspection des FTP-Protokolls wurde ursprünglich in kommerziellen Firewalls entwickelt. Sie wird seit dem Kernel 2.4 auch von Linux unterstützt. Hierbei passiert nun Folgendes: Sobald der Linux-Kernel erkennt, dass eine FTP-Steuerungsverbindung durch die Firewall erlaubt wurde, betrachtet er jedes Paket dieser Verbindung. Dabei achtet er speziell auf das `PORT`- und das `PASV`-Kommando. Sobald der Kernel eines dieser Kommandos erkennt, trägt er die resultierende Verbindung selbstständig und automatisch in der Zustandstabelle (Connection Tracking Table) ein. Die Stateful Inspection erfordert auch den Einsatz des Connnection Tracking. Diese Verbindung und alle zu ihr gehörenden Pakete erhalten nun den Zustand `RELATED`, da sie mit einer anderen Verbindung (der Steuerungsverbindung) verwandt sind.

INFO

Diese verwandten Verbindungen werden auch Expectations genannt, da es sich um erwartete Verbindungen handelt. Daher werden diese Verbindungen auch nicht in der eigentlichen Zustandstabelle eingetragen. Der Kernel verwaltet eine eigene Tabelle für diese Expectations.

Um nun die Stateful Inspection für das FTP-Protokoll in Ihrer Linux-Firewall anzuschalten, genügt es, das entsprechende Kernelmodul `ip_conntrack_ftp` zu laden. In Abhängigkeit der Kernelversion lautet der Name des Moduls auch `nf_conntrack_ftp`. Dieses Modul erkennt automatisch die FTP-Steuerungsverbindung und beobachtet die transportierten FTP-Befehle. Es unterstützt sowohl aktives als auch passives FTP. Die Erkennung erfolgt über den TCP-Zielport 21. Wenn Sie auf FTP-Server zugreifen wollen, die einen anderen Port für die Steuerungsverbindung verwenden beziehungsweise die aktive Datenverbindung von einer anderen IP-Adresse aufbauen, müssen Sie dieses Modul bei dem Laden mit dem Befehl `modprobe` auch noch konfigurieren.

Der Befehl `modinfo` zeigt Ihnen die Möglichkeiten des Moduls `nf_conntrack_ftp`:

```
[root@bibo ~]# modinfo nf_conntrack_ftp
filename:       /lib/modules/2.6.35.11-83.fc14.x86_64/kernel/net/ ↵
        netfilter/nf_conntrack_ftp.ko
alias:          nfct-helper-ftp
alias:          ip_conntrack_ftp
description:    ftp connection tracking helper
author:         Rusty Russell <rusty@rustcorp.com.au>
license:        GPL
srcversion:     34766728C20755A54E1DBFF
```

```
depends:
vermagic:       2.6.35.11-83.fc14.x86_64 SMP mod_unload
parm:           ports:array of ushort
parm:           loose:bool
```

INFO

Hier können Sie auch erkennen, dass das Modul `nf_conntrack_ftp` auch über den älteren Namen `ip_conntrack_ftp` geladen werden kann. Es existiert ein entsprechender Alias.

Bei dem Laden des Moduls können Sie verschiedene Parameter angeben. Wenn Sie zum Beispiel auf einen FTP-Server zugreifen möchten, der auf dem Port 2121 seine Dienste anbietet, so sollten Sie das Modul mit folgendem Befehl in Ihrem Skript laden:

```
MODPROBE=/sbin/modprobe
$MODPROBE nf_conntrack_ftp ports=21,2121
```

Vergessen Sie bitte nicht, auch den Port 21 anzugeben. Ansonsten können Sie nicht mehr mit regulären FTP-Servern kommunizieren. Sie können maximal 8 Ports angeben.

Die Option `loose=1` erlaubt es, mit FTP-Servern zu kommunzieren oder FTP-Server zu betreiben, die für die Datenverbindung eine andere IP-Adresse verwenden als für die Steuerungsverbindung. Normalerweise ist dieser Parameter jedoch nicht erforderlich. Wenn Sie aber zu einem bestimmten FTP-Server absolut keine Verbindung aufbauen können, lohnt es sich, den Parameter auszuprobieren. Leider ist es nicht möglich, diesen Parameter nur für eine bestimmte IP-Adresse anzuschalten.

Befindet sich der Client oder der FTP-Server hinter einer Firewall, die gleichzeitig auch noch eine Network Address Translation (NAT) durchführt, dann müssen die Datenverbindungen auch noch korrekt genattet werden. Hierzu gibt es ein weiteres Kernelmodul, das auch die korrekte Adressumsetzung innerhalb des Protokolls garantiert. Auch dieses Modul, `nf_nat_ftp`, müssen Sie selbst laden, wenn Sie die Funktionalität wünschen. Es wird nicht automatisch von dem Kernel geladen.

```
$MODPROBE nf_nat_ftp
```

Bei älteren Kerneln (<2.6.11) müssen Sie bei diesem Modul genauso wie bei dem`ip_conntrack_ftp`-Modul die zu überwachenden Ports angegeben. Erst ab Kernel 2.6.11 fällt diese Anforderung weg und das Modul übernimmt die dort angegebenen Ports.

32.10.3 Iptables-Regeln

Bei der Anwendung der Stateful Inspection sind die Regeln nun sehr einfach. Zunächst betrachten wir die Regeln für eine Firewall, die Clients den Aufbau einer FTP-Verbindung erlaubt.

```
$MODPROBE nf_conntrack_ftp   # Aktiviert die Stateful Inspection
$MODPROBE nf_nat_ftp         # Nur bei NAT
```

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp --dport 21 -m state --
        state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Einige Leser werden sich wahrscheinlich verwundert die Augen reiben. Nach den komplizierten Ausführungen über das FTP-Protokoll sollen die Regeln nun so einfach sein? Nun, das ganze Geheimnis liegt in dem Modul `nf_conntrack_ftp`. Durch das Laden des Moduls aktivieren Sie die Stateful Inspection für das FTP-Protokoll. Alle Datenverbindungen werden erkannt und mit dem Zustand `RELATED` in der Verbindungstabelle eingetragen. Die zweite Regel akzeptiert alle Verbindungen mit diesem Zustand. Sie müssen sich als Firewall-Administrator nicht mehr mit aktivem oder passivem FTP herumschlagen. Darum kümmert sich nun Ihre Firewall. Dabei ist diese Behandlung auch besonders sicher, da nun nur dann eine Datenverbindung erlaubt wird, wenn diese zuvor durch eine Steuerungsverbindung angefordert wurde. Obwohl für FTP-Datenverbindungen nun die Ports >1023 geöffnet sind, können Anwendungen wie KaZaA oder Edonkey diese nicht nutzen. Sie müssten zunächst eine FTP-Steuerungsverbindung etablieren.

Sie sollten zusätzlich auch folgende Regel auf Ihrer Firewall eintragen:

```
$IPTABLES -A INPUT -p tcp --dport 113 -j REJECT
```

Viele FTP-Server (besonders auf dem Betriebssystem UNIX) bauen bei einer FTP-Verbindung eine Identd-Verbindung zum Client auf. Wenn Sie diese Verbindung auf Ihrer Firewall nur verwerfen (DROP), kommt es zu einer Verzögerung bei Ihrem Verbindungsaufbau. Lehnen Sie die Verbindung hingegen ab, erkennt der Server, dass kein Aufbau möglich ist, und fährt mit Ihrer Anmeldung fort.

Weitere Dienste, die dieses Verhalten zeigen können, sind SMTP-Server und IRC-Server.

Das Laden des `nf_nat_ftp`-Moduls ist nur erforderlich, wenn Sie auf Ihrer Firewall auch gleichzeitig ein NAT durchführen.

Wenn Sie einen FTP-Server mit einer lokalen Firewall schützen möchten, können Sie auch das recht einfach mit der Stateful Inspection erreichen. Hierzu verwenden Sie folgende Regeln:

```
$MODPROBE nf_conntrack_ftp   # Aktiviert die Stateful Inspection

$IPTABLES -A INPUT -p tcp --dport 21 -m state --state NEW -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.11 SNMP

Das Simple Network Management Protocol (SNMP) ist ein sehr einfaches Protokoll für die Überwachung und Verwaltung von Netzwerkkomponenten wie zum Beispiel Routern. Dieses Protokoll erlaubt das Auslesen von Leistungsdaten und die Konfiguration des Geräts. Da sensible Daten von SNMP in den Versionen 1 und 2 im Klartext übertragen werden, sollten diese

Protokolle nicht in ungeschützten Netzwerken eingesetzt werden. Hier sollte SNMPv3 genutzt werden. Dieses bietet eine sichere Authentifizierung und Verschlüsselung.

Meistens baut der Client (Manager) die Verbindung zum Server (Agent) auf. Jedoch kann auch der Agent sich in wichtigen Fällen bei dem Manager melden. Dies bezeichnet man als SNMP-Trap. Als Transportprotokoll verwendet SNMP das UDP-Protokoll. Während die normalen Abfragen an den UDP-Port 161 gesendet werden, werden SNMP-Traps an den UDP-Port 162 gesendet.

Bei gleichzeitiger Verwendung der Network Address Translation auf der Firewall gibt es jedoch Probleme, da die IP-Adresse des SNMP-Systems auch in dem Paket transportiert wird. NAT tauscht jedoch nur die Adresse in dem IP-Header aus. Wenn Sie auch einen Austausch der IP-Adresse in der SNMP-Payload des Pakets wünschen, müssen Sie ein NAT-Helfermodul (`nf_nat_snmp_basic`) laden:

```
$MODPROBE nf_nat_snmp_basic
```

Mit der Option `debug=1` können Sie zusätzliche Protokolleinträge bei der Arbeit des Moduls erhalten.

32.11.1 Iptables-Regeln

Die Regeln für SNMP sind sehr einfach. Wenn Ihr Network-Management-System durch eine Firewall geschützt ist, genügen die folgenden Regeln:

```
MANAGER=192.168.0.5
AGENT=192.168.0.6

# Erlaube den SNMP-Zugriff
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -s $MANAGER -d $AGENT -p udp ↵
        --dport 161 -m state --state NEW -j ACCEPT

# Erlaube SNMP-Traps
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -s $AGENT -d $MANAGER -p udp ↵
        --dport 162 -j ACCEPT

$IPTABLES -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Wenn Sie auf Ihrer Firewall gleichzeitig Source-NAT verwenden, sollten Sie daran denken, dass Sie für die SNMP-Traps ein Destination-NAT zu Ihrer Management-Station einrichten müssen. Sobald Sie SNAT oder DNAT einsetzen, sollten Sie zusätzlich das Modul `nf_nat_snmp_basic` laden. Dieses Modul ist in allen aktuellen Linux-Distributionen enthalten.

32.12 Amanda

Der Advanced Maryland Automatic Network Disc Archiver (Amanda) ist ein Backup-System, mit dem automatisch verteilte UNIX-Clients über ein Netzwerk auf einem zentralen Backup-

Server gesichert werden können (http://www.amanda.org). Dieses System wird in vielen Umgebungen eingesetzt, da es unter der GPL veröffentlicht worden ist.

Amanda zeichnet sich dadurch aus, dass der Server sich bei Bedarf mit dem Client verbindet und ihn auffordert, jetzt seine Daten für die Sicherung bereitzustellen. So kommt es auf dem Server nicht zur Überlastung, und der Server kann auch die genutzte Netzwerkbandbreite regulieren.

32.12.1 Das Amanda-Protokoll

Amanda verwendet ein recht kompliziertes Protokoll für das Backup. Der Server verbindet sich über das UDP-Protokoll auf Port 10080 auf dem Client mit dem Amandad. Der Client forkt einen Amandad-Prozess, der sich anschließend mit einem vom Server bestimmten UDP-Port auf dem Server verbindet. Anschließend verbindet sich der Server wieder mit zwei oder drei TCP-Ports auf dem Client. Auch diese Ports werden zwischen dem Amanda-Client und dem Backup-Server ausgehandelt.

Die Dynamik der Ports ist für jeden Firewall-Administrator ein Gräuel. Sie können nicht in Ihrer Firewall feste Regeln für die Verbindung definieren. Das Amanda-Handbuch beschreibt in Kapitel 21, wie Sie die Ports auf einen bestimmten Bereich einschränken können (http://www.amanda.org/docs/portusage.html). Dies ist aber auch keine richtige Lösung.

32.12.2 Iptables-Regeln

Eine richtige Filterung des Amanda-Netzwerkverkehrs ist nur unter Anwendung der Stateful Inspection möglich. Diese wurde bereits im Kapitel zu FTP (siehe Abschnitt 32.10) besprochen. Auch für Amanda gibt es zwei Kernel-Module, die die einfache Filterung des Protokolls erlauben:

```
$MODPROBE nf_conntrack_amanda
$MODPROBE nf_nat_amanda
```

Das Modul nf_conntrack_amanda verfügt über die Option master_timeout. Hiermit können Sie den Timeout für die Zustandsüberwachung der Master-Verbindung auf Port 10080 einstellen. Der übliche Timeout für die Master-Verbindungen beträgt 300 Sekunden und ist häufig zu kurz. Dies ist aber bereits mehr, als üblicherweise von Iptables für UDP-Verbindungen genutzt wird (180 Sekunden). Sie können hier aber auch eine Stunde angeben:

```
$MODPROBE nf_conntrack_amanda master_timeout=3600
```

Die weiteren Regeln gestalten sich dann sehr einfach:

```
AM_SERVER=192.168.0.5
AM_CLIENTS=192.168.1.0/24

$IPTABLES -A FORWARD -s $AM_SERVER -d $AM_CLIENTS -p udp --dport 10080 -m ↵
         state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Wenn Sie eine lokale Firewall auf dem Amanda-Server betreiben möchten, können Sie folgenden Regelsatz verwenden:

```
$MODPROBE nf_conntrack_amanda master_timeout=3600
$MODPROBE nf_nat_amanda

AM_CLIENTS=192.168.1.0/24

$IPTABLES -A OUTPUT -d $AM_CLIENTS -p udp --dport 10080 -m state --state ↵
        NEW -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.13 PPTP

Das Point-to-Point-Protokoll wurde häufig für den Aufbau von Einwahl- und VPN-Verbindungen genutzt. Es verwendet einen Steuerungskanal auf dem TCP-Port 1723 und einen Datenkanal, der das GRE-Protokoll (IP-Protokoll 47) nutzt. Dabei erlaubt das PPTP-Protokoll die Zuweisung von IP-Adresse, Netzmaske, Route, DNS- und WINS-Servern.

Grundsätzlich ist die Konfiguration einer Firewall für die Anwendung des PPTP-Protokolls sehr einfach. Schwierig wird die gesamte Konstruktion nur bei gleichzeitiger Anwendung der Network Address Translation (NAT). Sobald Sie eine Firewall einsetzen, die ein Source-NAT für die dahinter befindlichen Clients durchführt, und gleichzeitig zwei oder mehr Clients mit dem PPTP-Protokoll auf einen Server zugreifen möchten, werden beide Verbindungen nicht mehr richtig funktionieren. Dieser Fall ist, nebenbei bemerkt, sehr häufig. In jedem WLAN-Hotspot im Hotel und Flughafen kommt es regelmäßig zu diesem Szenario.

Warum ist Source-NAT kritisch? Die Firewall muss sich bei einem Source-NAT merken, welcher Client die Verbindung aufgebaut hat, um anschließend die Antwort-Pakete der Verbindung auch an den richtigen Client zurückzusenden. Hierzu pflegt die Firewall eine Tabelle, in der alle genatteten Verbindungen eingetragen werden.[2] Sobald ein Paket zurückkommt, schaut die Firewall in die Tabelle, um nachzusehen, an welchen Client das Paket intern weiterzusenden ist. Hierbei verwendet die Firewall als Kriterium das IP-Protokoll, die Zieladresse und den Zielport des Pakets. Leider besitzt das GRE-Protokoll keine Ports. Daher bleiben als Unterscheidungsmerkmale nur das IP-Protokoll (immer GRE) und die Zieladresse (immer die der Firewall). Sobald also zwei GRE-Verbindungen gleichzeitig genattet werden müssen, kann die Firewall die Pakete nicht mehr auseinanderhalten und weist sie immer der zuletzt aktiven Verbindung zu. Dieses Problem lässt sich mit der Stateful Inspection lösen.

2 Bei Linux ist dies auch die Connection-Tracking-Tabelle.

32.13.1 Iptables-Regeln

Die Iptables-Regeln sind recht einfach. Möchten Sie mit Ihrer Firewall einen Client bei dem Zugriff auf einen PPTP-Server im Internet schützen, so können Sie folgende Regeln verwenden:

```
PPTP_SRV=5.0.0.1

$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp --dport 1723 -d ↵
        $PPTP_SRV -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p gre -m state -d $PPTP_SRV ↵
        --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Falls Sie nur die zweite GRE-Verbindung erlauben wollen, wenn diese durch die erste Verbindung angefordert wurde, können Sie die Stateful Inspection nutzen.

Sie benötigen die Module `nf_conntrack_proto_gre` und `nf_conntrack_pptp`:

```
$MODPROBE nf_conntrack_proto_gre
$MODPROBE nf_conntrack_pptp
```

Dann genügen die folgenden Firewall-Regeln:

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp --dport 1723 -d ↵
        $PPTP_SRV -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Sobald die PPTP-Steuerungsverbindung erkannt wurde, wird die entsprechende GRE-Verbindung im Zustand `RELATED` erlaubt.

Sobald nun aber mehrere Clients mit dem PPTP-Protokoll gleichzeitig zugreifen möchten, wird die Verbindung wieder fehlschlagen. Hier hilft ein NAT-Helper-Modul, das ebenfalls nicht in allen Distributionen enthalten ist. Mit dem obigen Patch stehen diese Module aber zur Verfügung.

```
$MODPROBE nf_nat_proto_gre
$MODPROBE nf_nat_pptp
```

Natürlich kann es auch sein, dass Sie einen PPTP-Server in Ihrer DMZ betreiben möchten. Dann benötigen Sie die folgenden Regeln:

```
PPTP_SRV=192.168.0.5

$IPTABLES -t nat -A PREROUTING -p tcp --dport 1723 -j DNAT --to- ↵
        destination $PPTP_SRV
$IPTABLES -t nat -A PREROUTING -p gre -j DNAT --to-destination $PPTP_SRV
```

```
$IPTABLES -A FORWARD -d $PPTP_SRV -p tcp --dport 1723 -m state --state ↵
        NEW -j ACCEPT
$IPTABLES -A FORWARD -d $PPTP_SRV -p gre -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Die ersten beiden DNAT-Regeln sorgen dafür, dass der PPTP-Verkehr, der von außen die Firewall erreicht, an den PPTP-Server weitergereicht wird.

32.14 SMTP

Das Simple Mail Transport Protocol (SMTP) wird für den Transport von E-Mail zwischen den E-Mail-Servern im Internet genutzt. Auch E-Mail-Clients wie Evolution, Kontact und Outlook nutzen dieses Protokoll für den Versand von E-Mail.

32.14.1 Das SMTP-Protokoll

Das SMTP-Protokoll ist ein Klartextprotokoll und verwendet Befehle, die vier Buchstaben lang sind. Im Folgenden ist eine Beispielsitzung abgedruckt. Sie können das SMTP-Protokoll mit einem Telnet-Client und ein wenig Wissen um die Befehle leicht selbst testen.

```
[root@bibo ~]# telnet mail.spenneberg.net 25
Trying 217.160.128.61...
Connected to mail.spenneberg.net (217.160.128.61).
Escape character is '^]'.
220 mail.spenneberg.net ESMTP Postfix
helo linuxclient.spenneberg.net  (1)
250 mail.spenneberg.net
mail from: ralf@gmx.de   (2)
250 Ok
rcpt to:ralf@spenneberg.net   (3)
250 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: SMTP-Test  (4)
From: Ich bins <ralf@gmx.de>
To: Ralf

Test   (5)
.    (6)
250 Ok: queued as 556B857AAA
quit
221 Bye
Connection closed by foreign host.
```

Zunächst begrüßt der Client den Server und stellt sich selbst vor (1). Anschließend nennt der Client die Envelope-Absenderadresse (2). Dies ist vergleichbar mit der Adresse außen auf einem Briefumschlag. Dann gibt er den Envelope-Empfänger an (3). Es folgt der Inhalt der E-Mail (Inhalt des Briefumschlages). Dort gibt der Client im Header einen Betreff, den Absender und den Empfänger an. Diese Angaben sind optional und können auch weggelassen werden. Das Verhalten des Mailservers bei fehlenden Angaben ist von dem Produkt abhängig und kann nicht allgemeingültig angegeben werden. Abgetrennt durch eine Leerzeile folgt dann der tatsächliche Inhalt der E-Mail (5). Die Eingabe wird durch einen Punkt auf einer ansonsten leeren Zeile beendet (6). Der Client beendet die Verbindung mit `quit`.

Das SMTP-Protokoll überträgt normalerweise sämtliche Informationen im Klartext. Jedoch gibt es mit SMTP über SSL (smtps, Port 465) auch die Möglichkeit, SMTP über die Secure Socket Layer zu verwenden. Dies hat sich jedoch nicht durchgesetzt. Heute beherrschen fast alle Mailserver als Alternative die Transport Layer Security (TLS). Nachdem das Kommando `STARTTLS` auf dem unverschlüsselten Wege übertragen wurde, handeln der Client und der Server innerhalb der bestehenden Verbindung eine TLS-Verbindung aus. Die Verbindung wird nicht unterbrochen oder auf einem anderen Port geöffnet. Hierfür genügt der Standardport 25/tcp.

Häufig wird heute für die Zustellung von E-Mails durch Mailclients an den Mailserver zusätzlich der Port 587/tcp (*submission*) genutzt. Diesen müssen Sie dann auch erlauben.

32.14.2 Iptables-Regeln

Die Behandlung des SMTP-Protokolls in einer Firewall ist sehr einfach. Da das Protokoll wie Telnet, HTTP und SSH nur einen einzigen Port verwendet, genügt eine Zeile, um das Protokoll über eine Firewall zu erlauben.

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp -m multiport --dport ↵
        25,587 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Sie sollten zusätzlich auf Ihrer Firewall Identd-Anfragen ablehnen. Ansonsten kann es bei dem Aufbau einer SMTP-Verbindung zu Verzögerungen kommen, da einige SMTP-Server bei einer Verbindungsanfrage zunächst eine Identd-Verbindung zum Client aufbauen und erst anschließend fortfahren. Erhalten sie eine Fehlermeldung, fahren sie ebenfalls sofort fort.

```
$IPTABLES -A INPUT -i $EXTDEV -p tcp --dport 113 -j REJECT
```

Wenn Sie einen Mailserver direkt mit einer lokalen Firewall schützen möchten, können Sie die folgenden Regeln einsetzen:

```
$IPTABLES -A INPUT -p tcp -m multiport --dport 25,587 -m state --state ↵
        NEW -j ACCEPT
$IPTABLES -A OUTPUT -p tcp --dport 25 -m state --state NEW -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Ein E-Mailserver benötigt zusätzlich auch noch Zugang zu DNS-Diensten. Denken Sie daran, ihm auch diesen Zugriff zu gestatten!

```
$IPTABLES -A OUTPUT -p tcp --dport 53 -m state --state NEW -j ACCEPT
$IPTABLES -A OUTPUT -p udp --dport 53 -m state --state NEW -j ACCEPT
```

Weitere Beispiele für die Konfiguration einer Firewall beim Einsatz eines E-Mailservers in einer DMZ finden Sie in Kapitel 8.

32.15 IRC

Das Protokoll Internet Relay Chat (IRC) ist ein Kommunikationsmedium, das es ermöglicht, weltweit mit anderen Benutzern textgestützt in Echtzeit zu kommunizieren. Realisiert wird es von einem verteilten Netzwerk aus miteinander vernetzten Servern. Dabei erfolgt die Kommunikation in thematisch organisierten Chaträumen, sogenannten Channels. Die Channels tragen Namen wie #Linux oder #VPN. Bei den größeren Channels sind mehrere Tausend Benutzer gleichzeitig angemeldet. Die Anfänge gehen auf das BITNET zurück, für das das Relay-Chat-Protokoll entwickelt wurde. Dieses wurde 1988 auf das Internet übertragen. Aufgrund von organisatorischen Problemen entstanden ab 1993 mehrere unabhängige Netzwerke. 1996 erfolgte die Aufspaltung in zwei große Netze, die heute unter den Namen IRCNet und EFnet wiederzufinden sind. Aufgrund der besonderen Struktur genügt es, wenn der Client Zugang zu einem Server dieser Netzwerke hat, um an sämtlichen Chaträumen teilzunehmen. Weitere heute übliche Netze sind QuakeNet, Undernet, DALnet, Freenode etc.

Das IRC-Protokoll erlaubt neben dem reinen Chat auch den Austausch von Dateien über eigene dedizierte Verbindungen (ähnlich den Datenverbindungen bei FTP). Diese Verbindungen werden Direct-Client-to-Client-Verbindung (DCC) genannt. Die für diese Verbindungen verwendeten Ports werden von dem Client und Server dynamisch ausgehandelt. Um diese Verbindungen sicher zuzulassen, bietet der Linux-Kernel ein Stateful-Inspection-Modul namens `nf_conntrack_irc`.

32.15.1 Iptables-Regeln

Die Iptables-Regeln für den IRC-Verkehr sind bei Anwendung der Stateful Inspection sehr einfach. Es genügt, das Modul zu laden und anschließend die Verbindung zum IRC-Server zuzulassen.

```
$MODPROBE nf_conntrack_irc ports=6667
$MODPROBE nf_nat_irc  # bei gleichzeitiger Anwendung von NAT

$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp --dport 6667 -m state ↵
        --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Das `nf_conntrack_irc`-Modul erkennt automatisch angekündigte DCC-Verbindungen und trägt diese mit dem Zustand `RELATED` in der Verbindungstabelle als Expectation ein. Diese

Verbindung wird dann von der zweiten Regel auch erlaubt. Bei dem Laden des Moduls können Sie einige Werte als Parameter angeben. Zunächst ist dies der Port, auf dem der IRC-Server angesprochen wird: `ports=6667,6668`. Der Parameter `max_dcc_channels` gibt die maximale Anzahl gleichzeitiger DCC-Verbindungen an (Default 8). Der Parameter `dcc_timeout` definiert schließlich den Timeout für nicht beantwortete DCC-Verbindungen (Default: 300 Sekunden).

32.16 TFTP

Das Trivial File Transfer Protocol (TFTP) hat mit dem FTP-Protokoll nur den Namen gemeinsam. Es verwendet das UDP-Protokoll und den Port 69. Außerdem kennt es keine Anmeldung. Es wird häufig eingesetzt, um auf Netzwerkkomponenten eine neue Firmware einzuspielen oder um Konfigurationseinstellungen zu laden. Auch das Preboot Execution Environment (PXE) verwendet TFTP.

Eine Implementierung des Protokolls in einer Firewall ist daher selten, aber soll trotzdem hier kurz vorgestellt werden, da es ebenfalls dynamisch zugewiesene Ports verwendet.

Der Client sendet eine Anfrage (Request: RRQ/WRQ) an den Server auf Port 69. Der Client verwendet in seiner Anfrage einen beliebigen Port >1023. Der Server antwortet und verwendet in seiner Antwort als Sourceport nicht 69, sondern ebenfalls einen Port >1023. Die erste Antwort wird also bereits mit einem neuen Port gesendet. Der Client erkennt die Antwort lediglich an dem gleich gebliebenen Zielport. Die nun gewählten Ports bleiben für die gesamte Übertragung der Datei identisch (siehe Abbildung 32.5).

Eine Implementierung dieses Protokolls in einer Firewall ist nur möglich, wenn die Stateful Inspection dieses Protokoll unterstützt. Ansonsten müssen ernorme Löcher in der Firewall geöffnet werden, damit von einem beliebigen UDP-Port eine Verbindung zu einem anderen beliebigen UDP-Port aufgebaut werden kann.

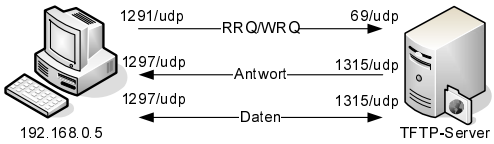


Abbildung 32.5: **Das TFTP-Protokoll verwendet beliebige Ports.**

32.16.1 Iptables-Regeln

Erfreulicherweise unterstützen die meisten Distributionen in ihrem Kernel bereits das `ip_conntrack_tftp`-Modul. Wenn dies bei Ihrer Distribution nicht der Fall ist, müssen Sie Ihren Kernel neu übersetzen.

Wenn Sie dieses Modul laden, können Sie analog dem FTP-Stateful-Inspection-Modul maximal acht Ports angeben, auf denen Sie eine Verbindung zu einem TFTP-Server aufbauen.

Wenn die Verbindung gleichzeitig genattet wird, müssen Sie auch das Modul `nf_nat_tftp` laden. Dieses Modul benötigt in einer älteren Ausgabe auch die Angabe der Ports. Das können Sie aber selbst prüfen:

```
[root@bibo ~]# modinfo nf_nat_tftp
filename:       /lib/modules/2.6.35.11-83.fc14.x86_64/kernel/net/ipv4/ ↵
        netfilter/nf_nat_tftp.ko
alias:          ip_nat_tftp
license:        GPL
description:    TFTP NAT helper
author:         Magnus Boden <mb@ozaba.mine.nu>
srcversion:     DFB7D90EA3DE94CF9A0EA9A
depends:        nf_conntrack_tftp,nf_nat
vermagic:       2.6.35.11-83.fc14.x86_64 SMP mod_unload
```

Dieses Modul benötigt die Angabe nicht mehr. Nun können Sie mit der folgenden Regel TFTP-Verbindungen erlauben:

```
$MODPROBE nf_conntrack_tftp
$MODPROBE nf_nat_tftp   # bei SNAT oder DNAT

$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p udp --dport 69 -m state -- ↵
        state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.17 IMAP

Das Internet Message Access Protocol (IMAP, siehe RFC3501) erlaubt den Zugriff auf und die Verwaltung von E-Mails und Dateien auf dem Server. Im Gegensatz zum POP3-Protokoll verbleiben die E-Mails bei dem IMAP-Protokoll auf dem Server und werden nicht nach einem Download gelöscht. Dadurch können Sie mit mehreren Clients von unterschiedlichen Orten die E-Mails lesen und bearbeiten.

Der IMAP-Server verwendet zum Anbieten seines Dienstes den TCP-Port 143. Verbindungen auf diesem Port übertragen die Daten normalerweise unverschlüsselt. Lediglich die Authentifizierung kann verschlüsselt erfolgen. Allerdings kann auch das IMAP-Protokoll auf die Dienste des SSL-Protokolls zurückgreifen. Der IMAP-Server bietet dann IMAP-über-SSL-Verbindungen auf dem TCP-Port 993 an. Zusätzlich unterstützen moderne IMAP-Server die Transport Layer Security. Hier wird nach einem `STARTTLS`-Kommando die Verbindung auf dem Port 143/tcp verschlüsselt.

Die Verwendung des `STARTTLS`-Kommandos und der TLS ist optional. Wenn Sie eine verschlüsselte Verbindung erzwingen möchten, müssen Sie auf SSL (Port 993/tcp) ausweichen.

Sie können das IMAP-Protokoll selbst mit einem Telnet-Client nachstellen:

```
[root@bibo ~]# telnet mail.spenneberg.net 143
Trying 217.160.128.61...
Connected to mail.spenneberg.net (217.160.128.61).
Escape character is '^]'.
* OK dovecot ready.
a001 LOGIN test test
a001 OK Logged in.
a142 SELECT INBOX
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* OK [PERMANENTFLAGS (\Answered \Flagged \Deleted \Seen \Draft \*)] Flags ↵
         permitted.
* 0 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 1122315709] UIDs valid
* OK [UIDNEXT 1] Predicted next UID
a142 OK [READ-WRITE] Select completed.
a752 close
a752 OK Close completed.
```

32.18 Iptables-Regeln

Die Iptables-Regeln sind angesichts des einfachen Protokolls sehr einfach. Hier sind zunächst die Regeln für ein Gateway:

```
IMAP_PORTS="143,993"

$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp -m multiport --dport ↵
        $IMAP_PORTS -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Wenn Sie einen IMAP-Server mit einer lokalen Firewall schützen möchten, sehen die Regeln analog wie folgt aus:

```
IMAP_PORTS="143,993"

$IPTABLES -A INPUT -p tcp -m multiport --dport $IMAP_PORTS -m state -- ↵
        state NEW -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.19 POP3

Das Post Office Protocol (POP, siehe RFC1939) wird von einem E-Mail-Client verwendet, um E-Mails von einem Server abzuholen. Dazu werden die E-Mails anschließend üblicherweise auf dem Server gelöscht. Das ASCII-Protokoll verwendet zur Steuerung vier Buchstaben lange Befehle, die an den Server auf dem Port 110/tcp gesendet werden. Auch POP3 kann wie IMAP über eine SSL-gesicherte Verbindung genutzt werden. Dann verwendet der Server den TCP-Port 995.

Das POP3-Protokoll überträgt sämtliche Daten unverschlüsselt. Lediglich die Anmeldung kann bei Unterstützung des `APOP`-Befehls verschlüsselt erfolgen. Leider unterstützen nur sehr wenige Anbieter den `APOP`-Befehl.

Sie können das POP3-Protokoll ganz einfach mit einem Telnet-Client testen:

```
[root@bibo ~]# telnet mail.spenneberg.net 110
Trying 217.160.128.61...
Connected to mail.spenneberg.net (217.160.128.61).
Escape character is '^]'.
+OK dovecot ready.
user test
+OK
pass test
+OK Logged in.
list
+OK 1 messages:
1 754
.
retr 1
+OK 754 octets
Return-Path: <root@bibo.spenneberg.de>
Received: from bibo.spenneberg.de (bibo.spenneberg.de [127.0.0.1])
        by bibo.spenneberg.de (8.13.4/8.13.4) with ESMTP id ↵
                j6PIRfDM018622
        for <test@bibo.spenneberg.de>; Mon, 25 Jul 2005 20:27:41 +0200
... gelöscht ...

.
dele 1
+OK Marked to be deleted.
quit
+OK Logging out, messages deleted.
Connection closed by foreign host.
```

32.19.1 Iptables-Regeln

Die Iptables-Regeln für das POP3-Protokoll sind sehr einfach und entsprechen denen für das IMAP-Protokoll. Hier sind zunächst die Regeln für ein Gateway:

```
POP3_PORTS="110,995"

$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp -m multiport --dport ↵
        $POP3_PORTS -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Wenn Sie einen POP3-Server mit einer lokalen Firewall schützen möchten, sehen die Regeln analog wie folgt aus:

```
POP3_PORTS="110,995"

$IPTABLES -A INPUT -p tcp -m multiport --dport $POP3_PORTS -m state -- ↵
        state NEW -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.20 NTP

Das Network Time Protocol (NTP) dient zur Synchronisation der Systemzeit mit einer oder mehreren zentralen Zeitquellen (siehe RFC2030). Das NTP-Protokoll unterstützt eine Authentifizierung des Servers, um Spoofing-Angriffen vorzubeugen. Leider bieten viele öffentliche NTP-Server diese Authentifizierung nicht an.

Das NTP-Protokoll verwendet den UDP-Port 123 für den Austausch der Informationen. Wenn NTP-Server miteinander kommunizieren, verwenden sie auch als Client den Port 123. Alle anderen Clients, die zur Synchronisation eingesetzt werden, verwenden einen UDP-Port >1023.

32.20.1 Iptables-Regeln

Die Regeln für die Unterstützung des NTP-Protokolls sind recht einfach. Die folgenden Regeln erlauben das NTP-Protokoll durch ein Gateway:

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p udp --dport 123 -m state -- ↵
        state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.21 NNTP

Das Network News Transfer Protocol (NNTP, siehe RFC997) dient zum Transport von Nachrichten zu und von einem Newsserver. Das textbasierte Protokoll erlaubt das Auflisten der Newsgruppen, das Anfordern der vorhandenen Artikel und das Posten weiterer Artikel.

Das NNTP-Protokoll verwendet den TCP-Port 119. Um eine sichere und verschlüsselte Übertragung der Daten zu ermöglichen, bietet das NNTP-Protokoll eine Sicherung mit Secure Socket Layer (SSL). Hierzu wird der Port 563/tcp verwendet.

32.21.1 Iptables-Regeln

Die Iptables-Regeln für das NNTP-Protokoll sind sehr einfach und entsprechen denen für das IMAP-Protokoll. Hier sind zunächst die Regeln für ein Gateway:

```
NNTP_PORTS="119,563"

$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp -m multiport --dport ↵
        $NNTP_PORTS -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Wenn Sie einen NNTP-Server mit einer lokalen Firewall schützen möchten, sehen die Regeln analog wie folgt aus:

```
NNTP_PORTS="119,563"

$IPTABLES -A INPUT -p tcp -m multiport --dport $NNTP_PORTS -m state -- ↵
        state NEW -j ACCEPT
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

32.22 H.323

Das H.323-Protokoll ist in Wirklichkeit ein ganzer Satz von Protokollen, die von Programmen wie Microsoft Netmeeting oder Gnomemeeting verwendet werden, um Audio- und Video-Informationen über das Internet zu transportieren. Einen Überblick über die beteiligten Protokolle und Standards erhalten Sie auf `http://www.packetizer.com/voip/h323/standards.html`.

H.323 verwendet die folgenden Ports für seine unterschiedlichen Dienste:

Port	Beschreibung
389/tcp	Internet Locator Server
522/tcp	User Location Server
1503/tcp	T.120 Protocol
1720/tcp	H.323 (H.225 call setup)
1731/tcp	Audio Call Control
>1023/tcp	H.245 Call Control
>1023/udp	RTCP/RTP Streaming

Die Vielzahl der beteiligten Protokolle und die Verwendung von dynamischen Ports zeigt die Komplexität des Protokolls. Daher soll hier auf eine Beschreibung des Protokolls verzichtet

werden. Interessierte Leser finden unter der oben angegebenen URL ausführliche Informationen über das Protokoll.

32.22.1 Iptables-Regeln

Die Komplexität des Protokolls ist auf einer Firewall durch Einsatz der Stateful Inspection handhabbar. Erfreulicherweise gibt es auf den modernen Distributionen Netfilter-Kernel-Module, die dies unterstützen. Ob Ihre Distribution diese Module enthält, können Sie sehr leicht mit dem folgenden Befehl prüfen:

```
[root@bibo ~]# modinfo nf_conntrack_h323
modinfo: could not find module nf_conntrack_h323
```

In diesem Fall können Sie die Stateful Inspection nicht nutzen.

Anschließend müssen Sie die Module laden und mindestens die Verbindung zum TCP-Port 1720 öffnen, um internen Clients die Verbindung mit Gegenstellen im Internet zu erlauben.

```
$MODPROBE nf_conntrack_h323
$MODPROBE nf_nat_h323

$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp --dport 1720 -m state ↵
        --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Die Connection-Tracking-Helper-Module erkennen dann die Verbindungen auf den dynamischen Ports und tragen diese im Zustand `RELATED` in der Zustandstabelle ein.

Wenn Sie auch Verbindungen von außen entgegennehmen möchten, so ist ein DNAT des TCP-Ports 1720 nach innen erforderlich.

32.23 SIP

Das Session Initiation Protocol (SIP) wurde entwickelt, um allgemeine Sitzungen zwischen mehreren Clients auszuhandeln. Es wird im Moment in erster Linie für die Sprachübertragung über das Internet (VoIP, Voice over IP) verwendet. Für die Sprachübertragung verwendet das SIP-Protokoll zusätzlich das Real Time Transport Protocol (RTP). SIP handelt die Verbindung aus, und RTP überträgt tatsächlich die Sprachdaten.

Aufgrund der dynamischen Aushandlung der UDP-Ports für das RTP-Protokoll stellt SIP hohe Anforderungen an eine Firewall. Dies ist insbesondere der Fall, wenn auch NAT zum Zuge kommt, da auch in den Paketen die IP-Adressen der Clients transportiert werden. In der Vergangenheit wurden daher häufig für diesen Zweck SIP-Proxies eingesetzt.

32.23.1 Iptables-Regeln

Die Komplexität des Protokolls mit dynamisch ausgehandelten UDP-Ports verlangt für eine sichere Firewall-Implementierung die Unterstützung von Stateful Inspection. Da diese für den Linux-Kernel aber erst ab der Version 2.6.11 zur Verfügung steht, sollen hier zunächst die Regeln ohne Stateful Inspection für eine Firewall als Gateway vorgestellt werden.

```
# SIP verwendet den UDP-Port 5060
$IPTABLES  -A FORWARD -i $INTDEV -o $EXTDEV -p udp -m udp --dport 5060 -m ↵
         state --state NEW -j ACCEPT

# RTP - the media stream
iptables -A FORWARD -i $INTDEV -o $EXTDEV -p udp -m udp --dport  ↵
        10000:20000 -m state --state NEW -j ACCEPT -j ACCEPT

$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Falls Sie gleichzeitig auch NAT einsetzen, kann es in Abhängigkeit von dem Client aber wieder zu Problemen kommen.

Als Lösung können Sie die Stateful-Inspection-Module nutzen. Diese erkennen die dynamischen RTP-Verbindungen und führen ein korrektes NAT durch.

```
$MODPROBE nf_conntrack_sip
$MODPROBE nf_nat_sip

# SIP verwendet den UDP-Port 5060
$IPTABLES  -A FORWARD -i $INTDEV -o $EXTDEV -p udp -m udp --dport 5060 -m ↵
         state --state NEW -j ACCEPT

$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

33. L7-Filter

Der L7-Filter analysiert Pakete nicht durch Betrachtung der IP- und TCP-Header, sondern wendet stattdessen reguläre Ausdrücke auf den Inhalt der Pakete an. Dadurch kann er auch die Applikationen und Protokolle erkennen, die nicht auf ihren Standard-Ports betrieben werden bzw. diese Ports dynamisch aushandeln.

Die Analyse der Paket-Payload ist natürlich wesentlich aufwendiger als die Analyse der standardisierten Header. Daher benötigt der L7-Filter mehr CPU- und Speicherressourcen als ein einfacher Paketfilter. Sinnvoll ist daher die Anwendung des L7-Filters nur, wenn Sie mit einem normalen Paketfilter nicht die gewünschte Erkennung erreichen. Folgende Gründe sprechen für den L7-Filter:

- Sie möchten Verbindungen erkennen, die nicht vorhersagbare Ports verwenden. Das klassische Beispiel sind viele Peer-to-Peer-Protokolle.
- Sie möchten Verbindungen und Protokolle erkennen, die nicht den Standard-Port verwenden (z. B. Telnet auf dem Port 2323).
- Sie möchten sicherstellen, dass auf einem Port tatsächlich das richtige Protokoll gesprochen wird (z. B. soll der Port 80 HTTP-Verkehr und nicht Peer-to-Peer-Verkehr transportieren).

Leider gibt es auch Gründe, die gegen den Einsatz sprechen. Nicht nur sind die benötigten Ressourcen enorm – die Erkennung ist auch nicht eindeutig. Es ist kaum möglich, eine eindeutige Filterung mit dem L7-Filter zu implementieren. Wenn Sie den L7-Filter einsetzen möchten, um nur bestimmte Protokolle zuzulassen, werden Sie wahrscheinlich Schiffbruch erleiden, da die Erkennung nicht zu 100 % gelingt. Einzelne Pakete werden häufig von dem L7-Filter nicht korrekt erkannt.

Andersherum funktioniert es besser. Wenn Sie ein bestimmtes mit dem L7-Filter erkanntes Protokoll verbieten möchten, genügt es häufig, wenn Sie einen Großteil der Pakete verwerfen. Das Protokoll funktioniert dann nicht mehr. Dennoch kann es auch hier zu einer falschpositiven Erkennung von Paketen kommen, die das Protokoll gar nicht verwenden!

Speziell für die Bandbreitenregulation lässt sich der L7-Filter sehr gut einsetzen. Wenn Sie die Bandbreite oder Geschwindigkeit von 80 % der Pakete einschränken können, hat dies Auswirkungen auf alle Verbindungen des Protokolls. Dies ist dann auch die häufigste Anwendung des L7-Filters: Reduktion der Bandbreiten von unerwünschten Protokollen.

33.1 L7-Hintergründe

Der L7-Filter wurde 2003 ins Leben gerufen. Das Ziel war die Entwicklung einer OpenSource-Lösung für die Erkennung von Applikationsprotokollen, um anschließend diesen Verkehr in seiner Bandbreite und Geschwindigkeit kontrollieren zu können.

Die erste Version wurde im Mai 2003 vorgestellt. Hier wurde der QoS-Stack des Linux-Kernels um die Funktion erweitert. Diese Version wurde noch im selbem Jahr umgeschrieben, sodass sie nun im Netfilter-Stack arbeitete. Diese Variante wurde in den nächsten Jahren verbessert. Im Jahr 2006 wurde erstmals eine Version für den Userspace vorgestellt. Diese Version erhält ihre Daten über die Queue-Schnittstelle von Netfilter. So können Sie mit L7-Filter alle Möglichkeiten nutzen, die Netfilter bietet. Denken Sie jedoch daran, dass L7-Filter nie dafür gedacht war, tatsächlich zu filtern, sondern dass er Bandbreitenregulierungen unterstützen soll.

Heute existieren immer noch beide Versionen. So können Sie mit einem Patch L7-Filter in Ihren Kernel integrieren oder die L7-Filter-Userspace-Variante mit beliebigen Kerneln nutzen. Im Folgenden beschreibe ich beide Varianten.

33.2 L7: Kernel-Version

Die Installation ist kompliziert, da sie einen Patch des Linux-Kernels und des IPtables-Paketes voraussetzt. Keine mir bekannte Distribution hat diesen Patch bisher integriert. Auf Mehrprozessorsystemen kann es zu einer Kernel-Panic kommen. Darüber hinaus unterstützt diese Version nur einfache reguläre Ausdrücke. Ich empfehle daher zunächst die Userspace-Version (siehe Abschnitt 33.3 und 33.3.1).

33.2.1 Nutzung der Kernel-Version

Um diese Version zu nutzen, sollten Sie zunächst einen passenden Kernel herunterladen. Die aktuelle Version von L7-Filter unterstützt den Linux-Kernel 2.4 und die Linux-Kernel 2.6.25–2.6.28. Ältere Kernel werden ebenfalls unterstützt. Sie erhalten den neuesten Kernel-Patch unter `http://l7-filter.clearfoundation.com/downloads/start`. Dort finden Sie auch die Protokolldefinitionen, die Sie ebenfalls herunterladen müssen.

Nach dem Patch müssen Sie in der Kernel-Konfiguration neben den üblichen Netfilter-Optionen auch die neue Option „Layer 7 match support“ einschalten. Nach dem Kompilieren und Installieren Ihres Kernels steht die Funktion im Kernel zur Verfügung. Nun müssen Sie noch IPtables patchen und installieren. Hierzu ist im L7-Filter-Quelltextverzeichnis ebenfalls ein Patch vorhanden.

Nun müssen Sie noch die Protokolldefinitionen entpacken und in dem Verzeichnis `l7-protocols` unterhalb von `/etc/` installieren. In dem Protokoll-Archiv von der Homepage sind viele Protokolle enthalten. Wählen Sie die Protokolle, die Sie verwenden möchten. Andere Orte verlangen in den späteren Regeln die Angabe des Verzeichnisses mit `--l7dir`.

Die allgemeine Verwendung des Netfilter-Matches `layer7` ist:

```
iptables -A FORWARD -m layer7 --l7proto edonkey -j DROP
```

In zwei Fällen kann L7-Filter das Protokoll nicht erkennen:

» Es sind noch nicht genug Pakete analysiert worden. L7-Filter versucht noch, das Protokoll zu bestimmen. Diese Verbindungen werden von L7-Filter als `unset` bezeichnet. Sie können diese Verbindungen mit `--l7proto unset` testen.

» L7-Filter hat bereits aufgegeben, das Protokoll zu erkennen, da es keine passenden regulären Ausdrücke besitzt. Diese Verbindungen werden von L7-Filter als `unknown` bezeichnet. L7-Filter gibt nach 10 Paketen bzw. 2 Kbyte Daten auf. Möchten Sie, dass L7-Filter mehr Pakete analysiert, so können Sie diesen Wert ändern: `/prc/net/layer7_numpackets`.

33.3 L7: Userspace-Version

Statt der Kernelspace-Version ist es häufig sinnvoller, die Userspace-Version zu nutzen. Diese ist auch Bestandteil zum Beispiel der Debian-Squeeze-Distribution.[1] Wenn Sie diese Version manuell installieren möchten, so genügt es, nach dem Herunterladen und Auspacken des Quelltextarchives die folgenden drei Befehle aufzurufen:

```
./configure
make
sudo make install
```

Als Voraussetzung für die erfolgreiche Übersetzung müssen Sie die Bibliotheken `libnetfilter_conntrack` und `libnetfilter_queue` installieren.

Zusätzlich benötigen Sie wie bei der Kernel-Version die Protokolldefinitionen, die Sie ebenfalls in dem Verzeichnis `/etc/l7-protocols` installieren.

33.3.1 Nutzung der Userspace-Version

Wenn Sie die Userspace-Version nutzen, erfolgt die Klassifizierung der Pakete nicht im Kernel, sondern durch ein Userspace-Programm. Damit das Programm die Pakete erhält, müssen Sie es über eine Netfilter-Queue anbinden. Hierzu müssen Sie alle Pakete, die analysiert werden sollen, entsprechend mit `iptables` an das `NFQUEUE` Target senden:

```
iptables -t mangle -A FORWARD -j NFQUEUE
```

Der Aufruf des Userspace-Programms erfolgt dann mit:

```
l7-filter -f /etc/l7-filter.conf -q 0 -p /etc/l7-protocols/
```

1 l7-filter-userspace

Die Optionen `-q` und `-p` sind optional und geben die Queue-Nummer und den Ort der Protokolldefinitionen an.

Das Userspace-Programm `l7-filter` bindet sich an die Netfilter-Queue und analysiert die Pakete. Wichtig ist, dass dieses Programm die Pakete in beiden Richtungen erhält, da nur so häufig eine Erkennung erfolgreich ist. Die Pakete werden dann von dem Programm entsprechend dem erkannten Protokoll markiert und akzeptiert. Die Markierung können Sie dann wieder mit `iptables`-Regeln auswerten. Um den einzelnen Protokollen Markierungswerte zuzuweisen, nutzen Sie eine Konfigurationsdatei `l7-filter.conf`:

```
# The format of this file is:
# protocol mark

# Do not use marks less than 3, since a mark of 0 means that l7-filter ↵
        hasn't
# seen the packet yet, and a mark of 1 means that it has failed to ↵
        classify
# it, but will try again with the next packet of the connection and a ↵
        mark
# of 2 means that l7-filter has given up trying to match.

gnutella        3
imap            4
aim             5
smtp            6
dns             7
validcertssl    8
...
```

Hierbei dürfen Sie nur Markierungen ab 3 vergeben, da die kleineren Werte für interne Zwecke von `l7-filter` verwendet werden:

1: Die Erkennung der Pakete dieser Verbindung erfolgt noch.

2: Die Erkennung der Pakete dieser Verbindung wurde aufgegeben.

Wichtig ist die Erkenntnis, dass der Userspace `l7-filter` alle Pakete akzeptiert. Die entsprechende Regel zur Weiterleitung an die Netfilter-Queue muss daher die letzte Regel der entsprechenden Kette sein. Die Prüfung der Markierung muss in einer anderen Kette, die anschließend durchlaufen wird, erfolgen. Sinnvoll kann daher der `NFQUEUE`-Aufruf in der Mangle-`FORWARD`-Kette CE: erfolgen, während sich die Prüfung der Markierung in der Filter-`FORWARD`-Kette anschließt.

STOP

Trägt das Paket bereits eine Markierung, wenn es an `l7-filter` geschickt wird, wird diese nicht verändert. L7-Filter wird das Paket nicht analysieren.

CE: Bitte Ergänzung bestätigen.

Editor's or typesetter's annotations (will be removed before the final TEX run)

33.4 Eigene L7-Protokolle

Sie können die Protokolldefinitionen um eigene Protokolle erweitern. Hierzu müssen Sie lediglich eine Protokollbeschreibung erzeugen. Diese besteht aus einer einzelnen Datei für jedes zu beschreibene Protokoll. Die Datei enthält zwei Zeilen. In der ersten Zeile benennen Sie das Protokoll. Die zweite Zeile enthält einen regulären Ausdruck, der für die Erkennung genutzt werden soll:

```
dns
# here's a sane way of doing it
^.?.?.?.?[\x01\x02].?.?.?.?.?.?[\x01-?][a-z0-9][\x01-?a-z]*[\x02-\x06][a- ↵
        z][a-z][fglmoprstuvz]?[aeop]?(um)?[\x01-\x10\x1c][\x01\x03\x04\ ↵
        xFF]
```

Zeilen, die mit einem Kommentarzeichen beginnen, werden ignoriert. Die Datei muss dann unter dem Namen des Protokolls mit der Endung `.pat` unterhalb der Verzeichnisstruktur `/etc/l7-protocols` gespeichert werden.

34. IPsec

Die Filterung von IPsec-Verbindungen ist bei jeder Firewall eine besondere Aufgabe. Unter Linux wird dies zusätzlich durch fehlende virtuelle Netzwerkkarten ab dem Kernel 2.6 kompliziert.

34.1 IPsec-Grundlagen

Die IPsec-Protokolle werden verwendet, um Informationen in geschützter Form über das Internet zu transportieren. Sie schützen die Authentizität, die Integrität und die Vertraulichkeit der Pakete.

Dies wird mit drei verschiedenen Protokollen realisiert:

Authentication Header (AH, IP-Protokoll 51): Dieses Protokoll kann die Authentizität und Integrität von Paketen garantieren. Dabei schließt die Garantie auch die unveränderlichen Bestandteile des äußeren IP-Headers mit ein. Eine nachträgliche Änderung dieser Bestandteile (z. B. NAT) wird von dem AH-Protokoll als Fehler erkannt, und das Paket wird verworfen.

Encapsulated Security Payload (ESP, IP-Protokoll 50): Dieses Protokoll kann die Authentizität, Integrität und Vertraulichkeit von Paketen garantieren. Es schließt nicht den äußeren IP-Header in seine Überprüfung mit ein. Ein NAT erzeugt also kein ungültiges Paket. Daher wird heute häufig nur noch dieses Protokoll eingesetzt.

Internet Key Exchange (IKE, 500/udp): Sowohl AH als auch ESP benötigen Schlüsselmaterial für die Realisierung ihrer Aufgaben. Das IKE-Protokoll authentifiziert die Kommunikationspartner und erzeugt dieses Schlüsselmaterial.

Da fast alle Router mit dem NAT von portlosen Protokollen Schwierigkeiten bekommen (siehe Abschnitt 20.2), wurden noch Erweiterungen für diese Protokolle vorgesehen, die heute von fast allen Implementierungen unterstützt werden. Sobald das IKE-Protokoll während der Verhandlung der Verbindung ein NAT erkennt, wechselt es den Port von 500/udp auf den UDP-Port 4500. Sobald die ESP-Verbindung ausgehandelt wurde, werden auch die ESP-Pakete erneut in UDP-Pakete mit dem Port 4500 gekapselt. Damit weisen die Pakete für einen NAT-Router unterwegs wieder einen Port auf, der als Kriterium für die Unterscheidung der genatteten Verbindungen genutzt werden kann.

Es gibt für den Kernel 2.6 praktisch nur noch eine IPsec-Implementierung:

1. 26sec: Dieser Name hat sich für den neuen IPsec-Stack des Linux-Kernels 2.6 eingebürgert. Dieser Stack ist ohne Patch verfügbar. Er besitzt keine virtuellen Netzwerkkarten.

2. KLIPS: KLIPS ist als Patch für den Kernel 2.4 in FreeS/wan, Openswan und strongSwan enthalten. Für den Kernel 2.6 gibt es im Moment nur einen Patch in der Openswan-Distribution. Diese Implementierung stellt virtuelle Netzwerkkarten mit dem Namen `ipsecX` zur Verfügung. Der Verkehr auf diesen Netzwerkkarten findet im Klartext statt. Alle über diese Netzwerkkarten transportierten Pakete werden im Weiteren verschlüsselt über die physikalischen Netzwerkkarten gesendet oder empfangen. Diese IPsec-Implementierung wird auf modernen Distributionen jedoch nicht genutzt.

34.2 Iptables-Regeln zum Durchleiten von IPsec

Wenn Sie einem Client die Möglichkeit geben möchten, durch eine Firewall auf ein VPN-Gateway zuzugreifen, können Sie die folgenden Regeln verwenden:

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p udp -m multiport --dport ↵
        500,4500 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p esp -m state --state NEW -j ↵
         ACCEPT
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p ah -m state --state NEW -j ↵
        ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Falls es bei der Verwendung der Protokollnamen `esp` und `ah` zu Fehlern kommt, können Sie stattdessen auch die numerischen Werte 50 und 51 nutzen.

Die Connection-Tracking-Timeout-Werte sind möglicherweise zu klein für das VPN. Daher sollten Sie diese anpassen und für UDP und generische Protokolle erhöhen (siehe Abschnitt 5.5).

Wenn Sie ein VPN-Gateway hinter einer Firewall in einer DMZ betreiben möchten, müssen Sie die Protokolle von der Firewall an das Gateway weiterleiten. Dies können Sie mit den folgenden Befehlen erreichen:

```
VPN_GW=192.168.0.5

$IPTABLES -t nat -A PREROUTING -p udp -m multiport --dport 500,4500 -j ↵
        DNAT --to-destination $VPN_GW
$IPTABLES -t nat -A PREROUTING -p esp -j DNAT --to-destination $VPN_GW

$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -d $VPN_GW -p udp -m multiport ↵
         --dport 500,4500 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -d $VPN_GW -p esp -m state -- ↵
        state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Das Protokoll AH (51) kommt in den Regeln nicht vor, da es kein NAT erlaubt.

34.3 IPsec und Firewall auf demselben System

Häufig werden VPN- und Firewallfunktionen auf demselben System zusammengefasst. Dann müssen Sie mit Ihrer Firewall zwischen den Paketen, die im Klartext mit dem Internet ausgetauscht werden, und den Paketen, die über das VPN transportiert werden, unterscheiden. Meist soll dem Zugriff über das VPN mehr Privilegien eingeräumt werden.

Bei der Verwendung des IPsec-Stacks, der in dem Linux-Kernel 2.6 enthalten ist, ist die Filterung der relevanten Pakete nicht einfach. Zunächst sollten Sie sich selbst Klarheit darüber verschaffen, wie Ihr Kernel die Pakete in die verschiedenen Ketten einsortiert. Dies ist nämlich abhängig von möglichen Patches, die Ihre Distribution bereits in den Kernel eingearbeitet hat. Leider ist es nicht möglich, dies allgemeingültig für alle Distributionen und alle Kernel anzugeben. Auch kann ich nicht garantieren, dass Sie die Informationen, die Sie zum Beispiel für einen SUSE-Professional-9.2-Kernel ermitteln, unverändert auf einen OpenSUSE-11.3-Kernel übertragen können. In der Vergangenheit existierten sogar Unterschiede bei den verschiedenen Kerneln innerhalb einer Distributionsversion. Mit viel Glück wurden diese Modifikationen aber in dem Changelog des RPM- oder Debian-Pakets hinterlegt. Sie können das Changelog eines RPM-Pakets mit dem folgenden Befehl auslesen:

```
rpm -q --changelog kernel-<version>.<arch>.rpm
```

Um sich einen Überblick über die Einsortierung der Pakete in den Ketten zu verschaffen, sollten Sie sich eine Testumgebung wie in Abbildung 34.1 aufbauen. Dann fügen Sie zu allen Ketten in allen Tabellen LOG-Regeln hinzu und erzeugen IPsec-Verkehr. Hierzu können Sie das folgende Skript verwenden:

```
iptables -t raw -A PREROUTING -j LOG --log-prefix "RAW:PREROUTING"
iptables -t raw -A OUTPUT -j LOG --log-prefix "RAW:OUTPUT"

iptables -t mangle -A PREROUTING -j LOG --log-prefix "MANGLE:PREROUTING"
iptables -t mangle -A INPUT -j LOG --log-prefix "MANGLE:INPUT"
iptables -t mangle -A FORWARD -j LOG --log-prefix "MANGLE:FORWARD"
iptables -t mangle -A OUTPUT -j LOG --log-prefix "MANGLE:OUTPUT"
iptables -t mangle -A POSTROUTING -j LOG --log-prefix"MANGLE:POSTROUTING"

iptables -t nat -A PREROUTING -j LOG --log-prefix "NAT:PREROUTING"
iptables -t nat -A OUTPUT -j LOG --log-prefix "NAT:OUTPUT"
iptables -t nat -A POSTROUTING -j LOG --log-prefix "NAT:POSTROUTING"

iptables -t filter -A INPUT -j LOG --log-prefix "FILTER:INPUT"
iptables -t filter -A FORWARD -j LOG --log-prefix "FILTER:FORWARD"
iptables -t filter -A OUTPUT -j LOG --log-prefix "FILTER:OUTPUT"
```

Achten Sie darauf, dass es ein Unterschied ist, ob Sie die Pakete auf dem VPN-Gateway selbst oder von einem Client hinter dem Gateway erzeugen lassen. Außerdem ist es ein Unterschied,

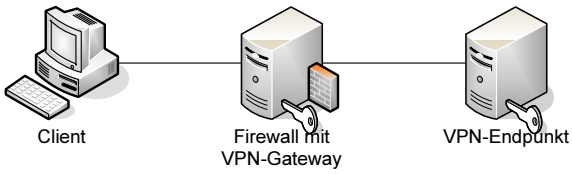


Abbildung 34.1: **Bauen Sie sich eine Testumgebung, in der Sie Ihren Kernel testen können.**

ob eine IPsec-Tunnel-Security-Association oder eine IPsec-Transport-Security-Association genutzt wird. Versuchen Sie möglichst Ihre Konstellation nachzubilden.

Im Folgenden wird das Verhalten des Kernels 2.6.26 beschrieben.

34.3.1 Transport-Modus

Im Transport-Modus ist die Betrachtung besonders einfach. Da die Pakete direkt als IPsec-Pakete erzeugt werden, kann Iptables nur die verschlüsselten Pakete filtern. Diese treten ganz normal in der INPUT- und in der OUTPUT-Kette auf.

Der Inhalt dieser Pakete kann vor ihrer Verschlüsselung nicht gefiltert werden.

34.3.2 Tunnel-Modus

Anders ist das bei Paketen, die in dem Tunnel-Modus übertragen werden.

Hier müssen Sie zwischen Paketen unterscheiden, die von dem Tunnel-Gateway selbst erzeugt werden, und Paketen, die von anderen Systemen erzeugt werden, die den Tunnel nutzen.

In Abbildung 34.2 sehen Sie den Paketverlauf durch die Ketten, wenn ein anderer Rechner durch das Tunnel-Gateway auf den IPsec-Tunnel zugreift. Links ist der Paketverlauf des ausgehenden Pakets und rechts der Paketverlauf des eingehenden Pakets gezeichnet.

In den Protokollen zeigt sich das folgendermaßen. Zunächst betrachten wir das ausgehende Paket. Hier handelt es sich um ein Ping (Echo-Request).

```
Apr  6 18:55:06 berlin kernel: [ 9224.598162] RAW:PREROUTINGIN=eth0 OUT=  ↵
        MAC=00:e0:81:80:c5:60:00:e0:81:80:c5:6f:08:00 SRC=10.0.2.100 DST ↵
        =10.0.1.100 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF PROTO=ICMP  ↵
        TYPE=8 CODE=0 ID=17160 SEQ=1
Apr  6 18:55:06 berlin kernel: [ 9224.598162] MANGLE:PREROUTINGIN=eth0  ↵
        OUT= MAC=00:e0:81:80:c5:60:00:e0:81:80:c5:6f:08:00 SRC ↵
        =10.0.2.100 DST=10.0.1.100 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=0 ↵
         DF PROTO=ICMP TYPE=8 CODE=0 ID=17160 SEQ=1
Apr  6 18:55:06 berlin kernel: [ 9224.598162] NAT:PREROUTINGIN=eth0 OUT=  ↵
        MAC=00:e0:81:80:c5:60:00:e0:81:80:c5:6f:08:00 SRC=10.0.2.100 DST ↵
        =10.0.1.100 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF PROTO=ICMP  ↵
        TYPE=8 CODE=0 ID=17160 SEQ=1
```

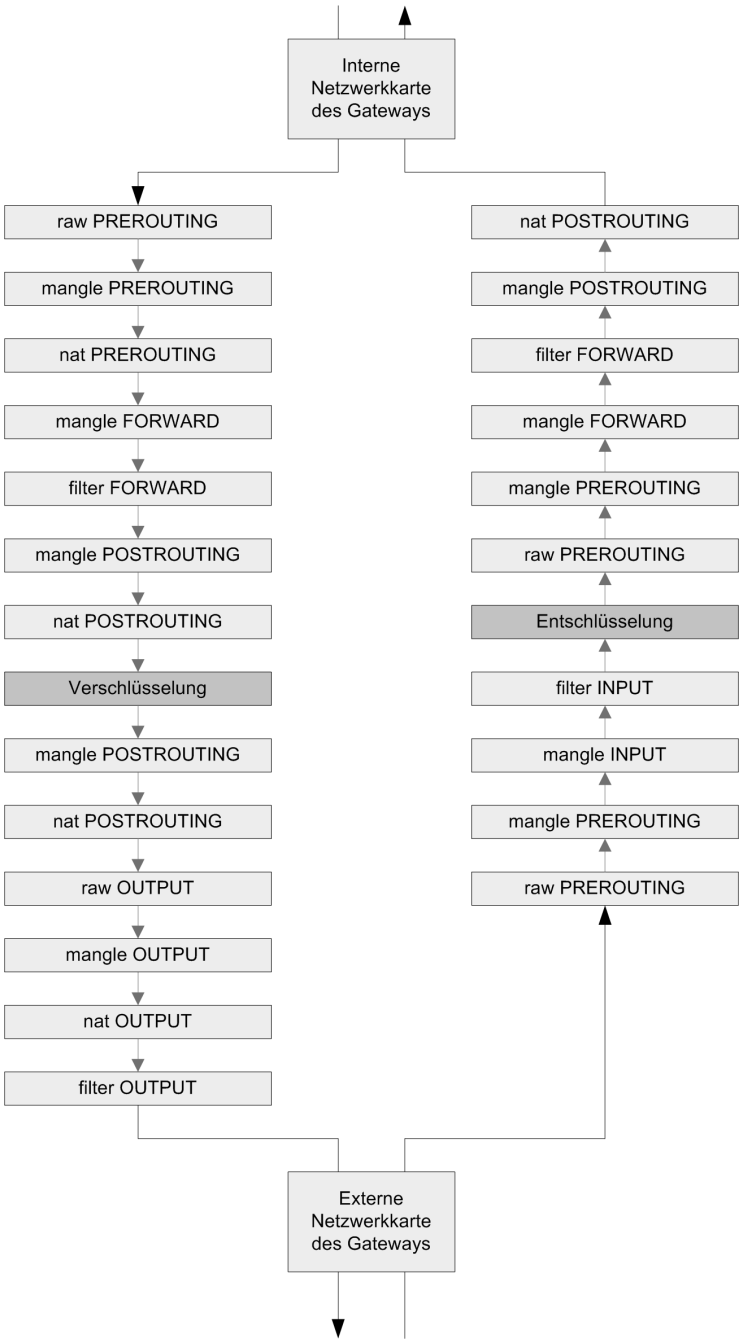


Abbildung 34.2: **Paketverlauf auf einem Gateway durch die Ketten**

```
Apr  6 18:55:06 berlin kernel: [ 9224.598162] MANGLE:FORWARDIN=eth0 OUT= ↵
        eth1 SRC=10.0.2.100 DST=10.0.1.100 LEN=84 TOS=0x00 PREC=0x00 TTL ↵
        =63 ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=17160 SEQ=1
Apr  6 18:55:06 berlin kernel: [ 9224.598162] FILTER:FORWARDIN=eth0 OUT= ↵
        eth1 SRC=10.0.2.100 DST=10.0.1.100 LEN=84 TOS=0x00 PREC=0x00 TTL ↵
        =63 ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=17160 SEQ=1
Apr  6 18:55:06 berlin kernel: [ 9224.598162] MANGLE:POSTROUTINGIN= OUT= ↵
        eth1 SRC=10.0.2.100 DST=10.0.1.100 LEN=84 TOS=0x00 PREC=0x00 TTL ↵
        =63 ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=17160 SEQ=1
Apr  6 18:55:06 berlin kernel: [ 9224.598162] NAT:POSTROUTINGIN= OUT=eth1 ↵
         SRC=10.0.2.100 DST=10.0.1.100 LEN=84 TOS=0x00 PREC=0x00 TTL=63  ↵
        ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=17160 SEQ=1
Apr  6 18:55:06 berlin kernel: [ 9224.598162] RAW:OUTPUTIN= OUT=eth1 SRC ↵
        =5.0.0.1 DST=3.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF  ↵
        PROTO=ESP SPI=0xa2fbcf20
Apr  6 18:55:06 berlin kernel: [ 9224.598162] MANGLE:OUTPUTIN= OUT=eth1  ↵
        SRC=5.0.0.1 DST=3.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=64 ID=0  ↵
        DF PROTO=ESP SPI=0xa2fbcf20
Apr  6 18:55:06 berlin kernel: [ 9224.598162] NAT:OUTPUTIN= OUT=eth1 SRC ↵
        =5.0.0.1 DST=3.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF  ↵
        PROTO=ESP SPI=0xa2fbcf20
Apr  6 18:55:06 berlin kernel: [ 9224.598162] FILTER:OUTPUTIN= OUT=eth1  ↵
        SRC=5.0.0.1 DST=3.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=64 ID=0  ↵
        DF PROTO=ESP SPI=0xa2fbcf20
Apr  6 18:55:06 berlin kernel: [ 9224.598162] MANGLE:POSTROUTINGIN= OUT= ↵
        eth1 SRC=5.0.0.1 DST=3.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=64  ↵
        ID=0 DF PROTO=ESP SPI=0xa2fbcf20
Apr  6 18:55:06 berlin kernel: [ 9224.598162] NAT:POSTROUTINGIN= OUT=eth1 ↵
         SRC=5.0.0.1 DST=3.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=64 ID=0  ↵
        DF PROTO=ESP SPI=0xa2fbcf20
```

Und nun die Antwort, der Pong (Echo-Response):

```
Apr  6 18:55:06 berlin kernel: [ 9224.598162] RAW:PREROUTINGIN=eth1 OUT=  ↵
        MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 DST ↵
        =5.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=63 ID=57885 PROTO=ESP  ↵
        SPI=0x3ddcef50
Apr  6 18:55:06 berlin kernel: [ 9224.598162] MANGLE:PREROUTINGIN=eth1  ↵
        OUT= MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1  ↵
        DST=5.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=63 ID=57885 PROTO=ESP ↵
         SPI=0x3ddcef50
Apr  6 18:55:06 berlin kernel: [ 9224.598162] MANGLE:INPUTIN=eth1 OUT=  ↵
        MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 DST ↵
```

```
        =5.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=63 ID=57885 PROTO=ESP ↵
        SPI=0x3ddcef50
Apr  6 18:55:06 berlin kernel: [ 9224.598162] FILTER:INPUTIN=eth1 OUT= ↵
        MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 DST ↵
        =5.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=63 ID=57885 PROTO=ESP ↵
        SPI=0x3ddcef50
Apr  6 18:55:06 berlin kernel: [ 9224.598162] RAW:PREROUTINGIN=eth1 OUT= ↵
        MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=10.0.1.100 DST ↵
        =10.0.2.100 LEN=84 TOS=0x00 PREC=0x00 TTL=63 ID=20746 PROTO=ICMP ↵
         TYPE=0 CODE=0 ID=17160 SEQ=1
Apr  6 18:55:06 berlin kernel: [ 9224.598162] MANGLE:PREROUTINGIN=eth1 ↵
        OUT= MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC ↵
        =10.0.1.100 DST=10.0.2.100 LEN=84 TOS=0x00 PREC=0x00 TTL=63 ID ↵
        =20746 PROTO=ICMP TYPE=0 CODE=0 ID=17160 SEQ=1
Apr  6 18:55:06 berlin kernel: [ 9224.598162] MANGLE:FORWARDIN=eth1 OUT= ↵
        eth0 SRC=10.0.1.100 DST=10.0.2.100 LEN=84 TOS=0x00 PREC=0x00 TTL ↵
        =62 ID=20746 PROTO=ICMP TYPE=0 CODE=0 ID=17160 SEQ=1
Apr  6 18:55:06 berlin kernel: [ 9224.598162] FILTER:FORWARDIN=eth1 OUT= ↵
        eth0 SRC=10.0.1.100 DST=10.0.2.100 LEN=84 TOS=0x00 PREC=0x00 TTL ↵
        =62 ID=20746 PROTO=ICMP TYPE=0 CODE=0 ID=17160 SEQ=1
Apr  6 18:55:06 berlin kernel: [ 9224.598162] MANGLE:POSTROUTINGIN= OUT= ↵
        eth0 SRC=10.0.1.100 DST=10.0.2.100 LEN=84 TOS=0x00 PREC=0x00 TTL ↵
        =62 ID=20746 PROTO=ICMP TYPE=0 CODE=0 ID=17160 SEQ=1
```

In Abbildung 34.3CE^W sehen Sie den Verlauf, wenn das Gateway selbst durch den Tunnel kommuniziert.

Sie sollten erkennen, dass die Filter-Ketten das ausgehende Paket zweimal analysieren. Zunächst wird das Klartextpaket, dann das ausgehende verschlüsselte Paket von den Filter-Ketten betrachtet. Das eingehende IPsec-Paket wird zunächst im verschlüsselten Zustand in der Filter-INPUT-Kette gefiltert, dann entschlüsselt und durchläuft dann erneut sämtliche Ketten, als ob das Paket gerade erst angekommen sei.

In den Logs zeigt sich dies bei einem Ping folgendermaßen. Betrachten Sie zunächst das ausgehende Paket:

```
Apr  6 18:27:15 berlin kernel: [ 7553.881407] RAW:OUTPUTIN= OUT=eth1 SRC ↵
        =5.0.0.1 DST=3.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF ↵
        PROTO=ICMP TYPE=8 CODE=0 ID=64777 SEQ=1
Apr  6 18:27:15 berlin kernel: [ 7553.881427] MANGLE:OUTPUTIN= OUT=eth1 ↵
        SRC=5.0.0.1 DST=3.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF ↵
         PROTO=ICMP TYPE=8 CODE=0 ID=64777 SEQ=1
Apr  6 18:27:15 berlin kernel: [ 7553.881438] NAT:OUTPUTIN= OUT=eth1 SRC ↵
        =5.0.0.1 DST=3.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF ↵
        PROTO=ICMP TYPE=8 CODE=0 ID=64777 SEQ=1
```


CE^W Bitte diesen Verweis bestätigen.

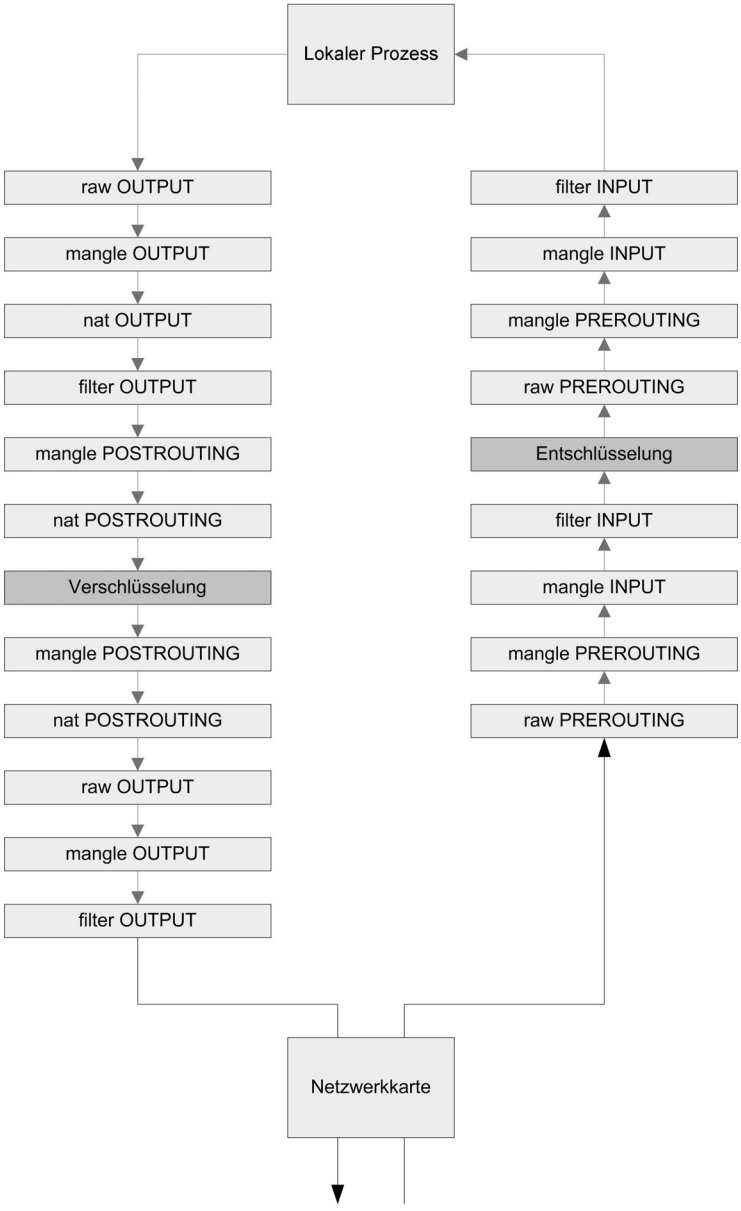


Abbildung 34.3: **Paketverlauf auf einem Host durch die Ketten**

```
Apr  6 18:27:15 berlin kernel: [ 7553.881451] FILTER:OUTPUTIN= OUT=eth1 ↵
        SRC=5.0.0.1 DST=3.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF ↵
         PROTO=ICMP TYPE=8 CODE=0 ID=64777 SEQ=1
Apr  6 18:27:15 berlin kernel: [ 7553.881461] MANGLE:POSTROUTINGIN= OUT= ↵
        eth1 SRC=5.0.0.1 DST=3.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID ↵
        =0 DF PROTO=ICMP TYPE=8 CODE=0 ID=64777 SEQ=1
Apr  6 18:27:15 berlin kernel: [ 7553.881470] NAT:POSTROUTINGIN= OUT=eth1 ↵
         SRC=5.0.0.1 DST=3.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=0 ↵
        DF PROTO=ICMP TYPE=8 CODE=0 ID=64777 SEQ=1
Apr  6 18:27:15 berlin kernel: [ 7553.881510] RAW:OUTPUTIN= OUT=eth1 SRC ↵
        =5.0.0.1 DST=3.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF ↵
        PROTO=ESP SPI=0xad2a510f
Apr  6 18:27:15 berlin kernel: [ 7553.881521] MANGLE:OUTPUTIN= OUT=eth1 ↵
        SRC=5.0.0.1 DST=3.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=64 ID=0 ↵
        DF PROTO=ESP SPI=0xad2a510f
Apr  6 18:27:15 berlin kernel: [ 7553.881530] FILTER:OUTPUTIN= OUT=eth1 ↵
        SRC=5.0.0.1 DST=3.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=64 ID=0 ↵
        DF PROTO=ESP SPI=0xad2a510f
Apr  6 18:27:15 berlin kernel: [ 7553.881538] MANGLE:POSTROUTINGIN= OUT= ↵
        eth1 SRC=5.0.0.1 DST=3.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=64 ↵
        ID=0 DF PROTO=ESP SPI=0xad2a510f
```

Und nun die Antwort auf das Ping (der „Pong“):

```
Apr  6 18:27:15 berlin kernel: [ 7553.882304] RAW:PREROUTINGIN=eth1 OUT= ↵
        MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 DST ↵
        =5.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=63 ID=57884 PROTO=ESP ↵
        SPI=0x68ba6e65
Apr  6 18:27:15 berlin kernel: [ 7553.882321] MANGLE:PREROUTINGIN=eth1 ↵
        OUT= MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 ↵
        DST=5.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=63 ID=57884 PROTO=ESP ↵
         SPI=0x68ba6e65
Apr  6 18:27:15 berlin kernel: [ 7553.882337] MANGLE:INPUTIN=eth1 OUT= ↵
        MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 DST ↵
        =5.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=63 ID=57884 PROTO=ESP ↵
        SPI=0x68ba6e65
Apr  6 18:27:15 berlin kernel: [ 7553.882351] FILTER:INPUTIN=eth1 OUT= ↵
        MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 DST ↵
        =5.0.0.1 LEN=152 TOS=0x00 PREC=0x00 TTL=63 ID=57884 PROTO=ESP ↵
        SPI=0x68ba6e65
Apr  6 18:27:15 berlin kernel: [ 7553.882404] RAW:PREROUTINGIN=eth1 OUT= ↵
        MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 DST ↵
        =5.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=57883 PROTO=ICMP ↵
        TYPE=0 CODE=0 ID=64777 SEQ=1
```

```
Apr  6 18:27:15 berlin kernel: [ 7553.882422] MANGLE:PREROUTINGIN=eth1 ↵
        OUT= MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 ↵
        DST=5.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=57883 PROTO=ICMP ↵
         TYPE=0 CODE=0 ID=64777 SEQ=1
Apr  6 18:27:15 berlin kernel: [ 7553.882437] MANGLE:INPUTIN=eth1 OUT= ↵
        MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 DST ↵
        =5.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=57883 PROTO=ICMP ↵
        TYPE=0 CODE=0 ID=64777 SEQ=1
Apr  6 18:27:15 berlin kernel: [ 7553.882452] FILTER:INPUTIN=eth1 OUT= ↵
        MAC=00:e0:81:80:c5:61:1e:0b:ef:1e:1a:16:08:00 SRC=3.0.0.1 DST ↵
        =5.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=57883 PROTO=ICMP ↵
        TYPE=0 CODE=0 ID=64777 SEQ=1
```

Die `nat-POSTROUTING`-Kette wird von dem ausgehenden Paket als Klartextpaket durchlaufen. Dadurch ist ein Source-NAT des Pakets vor der Verschlüsselung möglich. Dies ist auf älteren Kerneln nicht der Fall!

34.4 Filterung mit dem policy-Match

Vor der Einführung des Policy-Matches wurde die Filterung der Pakete mithilfe der Firewall-Markierung durchgeführt. Besser ist es jedoch, mit dem `policy`-Match zu arbeiten. Dieser Match ist in allen modernen Kerneln enthalten. Hiermit können Sie ohne umständliche Markierung die Pakete erkennen, die von einer IPsec-Policy geschützt werden. Zusätzlich können Sie sogar den Tunnel prüfen, der von dem Paket genutzt wird. Der Test hat die folgenden Optionen, um auf das Vorhandensein einer Policy zu testen:

- `--dir in|out`: Hiermit definieren Sie, ob Sie eine `in`- oder `out`-Policy testen möchten. Diese Option müssen Sie angeben.
- `--pol none|ipsec`: Hiermit wählen Sie aus, ob die Policy einen Schutz mit IPsec verlangen soll oder nicht.
- `--strict`: Wenn Sie diese Option angeben, müssen Sie die Policy genau beschreiben.
- `--reqid <id>`: Hiermit können Sie die genaue ID der Policy angeben. Diese ID kann mit dem `setkey`-Kommando mit der Angabe `unique:id` bei der Definition der Policy gesetzt werden.
- `--spi`: Hiermit können Sie die genaue SPI der Security Association auswählen, die genutzt werden muss.
- `--proto ah|esp|ipcomp`: Hiermit fragen Sie das Protokoll der Policy ab.
- `--mode tunnel|transport`: Dies wählt zwischen den IPsec-Modi Tunnel und Transport.
- `--tunnel-src <ip>[/<maske>]`: Bei einem Tunnel können Sie die spezifischen Tunnelendpunkte angeben. Dies definiert die Quelladresse.
- `--tunnel-dst <ip>[/<maske>]`: Dies definiert die Zieladresse.

» `--next`: Bei der Verwendung von `--strict` können Sie hiermit das nächste Element definieren.

Um nun in der FORWARD-Kette nur den Verkehr zuzulassen, der später von dem VPN geschützt wird oder über das VPN den Rechner erreicht hat, können Sie folgende Regeln verwenden:

```
$IPTABLES -P FORWARD DROP
$IPTABLES -A FORWARD -m policy --dir in --mode tunnel --pol ipsec --proto ↵
          esp -j ACCEPT
$IPTABLES -A FORWARD -m policy --dir out --mode tunnel --pol ipsec -- ↵
         proto esp -j ACCEPT
```

Die erste Zeile setzt die Default-Policy der FORWARD-Kette auf DROP. Alle Pakete, die nicht explizit akzeptiert werden, werden verworfen. Die zweite Zeile prüft, ob das Paket von einer IPsec-Policy in der Richtung `in` im Tunnel-Modus mit dem Protokoll ESP geschützt wird, und akzeptiert diese Pakete. Die letzte Zeile prüft, ob das Paket von einer entsprechenden Policy in der Richtung `out` geschützt wird. Damit haben Sie beide Richtungen abgedeckt und stellen sicher, dass die in der FORWARD-Kette akzeptierten Pakete im Internet immer von einer IPsec-Policy geschützt sind.

35. ICMP

Ohne ICMP würde das Internet nicht funktionieren. Häufig ist daher eine fehlerhafte Konfiguration dieses Protokolls auch für Funktionsstörungen verantwortlich. Meist hängt dies mit falsch konfigurierten Firewalls zusammen. Dieses Kapitel zeigt Ihnen, wie Sie das Protokoll für IPv4 richtig filtern. Speziell für IPv6 wird das entsprechende Protokoll ICMPv6 in einem eigenen Kapitel besprochen (siehe Kapitel 36).

35.1 ICMP-Grundlagen

Die Filterung des ICMP-Protokolls (RFC792) ist ein recht kompliziertes und umfangreiches Thema, da
es viele wichtige ICMP-Nachrichten gibt. Das Internet Control and Message Protocol ist in erster Linie für die Übertragung von Fehlermeldungen verantwortlich. Außerdem wird dieses Protokoll von dem `ping`-Befehl und dem `traceroute`-Befehl eingesetzt.

Erfreulicherweise unterstützt die Stateful-Inspection des Linux-Kernels auch das ICMP-Protokoll. Das bedeutet, dass der Kernel erkennen kann, ob eine ICMP-Fehlermeldung sich auf eine existente Verbindung bezieht oder nicht. So ist es möglich, ICMP-Fehlermeldungen nur zuzulassen, wenn sie sich tatsächlich auf eine aufgebaute Verbindung beziehen, und jede andere Form direkt zu verwerfen. Die Fehlermeldungen, die sich auf existente Verbindungen beziehen, werden von dem Linux-Kernel als `RELATED` eingeordnet. Wenn Sie `RELATED`-Pakete zulassen, dann werden auch diese Pakete akzeptiert. Dabei löscht dann der Linux-Kernel auch die Verbindung, auf die sich die Fehlermeldung bezog, aus der Verbindungsliste, denn es trat ja ein Fehler auf.

In den meisten Fällen genügt es daher, wenn Sie in Ihrer Firewall die `RELATED`-Pakete akzeptieren und alle weiteren ICMP-Nachrichten einfach ignorieren.

Dieses grundsätzliche Filtern der Meldungen ist sehr einfach. Wenn Sie einfach alle gültigen ICMP-Meldungen akzeptieren möchten, nutzen Sie folgende Regel:

```
$IPTABLES -A FORWARD -p icmp -m state --state RELATED -j ACCEPT
```

Denken Sie daran, dass Sie vielleicht sowieso schon eine Regel in Ihrem Skript haben, die alle `RELATED`-Pakete erlaubt!

Eine derart allgemeine Betrachtung ist jedoch in einigen Fällen nicht ausreichend. In Abhängigkeit von den Firewall-Richtlinien müssen die ICMP-Nachrichten einzeln betrachtet werden. Das wird im Folgenden gezeigt. Dazu betrachten wir zunächst die einzelnen Nachrichten, um uns dann anschließend einzelne Befehle näher anzusehen, die diese Funktionen nutzen (`ping` und `traceroute`).

Im Einzelnen betrachten wir die folgenden ICMP-Nachrichten:

- *destination-unreachable*
- *destination-unreachable*: *fragmentation-needed*
- *source-quench*
- *redirect*
- *router-advertisement*
- *router-solicitation*
- *time-exceeded*
- *parameter-problem*
- *timestamp-request/timestamp-reply*
- *address-mask-request/address-mask-reply*
- *echo-request/echo-reply*

Wenn Sie nur bestimmte Meldungen akzeptieren möchten, können Sie zum Beispiel die folgende Regel verwenden:

```
$IPTABLES -A FORWARD -p icmp --icmp-type destination-unreachable -m state ↵
          --state RELATED -j ACCEPT
```

Was nun wo und in welcher Richtung Sinn macht, erläutern die nächsten Abschnitte.

35.2 *destination-unreachable*

Die ICMP-Fehlermeldung *destination-unreachable* wird immer dann gesendet, wenn ein Ziel nicht erreichbar ist. Dabei kann es mehrere Gründe für diese Meldung geben. Im Folgenden sehen Sie eine Auswahl der häufigsten Gründe:

- Der angesprochene UDP-Port wird von keiner Applikation bedient. Der Host sendet dann die Fehlermeldung.
- Der angesprochene Rechner kennt das IP-Protokoll nicht. Der Host sendet dann die Fehlermeldung.
- Der angesprochene Rechner reagiert nicht auf eine ARP-Anfrage. Er ist wahrscheinlich ausgeschaltet. Erfolgte die Verbindung über einen Router, so sendet dieser dann die Fehlermeldung.
- Das Netzwerk, in dem sich der Zielrechner befindet, ist nicht erreichbar. Ein Router sendet dann die Fehlermeldung.
- Der Rechner oder das Netzwerk werden durch eine Firewall geschützt. Einige Firewalls senden dann ein *destination-unreachable*: *network-prohibited*.
- Das Paket ist zu groß, um in das nächste Netzwerk geroutet zu werden. Eine Fragmentierung ist erforderlich, aber nicht erlaubt. Diese Meldung wird nur bei IPv4 von einem Router gesendet. Nähere Informationen über diese Meldung finden Sie im nächsten Abschnitt.

Die meisten dieser Fehlermeldungen weisen auf fatale Fehler bei der Verbindung hin. Ein Client, der diese Meldungen erhält, erkennt den Fehler und bricht die Verbindungsversuche ab. Lediglich bei der Meldung *fragmentation-needed* wird ein erneuter Versuch mit einem kleineren Paket unternommen. Für eine Firewall bedeuten alle diese Fehlermeldungen mit Ausnahme der *fragmentation-needed*-Meldung, dass die Verbindung nicht aufrechterhalten werden kann. Die Firewall kann daher die Verbindung aus ihrer Zustandstabelle löschen.

Ein Filtern der ICMP-Meldungen ist sehr einfach möglich. Bereits im letzten Abschnitt habe ich die folgende Regel vorgestellt:

```
$IPTABLES -A FORWARD -p icmp --icmp-type destination-unreachable -m state ↵
        --state RELATED -j ACCEPT
```

Diese Regel akzeptiert alle ICMP-Fehlermeldungen vom Typ *destination-unreachable* unabhängig von ihrer Richtung und den IP-Adressen. Es ist lediglich erforderlich, dass die Firewall die Fehlermeldung einer Verbindung zuordnen kann.

Jedoch können diese Pakete auch Informationen über interne Systeme preisgeben. Zum einen kann ein Angreifer erkennen, ob ein System existiert, ob es bestimmte Protokolle kennt und ob die entsprechenden Ports offen sind.

So kann ein Angreifer zum Beispiel durch Senden eines ESP- oder AH-Pakets herausfinden, ob das System ein VPN-Gateway ist oder nicht. Normale Systeme antworten mit einer *protocol-unreachable*-Meldung. Ein VPN-Gateway versucht das Paket zu analysieren und zu verarbeiten.

Teilweise geben die Systeme in den Fehlermeldungen mehr Informationen preis, als dem Administrator lieb ist. So schickten Linux-Kernel der Version 2.0 beliebige Speicherinhalte in den Fehlermeldungen: `http://www.secdev.org/adv/CARTSA-20030314-icmpleak`. Diese Lücke ist behoben. Jedoch sind einige Windows-Versionen von ähnlichen Problemen betroffen.

Daher ist es sinnvoll, die Anzahl der Meldungen zu begrenzen. Wenn Sie eine Firewall implementieren, um ein internes Netz bei dem Zugriff auf das Internet zu schützen, so sollten Sie ICMP-*destination-unreachable*-Meldungen aus dem Internet in Ihr geschütztes Netz erlauben. Tun Sie dies nicht, werden Ihre Clients bei dem Zugriff auf das Internet nicht über Fehler bei dem Verbindungsaufbau informiert und werden es mehrfach erneut versuchen, bis ein clientinterner Timeout auftritt. Ihre Benutzer werden über diese Verzögerung nicht begeistert sein. Wenn Sie die Fehlermeldungen zulassen, erhält der Client sofort eine entsprechende Nachricht und kann die Information sofort an den Benutzer oder die Applikation weitergeben.

Der Transport von ICMP-*destination-unreachable*-Meldungen aus Ihrem internen Netz in das Internet ist mit einer Ausnahme (siehe Abschnitt 35.3) nicht erforderlich.

Sie können also folgende Regel verwenden:

```
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -p icmp --icmp-type ↵
        destination-unreachable -m state --state RELATED -j ACCEPT
```

Natürlich ist der Transport von ICMP-*destination-unreachable*-Nachrichten aus der DMZ unter Umständen sinnvoll. Das hängt davon ab, ob Sie den Clients, die auf Ihre Dienste zugreifen, Fehlermeldungen senden möchten oder nicht. Sie können mit der folgenden Regel die *destination-unreachable*-Meldungen zulassen:

```
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -p icmp --icmp-type ↩
        destination-unreachable -m state --state RELATED -j ACCEPT
```

35.3 *fragmentation-needed*

Diese ICMP-Meldung (beziehungsweise ihre falsche Filterung) ist für eine Vielzahl von Problemen im Internet verantwortlich. Schuld ist die Tatsache, dass unterschiedliche Netzwerkmedien unterschiedliche Paketgrößen transportieren können. Die maximale Größe ist die Maximum Transmission Unit (MTU). Bei dem Medium Ethernet beträgt diese 1500 Bytes. Andere Medien, wie Token Ring, einige VPN-Verbindungen oder auch DSL-PPPoE, nutzen unterschiedliche MTUs. Befindet sich nun ein Client in einem Netz A mit einer besonders großen MTU und versendet er ein Paket an einen Empfänger in einem Netz B mit einer kleineren MTU, so besteht die Gefahr, dass das Paket zu groß für das Netz B ist. Entweder wird das Paket fragmentiert oder es wird verworfen, und dabei wird diese Fehlermeldung erzeugt. Da eine Fragmentierung bei einem Paketverlust mit einem Overhead verbunden ist, führen fast alle modernen Betriebssysteme eine Path Maximum Transmission Unit Discovery (PMTUD, RFC1191) durch. Ab IPv6 ist diese sogar Pflicht! Sie ermitteln die MTU für den kompletten Pfad und bauen ihre Pakete dann in passender Größe. Die PMTUD ist aber auf die korrekte Zustellung dieser Fehlermeldung angewiesen. Erreicht die Fehlermeldung nicht ihr Ziel, kann die Verbindung nicht aufgebaut werden. Dummerweise kann der betroffene Client kaum etwas an dieser Tatsache ändern (siehe auch Abschnitt 17.25). Eine genauere technische Betrachtung finden Sie in dem folgenden Abschnitt.

35.3.1 MTU und die Path-MTU-Discovery

Da die MTU für unterschiedliche Netzwerkmedien unterschiedliche Werte haben kann, kann es vorkommen, dass ein Rechner so große Pakete erzeugt, dass ein Router auf dem Weg zum Ziel diese Pakete nicht weiterleiten kann. In Abbildung 35.1 ist ein Token-Ring-Netz und ein Ethernet dargestellt. Die MTU des Token-Ring (16 MBit/s) beträgt 17914 Bytes. Die MTU des Ethernet beträgt 1500 Bytes. Sendet nun der Rechner Eins an den Rechner Zwei 20 Kbyte Daten, so erzeugt er zwei Pakete. Das erste Paket ist etwa 17 Kbyte groß und das zweite etwa 3 Kbyte. Der Router kann diese Pakete jedoch nicht unverändert weitersenden, da die MTU des Ethernet nur 1500 Bytes beträgt. Er fragmentiert die Pakete. Dazu schneidet er zunächst das Paket in kleinere Bestandteile und kopiert jeweils den originalen IP-Header des Pakets davor. Die Länge dieser Bestandteile muss ein Vielfaches von acht aufweisen. Zusätzlich setzt der Router in allen Fragmenten außer dem letzten das More-Fragments-Bit (MF). Dieses zeigt dem Empfänger an, dass weitere Fragmente folgen. Damit der Empfänger die Fragmente richtig zusammensetzen kann, gibt er noch den Offset des Fragments im Gesamtpaket als

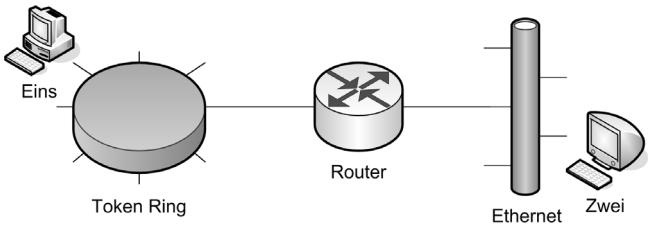


Abbildung 35.1: **Ein Token-Ring-Netz und ein Ethernet sind über einen Router verbunden.**

Vielfaches von acht an. Die Fragmente werden nun weiter zum Empfänger gesendet, der die Fragmente zusammenbaut und das IP-Paket prozessiert.

Wenn nun aber eines dieser Fragmente verloren geht, wartet der Empfänger vergebens auf dieses Fragment. Nach einem Timeout sendet er eine ICMP-*time-exceeded*-Fehlermeldung (siehe Abbildung 35.1CE[X] und Abschnitt 35.6). Der Absender sendet das Paket erneut, es wird erneut fragmentiert, und es besteht wieder die Gefahr, dass ein Fragment verloren geht. Das einzelne Fragment kann nicht neu angefordert werden. Der Router hat das Fragment nicht gespeichert, und der Absender weiß nichts von einer Fragmentierung. Bei einem Verlust von 1500 Bytes werden ca. 17.000 Bytes neu übertragen!

Die Path-MTU-Discovery versucht, dieses Problem intelligent zu lösen. Hierzu sendet ein Client, bei dem PMTUD aktiviert ist, alle Pakete mit einem gesetzten Don't-Fragment-Bit (DF). Dieses Bit weist alle Router an, das Paket nicht zu fragmentieren. In unserem Fall kann dann das Paket jedoch nicht weitergesendet werden. Der Router verwirft das Paket und sendet eine Fehlermeldung an den Absender. In dieser Fehlermeldung (*fragmentation-needed*) informiert der Router den Absender auch über die MTU des nächsten Netzes. Der Absender kann das Paket neu mit einer passenderen Größe bauen. Dieses Paket kann dann weitertransportiert werden. Befindet sich anschließend ein weiteres Netz mit einer noch kleineren MTU auf dem Weg zum Ziel, erfolgt dieser Vorgang erneut.

Sicherlich setzt kaum noch jemand heutzutage Token-Ring-Netze ein. Da fast alle Netze auf Ethernet basieren, könnte man annehmen, dass sich dieses Problem überlebt hat. Jedoch taucht es seit der Einführung von DSL-Internetzugängen verstärkt wieder auf. Die DSL-Anbindung des Rechners an das Modem erfolgt über eine Ethernet-Leitung. Dennoch beträgt die MTU für das IP-Protokoll nicht 1500 Bytes, da das IP-Protokoll noch in ein weiteres Protokoll eingepackt wird. In Deutschland handelt es sich meist um PPPoE. In anderen Ländern wird auch PPTP eingesetzt. Dadurch verringert sich die MTU für das IP-Protokoll um mindestens acht Bytes (PPPoE) auf 1492 Bytes.

In Kombination mit einer Firewall können nun zwei Probleme auftreten:

1. Die Firewall kann fragmentierte Pakete nicht richtig filtern.
2. Die Firewall erlaubt nicht die *fragmentation-needed*-Nachricht.

CE[X] Im Bild nicht zu sehen.

Editor's or typesetter's annotations (will be removed before the final TEX run)

Wenn ein Paketfilter fragmentierte Pakete filtern soll, taucht ein besonderes Problem auf. Die Filterung des ersten Fragments eines HTTP-Pakets ist kein Problem, denn dieses Fragment weist sowohl einen IP- als auch einen TCP-Header auf. Ab dem zweiten Fragment enthält das Fragment aber nur noch einen IP-Header, da der Router lediglich den IP-Header kopiert. Schließlich arbeitet ein Router auch auf der Schicht Drei und nicht auf der Schicht Vier. Ein klassischer Router kann mit einem TCP-Header gar nichts anfangen. Der Paketfilter kann nun ab dem zweiten Fragment nicht mehr entscheiden, ob es sich um ein HTTP- oder ein Telnet-Paket handelt. Soll er das Paket akzeptieren oder verwerfen?

Die ersten Paketfilter hatten eine einfache Lösung parat: Sie filterten nur das erste Fragment und ließen immer alle weiteren Fragmente außer dem ersten durch! Erhielt der Empfänger auch das erste Fragment, so konnte er das Paket defragmentieren und auswerten. Wurde das erste Fragment durch eine Firewall verworfen, konnte der Empfänger das Paket nicht defragmentieren und musste auch alle weiteren Fragmente verwerfen. Leider gab es in der Vergangenheit mehrfach Angriffe, die diese einfache Lösung umgingen. Daher sammeln heute alle Paketfilter zunächst sämtliche Fragmente eines Pakets, defragmentieren diese und analysieren erst anschließend das Paket mit ihren Regeln. Wird das Paket akzeptiert, so wird es weitertransportiert und möglicherweise hierzu erneut fragmentiert. Wird das Paket verworfen, so werden auch alle Fragmente verworfen.

Bei dem zweiten Problem handelt es sich nicht um ein Sicherheitsproblem, sondern vielmehr um eine Art von Denial-of-Service. In Abbildung 35.2 sehen Sie eine typische Konstellation, wie sie heute im Internet anzutreffen ist.

Der DSL-Anwender greift auf einen Webserver im Internet zu. Dieser Webserver wird durch eine Firewall geschützt, die grundsätzlich keine ICMP-Meldungen von außen erlaubt. Dies ist nicht ungewöhnlich, da bis vor 10 Jahren viele zustandsorientierte Paketfilter nicht in der Lage waren, ICMP-Meldungen sicher zu filtern. Ihre Stateful Inspection beschränkte sich nur auf TCP und konnte mit dem ICMP-Protokoll nicht umgehen.

Solange der Webserver nur kleine Datenmengen versendet, tritt kein Problem auf. Sobald der Webserver aber mehr als 1492 Byte große Pakete erzeugt, werden diese von dem DSL-Router auf der Seite des Internet-Service-Providers als zu groß abgelehnt. Dieser Router erzeugt

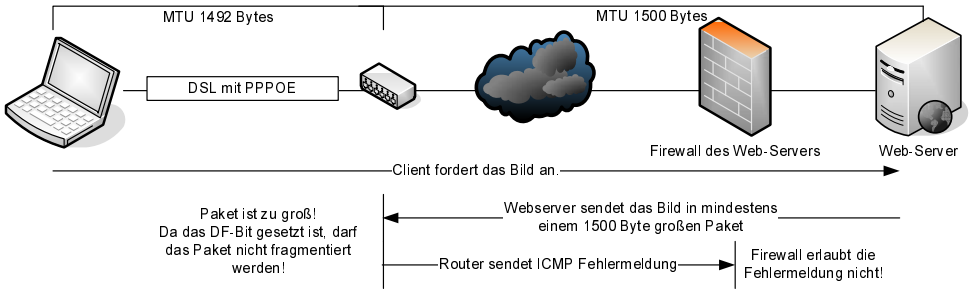


Abbildung 35.2: **Ein DSL-Anwender greift auf einen Webserver zu. Der Webserver wird durch eine Firewall geschützt. Die PMTUD funktioniert aufgrund der Firewall-Konfiguration nicht.**

eine ICMP-*fragmentation-needed*-Nachricht und sendet sie an den Webserver. Die Firewall verhindert, dass diese Nachricht den Webserver erreicht. Der Webserver erhält keine Empfangsbestätigung (TCP-ACK) und sendet das Paket nach einiger Zeit erneut. Da er den Grund für den Paketverlust nicht kennt, sendet er das Paket unverändert neu. Dieses neue Paket wird erneut verworfen, und eine Fehlermeldung wird generiert. Nach einiger Zeit wird der Webserver aufgeben und die Verbindung abbrechen. Die Daten werden den Client nicht erreichen.

Dummerweise ist der Client davon betroffen und kann kaum etwas zur Behebung des Problems tun. Als einzige Lösung im Fall des TCP-Protokolls kann er die Maximum Segment Size (MSS) setzen (siehe Abschnitt 17.25).

Die Fehlermeldung *fragmentation-needed* sollten Sie grundsätzlich in allen Richtungen durch Ihre Firewall erlauben. Ansonsten kann es bei der heutigen IT-Infrastruktur mit DSL und VPNs zu Verbindungsproblemen kommen, die Sie und Ihre Kunden betreffen. Am einfachsten erfolgt das mit der folgenden Zeile:

```
$IPTABLES -A FORWARD  -p icmp --icmp-type fragmentation-needed -m state ↵
        --state RELATED -j ACCEPT
```

35.4 *source-quench*

Diese Meldung kann ein Router oder ein Rechner versenden, wenn er die empfangenen Pakete nicht schnell genug verarbeiten kann und daher einzelne Pakete unverarbeitet aus seiner Empfangswarteschlange verwerfen muss. Da diese Pakete später erneut gesendet werden müssen und dadurch Verzögerungen auftreten, ist es sinnvoller, den Absender zu informieren, dass die Pakete langsamer gesendet werden müssen.

Heutzutage verwendet kaum noch ein Router diese Meldungen. Source-Quench ist durch moderne Algorithmen abgelöst worden. Auch die meisten TCP-Stacks verfügen über mächtigere Methoden der Flusskontrolle. Jedoch reagieren alle mir bekannten Betriebssysteme noch auf *source-quench*-Nachrichten. So ist es möglich, mit gefälschten *source-quench*-Nachrichten existierende Verbindungen zu verlangsamen. Dies kann bis zum Denial-of-Service führen (siehe http://xforce.iss.net/xforce/xfdb/17429).

Ein einfaches Werkzeug für die Erzeugung dieser Meldungen ist tcpnice von Dug Song. Es ist Teil des dsniff-Pakets: http://www.monkey.org/~dugsong/dsniff.

Sie sollten es nicht erlauben, dass diese *source-quench*-Nachrichten durch Ihre Firewall gelangen. Es könnte sinnvoll sein, diese Nachrichten zu protokollieren, sodass Sie erkennen können, ob einzelne Clients diese Nachrichten doch benötigen.

```
$IPTABLES -A FORWARD  -p icmp --icmp-type source-quench -j LOG --log- ↵
        prefix "Source-Quench: "
```

35.5 *redirect*

Die ICMP-*redirect*-Nachricht erlaubt es einem Router, einen Rechner auf einen besseren Router hinzuweisen. Wann wird eine ICMP-Redirect-Nachricht versandt (siehe Abbildung 5.17)?

1. Rechner A sendet ein Paket an Rechner B. Da Rechner B sich in einem anderen Netz befindet, sendet Rechner A das Paket an sein Default-Gateway G1.
2. Das Gateway G1 prüft seine Routing-Tabelle und stellt fest, dass der nächste Hop für das Paket das Gateway G2 ist.
3. Wenn sich der Rechner A, das Gateway G2 und das Gateway G1 in demselben Netz befinden, sendet das Gateway G1 eine Redirect-Nachricht an den Rechner A und teilt ihm mit, dass das Gateway G2 ein besserer Router auf dem Weg zum Ziel Rechner B sei.
4. Außerdem leitet das Gateway G1 das Paket an das Gateway G2 weiter.
5. Der Rechner A trägt das neue Gateway G2 in seiner Routing-Tabelle oder seinem Routing-Cache ein und verwendet nun dieses Gateway, um den Rechner B zu erreichen.

Sicherlich möchten Sie nicht, dass irgendjemand Ihrer Firewall oder einem Rechner in Ihrer DMZ oder Ihrem internen Netzwerk von außen einen anderen „besseren“ Router unterschieben kann. Diese ICMP-Nachricht kann natürlich für Angriffe in Form von Router-Spoofing verwendet werden. Sie sollten diese Nachricht nicht über Ihre Firewall zulassen. Vielleicht ist es sinnvoll, diese Nachrichten zu protokollieren (wenn Sie die Protokolle auch lesen):

```
$IPTABLES -A FORWARD  -p icmp --icmp-type redirect -j LOG --log-prefix " ➥
        Source-Quench: "
```

35.6 *time-exceeded*

Diese ICMP-Nachricht zeigt eine Zeitüberschreitung bei dem Pakettransport an. Bei dem Transport eines Pakets kann es zu zwei Arten der Zeitüberschreitung kommen:

» Bei der Defragmentierung eines Pakets hat der Empfänger nicht alle Fragmente innerhalb seiner vorgeschriebenen Zeit erhalten. Er verwirft das Paket und schickt eine *time-exceeded*: *ttl-zero-during-reassembly*-Nachricht zurück.

» Bei dem Transport eines Pakets hat das Paket seine Lebensdauer überschritten. Jedes Paket trägt im IP-Header einen Time-To-Live-Wert (TTL). Ursprünglich sollte dieser Wert für jede Sekunde des Transports um 1 reduziert werden. Da dies nur bei zeitsynchronisierten Systemen möglich ist, reduziert jeder Router, der das Paket weiterleitet, diesen Wert um eins. Erreicht der Wert 0, so verwirft der Router das Paket und sendet eine *time-exceeded*: *ttl-zero-during-transit*-Meldung an den Absender. Hiermit sollen Paket-Irrläufer durch fehlerhafte Routerkonfiguration vermieden werden.

Wenn Sie eine Firewall zum Schutz eines internen Netzes bei dem Zugriff auf das Internet einsetzen, möchten Sie wahrscheinlich diese Fehlermeldungen in Richtung Ihrer Clients zulassen, damit diese bei Fehlern in der Konfiguration eines Routers von Paketen informiert wer-

den. Fehler bei der Defragmentierung treten heute so gut wie gar nicht mehr auf, da die meisten Betriebssysteme die Path-MTU-Discovery (siehe Abschnitt 35.3.1) unterstützen.

```
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -p icmp --icmp-type time- ↵
        exceeded -m state --state RELATED -j ACCEPT
```

35.7 *parameter-problem*

Diese Fehlermeldung wird von einem System gesendet, wenn es aufgrund eines Fehlers in dem IP-Header ein Paket nicht verarbeiten konnte. Hierfür gibt es viele verschiedene Gründe. Die Fehlermeldung weist den Absender mit einem Zeiger auf den Grund hin. Es kann jedoch auch sein, dass eine IP-Option im Header für die Zustellung fehlte. Da die Option fehlt, kann nicht auf sie verwiesen werden. Hierfür gibt es dann die besondere Meldung *parameter-problem*: *required-option-missing*.

Diese Fehlermeldungen treten in modernen Netzen mit den ausgereiften IP-Stacks nur sehr selten auf, sodass ich üblicherweise empfehle, die Meldungen zu protokollieren und zu verwerfen:

```
$IPTABLES -A FORWARD  -p icmp --icmp-type parameter-problem -j LOG --log- ↵
        prefix "Parameter-Problem: "
```

35.8 *router-advertisement/router-solicitation*

Für das Funktionieren des IP-Protokolls ist es erforderlich, dass jeder Rechner die für ihn zuständigen Router kennt. Üblicherweise werden die Router eines Systems in Konfigurationsdateien festgelegt. Da dies aber einen gewissen manuellen Administrationsaufwand erzeugt, wurde die automatische Routererkennung entwickelt. Diese Routererkennung (Router Discovery, RFC1256) basiert auf den Router-Advertisement- und Router-Solicitation-Nachrichten. Hierzu kann ein Router in regelmäßigen Abständen (üblicherweise alle 10 Minuten) ICMP-*router-advertisement*-Nachrichten aussenden. Diese Nachrichten enthalten wichtige Informationen über den Router, wie zum Beispiel seine IP-Adresse. So kann der Rechner die Informationen in seiner Routingtabelle eintragen.

STOP

Diese Funktion wird bei IPv4 eigentlich nie genutzt. IPv6 macht jedoch von den entsprechenden ICMPv6-Nachrichten regen Gebrauch von ihr (siehe auch Kapitel 36).

Wenn nun ein Rechner neu gestartet wird, erhält er spätestens nach 10 Minuten die notwendigen Routerinformationen. Da dies häufig nicht akzeptabel ist, kann der Rechner auch eine *router-solicitation*-Nachricht senden. Diese weist alle Router in dem lokalen Netzwerk an, jetzt ihre Router-Advertisement-Nachricht zu senden.

Die *router-advertisement*-Nachricht wird an die All-Devices-Multicast-Adresse (224.0.0.1) geschickt, und die *router-solicitation*-Nachricht wird an die All-Routers-Multicast-Adresse (224.0.0.2) gesendet.

Sicherlich wollen Sie derartigen Nachrichten nicht erlauben, Ihre Firewall zu durchqueren.

35.9 *timestamp-request/timestamp-reply*

Mit den ICMP-*timestamp-request*- und *timestamp-reply*-Nachrichten (RFC792) kann ein Rechner sowohl die Uhrzeit eines anderen Rechners ermitteln als auch die Wegezeit berechnen. Der Absender der *timestamp-request*-Nachricht trägt seine Uhrzeit in Millisekunden seit Mitternacht (UTC) ein (Originator's Timestamp). Der Empfänger trägt bei Erhalt der Nachricht seine eigene Uhrzeit in das Receive-Timestamp-Feld ein. Die Transmit Timestamp trägt er kurz vor dem Versand der *timestamp-reply*-Nachricht ein (siehe Abbildung 35.3).

In einer gewissen Weise ähnelt dieser Mechanismus dem Ping (siehe unten), bietet aber mit der absoluten Uhrzeit noch zusätzliche Informationen. Die meisten modernen Betriebssysteme reagieren auf diese Nachrichten, obwohl nur sehr wenige Werkzeuge existieren, die diese Nachrichten erzeugen und auswerten können. Ein allgemein verfügbares Werkzeug ist `nmap`. Weitere Werkzeuge sind `icmppush` und `sing` (`http://sourceforge.net/projects/sing/`). Nmap kann mit der Option `-PP` mit *timestamp*-Nachrichten nach anderen Systemen im lokalen Netz suchen:

```
[root@bibo ~]# nmap -PP -sP 192.168.255.0/24
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2005-07-29 13:23 ↵
        CEST
Host 192.168.255.1 appears to be up.
MAC Address: 00:13:10:1F:B7:D1 (Unknown)
Host 192.168.255.100 appears to be up.
Host inspiron-8100.opensource-training.de (192.168.255.125) appears to be ↵
         up.
MAC Address: 00:20:E0:6C:72:1E (Actiontec Electronics)
Nmap finished: 256 IP addresses (3 hosts up) scanned in 2.295 seconds
```

Das Werkzeug `sing` ist noch mächtiger und erlaubt es, jede Art von ICMP-Nachricht zu erzeugen.

Sicherlich wollen Sie diese Nachrichten nicht von außen durch Ihre Firewall erlauben. Die Filterung dieser Nachrichten unterscheidet sich von den anderen ICMP-Nachrichten, da es sich hier nicht um eine Fehlermeldung handelt. Man kann fast von einem Client und einem Server, die miteinander kommunizieren, sprechen. Daher filtert auch Iptables diese Nach-

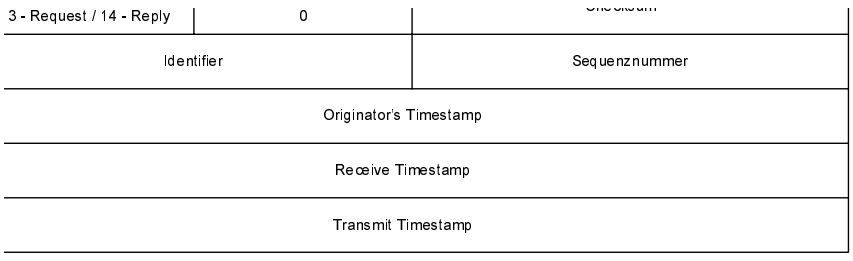


Abbildung 35.3: **Die Timestamp-Nachrichten**

richten anders! Jeder Timestamp-Request wird als neuer Verbindungsaufbau (State: `NEW`) gewertet. Jeder Timestamp-Reply wird als Teil einer aufgebauten Verbindung gewertet (State: `ESTABLISHED`). Zusätzlich beendet jeder Timestamp-Reply auch die Verbindung und entfernt sie aus der Tabelle. Die Zuordnung der Reply-Nachrichten zu den entsprechenden Request-Nachrichten erfolgt außer über die IP-Adresse über den Identifier und die Sequenznummer der Nachricht.

Möchten Sie die Verwendung dieser Nachrichten für Testzwecke oder für Nmap von innen durch Ihre Firewall nach außen zulassen, können Sie folgende Regel nutzen:

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p icmp --icmp-type timestamp- ↵
        request -m state --state NEW -j ACCEPT
```

Die Funktion dieser Regel setzt natürlich voraus, dass Sie eine allgemeine Regel für die Akzeptanz aller `ESTABLISHED`-Pakete verwenden.

35.10 *address-mask-request/address-mask-reply*

Diese Nachrichten (RFC950) erlauben es einem Rechner, einen anderen Rechner nach seiner Netzmaske zu fragen. Ursprünglich wurde diese Funktion implementiert, um einem Rechner, der keine Informationen über sein lokales Netz besitzt, die Möglichkeit zu geben, andere Rechner in demselben Netz nach der gültigen Netzmaske zu fragen. Dazu sendet der Rechner eine Address-Mask-Request-Nachricht entweder an die Broadcast-Adresse (255.255.255.255) oder, falls er den lokalen Router kennt, an den Router. Dies ähnelt der Möglichkeit, mithilfe der Router-Solicitation-Nachricht den lokalen Router zu ermitteln. Normalerweise werden Address-Mask-Reply-Nachrichten nur von Routern versandt.

Auch diese Nachrichten möchten Sie normalerweise nicht durch eine Firewall in Ihr geschütztes Netz lassen. Vielleicht möchten Sie aber Werkzeuge wie `sing -mask <ziel>` nutzen. Dann benötigen Sie die folgende Regel:

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p icmp --icmp-type address- ↵
        mask-request -m state --state NEW -j ACCEPT
```

35.11 *echo-request/echo-reply* (Ping)

Diese Nachrichten werden Sie sicherlich kennen. Und wenn Sie nicht die Nachrichten kennen, so kennen Sie zumindest den Befehl, der diese Nachrichten erzeugt: `ping`. Der Ping-Befehl erzeugt eine *echo-request*-Nachricht und sendet diese an den Zielrechner. Dieser beantwortet sie mit einer *echo-reply*-Nachricht. Da hier die Nachrichten wieder in einem scheinbaren Client-Server-Verhältnis stehen, kann Iptables die Nachrichten auch so filtern. Jedes Echo-Request-Paket wird als neue Verbindung erkannt (State: `NEW`) und eingetragen. Jedes Echo-Reply-Paket wird der entsprechenden Verbindung zugeordnet (State: `ESTABLISHED`) und beendet die Verbindung. Es werden also nicht mehrere Echo-Reply-Pakete für ein Echo-

Request-Paket akzeptiert. So können die Echo-Nachrichten sicher von einer Iptables-Firewall gefiltert werden.

Der `ping`-Befehl auf einem Linux-System kann auch einen Broadcast-Ping durchführen. Manche Implementierungen erlauben auch einen Multicast-Ping. Hierbei wird der Echo-Request an die Broadcast- oder Multicast-Adresse geschickt. Sämtliche Rechner, die auf diese Anfrage reagieren (UNIX, Router, kein MS Windows), antworten. Da nun aber nur eine Echo-Reply-Antwort von Iptables zugelassen wird und anschließend die Verbindung aus der Zustandstabelle gelöscht wird, ist ein Broadcast-Ping nicht möglich, wenn die Echo-Request- und Echo-Reply-Pakete zustandsorientiert gefiltert werden.

Mit der folgenden Regel erlauben Sie Ihren Benutzern, die Erreichbarkeit von Rechnern im Internet mit dem `ping`-Befehl zu testen:

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p icmp --icmp-type echo- ↵
        request -m state --state NEW -j ACCEPT
```

Häufig wird zusätzlich auch die Ping-Funktionalität bei dem Zugriff auf die DMZ gewünscht. Sie möchten aus Ihrem internen Netz und aus dem Internet prüfen, ob Ihr Webserver noch läuft.

```
$IPTABLES -A FORWARD -o $DMZDEV -p icmp --icmp-type echo-request -m state ↵
         --state NEW -j ACCEPT
```

Hier habe ich auf die Angabe der eingehenden Schnittstelle verzichtet. So wird nun jedes Echo-Request-Paket akzeptiert, das in die DMZ geroutet werden soll. Dies erfolgt unabhängig davon, ob das Paket von innen oder von außen die Firewall erreicht hat.

Gerade bei Ping ist die zustandsorientierte Filterung wichtig. Sie könnten auf die Idee kommen, grundsätzlich jedes Echo-Reply-Paket zu akzeptieren, um so auch Broadcast-Pings zu ermöglichen. Dann öffnen Sie aber möglichen Tunnel-Werkzeugen Tür und Tor. Es existiert eine Reihe von Werkzeugen, die auf der Basis von Echo-Paketen jedes beliebige Protokoll tunneln können. Eines der ersten Werkzeuge war `itunnel` von dem Hacker Team Teso. Die zustandsorientierte Filterung würde diesen Tunnel nicht erlauben.

35.12 Traceroute

Traceroute ist keine ICMP-Nachricht. Dennoch habe ich den Befehl hier aufgenommen, da die Traceroute-Funktion komplett auf ICMP-Nachrichten basiert. Vielleicht haben Sie sich schon immer gefragt, wie der `traceroute`-Befehl funktioniert. Die folgende Ausgabe kennen Sie sicherlich:

```
[spenneb@bibo ~]$ traceroute www.bsd-training.de
traceroute to www.bsd-training.de (81.169.129.12), 30 hops max, 38 byte ↵
        packets

 1  192.168.255.1 (192.168.255.1)  0.846 ms  0.764 ms  0.711 ms
```

```
 2  217.0.116.135 (217.0.116.135)  77.835 ms  46.357 ms  47.983 ms
 3  217.0.73.82 (217.0.73.82)  47.190 ms  46.582 ms  48.130 ms
 4  l-ea1.L.DE.net.DTAG.DE (62.154.89.122)  57.817 ms  57.110 ms  57.894 ↵
        ms
 5  so-1-1-1-0.lpz2-j2.mcbone.net (62.104.199.89)  59.129 ms  58.560 ms ↵
        171.903 ms
 6  L0.bln2-g.mcbone.net (62.104.191.139)  61.900 ms  62.711 ms  60.838 ↵
        ms
 7  strato-bln1.fdknet.de (194.97.172.146)  60.145 ms  59.459 ms  65.330 ↵
        ms
 8  81.169.160.38 (81.169.160.38)  57.699 ms  58.237 ms  58.084 ms
 9  81.169.160.158 (81.169.160.158)  57.951 ms  57.246 ms  58.089 ms
10  bsd-training.de (81.169.129.12)  59.158 ms  59.379 ms  58.027 ms
```

Der Traceroute-Befehl ermittelt die Router, über die ein Paket sein Ziel erreicht. Dabei geht das nicht immer so gut wie in diesem Fall. Häufig werden ab einem bestimmten Zeitpunkt Sternchen (`* * *`) angezeigt. Dann filtert eine Firewall den Verkehr:

```
[spenneb@bibo ~]$ traceroute www.redhat.de
traceroute to www.redhat.de (66.187.229.16), 30 hops max, 38 byte packets
 1  192.168.255.1 (192.168.255.1)  0.871 ms  4.139 ms  0.940 ms
 2  217.0.116.135 (217.0.116.135)  49.293 ms  79.062 ms  47.944 ms
 3  217.0.73.82 (217.0.73.82)  46.832 ms  46.187 ms  48.056 ms
 4  f-ea2.F.DE.net.DTAG.DE (62.154.18.18)  53.053 ms  51.602 ms  52.912 ↵
        ms
 5  62.156.139.146 (62.156.139.146)  55.010 ms  53.860 ms  55.884 ms
 6  ar1.str.de.colt.net (212.121.151.242)  63.789 ms  70.990 ms  57.906 ↵
        ms
 7  * * *
 8  * * <Strg-C>
```

Was versendet der Traceroute-Befehl, und warum erhält er eine Antwort von den Routern auf dem Weg zum Ziel? Häufig hört man auch von langjährigen Administratoren Erklärungen, dass das Paket die Router unterwegs aufzeichne oder ähnlichen Unsinn. Im Grunde ist es ganz einfach. Der Befehl versendet ein IP-Paket mit der Zieladresse, die bei dem Aufruf angegeben wird. Um eine Antwort von den Routern und nicht vom Ziel zu erhalten, modifiziert der Befehl den TTL-Wert im IP-Header des Pakets. Zunächst sendet der Befehl drei Pakete mit einem TTL-Wert von eins. Der erste Router reduziert den TTL-Wert um eins und muss das Paket verwerfen, da der TTL-Wert null erreicht hat. Zusätzlich sendet er dreimal – für jedes Paket eine einzelne – eine ICMP-*time-exceeded*-Fehlermeldung an den Absender. Der Absender erhält also drei Fehlermeldungen von dem Router mit dessen IP-Adresse. Für die IP-Adresse führt der Befehl nun noch eine Rückwärts-Namensauflösung durch (außer Sie geben beim Aufruf des Befehls die Option `-n` an). Der Befehl gibt nun diese Informationen aus:

```
 1  192.168.255.1 (192.168.255.1)  0.871 ms  4.139 ms  0.940 ms
```

Nun erzeugt der Befehl drei weitere Pakete, die sich nur in dem TTL-Wert unterscheiden. Dieser ist nun zwei. Der erste Router reduziert den TTL-Wert und routet das Paket weiter an den zweiten Router, der dann das Paket verwirft und nun die *time-exceeded*-Mitteilung mit seiner Absenderadresse verschickt. So erhält der Client die IP-Adresse des zweiten Routers.

Nun stellt sich die Frage, was für ein Paket der Traceroute-Befehl versendet? Ein IP-Paket mit wachsendem TTL-Wert, aber was für ein IP-Paket? Wenn Sie den Windows-`tracert.exe`-Befehl verwenden, handelt es sich um ein ICMP-*echo-request*-Paket. So kann der `tracert.exe`-Befehl leicht ermitteln, wann er das tatsächliche Ziel erreicht hat. Solange der TTL-Wert zu klein ist, um das Ziel zu erreichen, erhält er *time-exceeded*-Meldungen. Sobald das Ziel erreicht wurde, erhält der Befehl eine *echo-reply*-Nachricht von dem Zielhost.

Der UNIX-`traceroute`-Befehl arbeitet anders, auch wenn einige Varianten mit der Option `-I` ICMP-*echo-request*-Nachrichten verwenden können. Dieser Befehl versendet per Default UDP-Pakete. UDP-Pakete benötigen einen Quell- und einen Zielport. Der Quellport wird zufällig gewählt (>1023). Der Zielport wird nach einer einfachen Formel gewählt: BASE + TTL - 1. Der BASE-Wert kann bei dem Befehl mit der Option `-p` frei gewählt werden. Der Defaultwert ist 33434. Die Zielports liegen also im Bereich 33434–33689. Solange der TTL-Wert zu klein ist, um den Zielrechner zu erhalten, erhält der `traceroute`-Befehl eine *time-exceeded*-Nachricht. Sobald der Zielrechner erreicht wird, basiert die Portwahl auf der Hoffnung, dass auf diesen Ports kein Dienst horcht. Üblicherweise sendet dann ein Betriebssystem eine ICMP-*port-unreachable*-Nachricht zurück. So kann der `traceroute`-Befehl erkennen, dass er sein Ziel erreicht hat. Wird der Port von einem Dienst bedient, kommt es auf den Dienst an, ob eine Antwort erfolgt. Antwortet der Dienst nicht auf das UDP-Paket, so wird das nächste Paket den nächsthöheren Port ansprechen, der dann hoffentlich nicht von einem Dienst bedient wird.

Wenn Sie nun mit Traceroute die Erreichbarkeit von Systemen im Internet über Ihre Firewall testen möchten, müssen Sie folgende Regeln verwenden:

```
# Akzeptiere Time-Exceeded-Nachrichten von außen. Für Windows und Unix
# erforderlich
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -p icmp --icmp-type time- ↵
        exceeded -m state --state RELATED -j ACCEPT

# Für Windows-tracert.exe erlaube echo-request-Pakete
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p icmp --icmp-type echo- ↵
        request -m state --state NEW -j ACCEPT

# Die folgende Regel akzeptiert die Antworten auf den Echo-Request
# durch den Ziel-Host. Diese Regel ist in den meisten
# Skripten bereits vorhanden
$IPTABLES -A FORWARD -m state --state ESTABLISHED -j ACCEPT
# Für UNIX-traceroute erlaube UDP-Pakete
```

```
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p udp --dport 33434:33689 -m ↵
        state --state NEW -j ACCEPT



# Erlaube Port-Unreachable-Nachrichten
# Dies ist meistens bereits der Fall, da von außen meistens alle Dest- ↵
        Unreach-
# Nachrichten akzeptiert werden
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -p icmp --icmp-type port- ↵
        unreachable -m state --state RELATED -j ACCEPT
```

Die Firewall selbst wird nun in den Ausgaben der Traceroute-Befehle nicht auftauchen, da sie selbst nicht das Recht hat, *time-exceeded*-Nachrichten zu versenden. Statt der Firewall werden drei Sterne in der Ausgabe erscheinen. Wenn Sie möchten, dass auch die Firewall angezeigt wird, können Sie die folgende Regel hinzufügen:

```
$IPTABLES -A OUTPUT -o $INTDEV -p icmp --icmp-type time-exceeded -m state ↵
         --state RELATED -j ACCEPT
```

Diese Regel erlaubt es der Firewall, das lokal erzeugte *time-exceeded*-Paket nach innen zu versenden.

35.12.1 Optimierung der ICMP-Regeln

Wenn Sie das ICMP-Protokoll so detailliert filtern möchten, wie ich es in diesem Kapitel beschrieben habe, ist es sinnvoll, die Regeln irgendwie ein wenig zu optimieren. Hierfür bieten sich die benutzerdefinierten Ketten an (siehe Abschnitt 8.3). Im Folgenden demonstriere ich das am Beispiel der ICMP-Pakete der `FORWARD`-Kette. Zunächst müssen Sie sicherstellen, dass auch Ihre ICMP-Regeln die Pakete tatsächlich betrachten können. Hierfür ist es wichtig, dass diese Regeln früh genug in der `FORWARD`-Kette betrachtet werden. Am einfachsten erstellen Sie zu Beginn Ihrer `FORWARD`-Kette die folgenden beiden Regeln:

```
$IPTABLES -A FORWARD -m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -p icmp -j MY_ICMP
```

Die erste Regel akzeptiert sämtliche Pakete, die zu aufgebauten Verbindungen gehören. Dies sollte immer in jeder Kette die erste Regel sein, da die meisten Pakete in diese Kategorie gehören. Bei einer einfachen DNS-Anfrage ist es jedes zweite Paket, bei einer TCP-Verbindung sind es alle Pakete ab dem zweiten Paket. Das können schnell auch mal mehrere Hundert Pakete sein. Da Sie entscheiden, welche Verbindung aufgebaut werden darf (State: `NEW`), ist dies auch sicher.

Anschließend prüfen Sie in der zweiten Regel, ob es sich um ein ICMP-Paket handelt, und springen dann in die benutzerdefinierte Kette `MY_ICMP`. Diese müssen Sie nun anlegen und mit Regeln füllen. Wenn Sie meinem Rat folgen, werden Sie alle Destination-Unreachable-

Meldungen von außen, alle Fragmentation-Needed-Meldungen, alle Time-Exceeded-Meldungen von außen (für Traceroute) und alle Echo-Request-Pakete von innen (für den Ping) zulassen. Ihre Regeln könnten dann folgendermaßen aussehen:

```
$IPTABLES -N MY_ICMP
# Destination-Unreachable von außen
$IPTABLES -A MY_ICMP -i $EXTDEV -p icmp --icmp-type destination- ↵
        unreachable -m state --state RELATED -j ACCEPT

# Fragmentation-Needed
$IPTABLES -A MY_ICMP -p icmp --icmp-type fragmentation-needed -m state -- ↵
        state RELATED -j ACCEPT

# Time-Exceeded von außen für traceroute
$IPTABLES -A MY_ICMP -i $EXTDEV -o $INTDEV -p icmp --icmp-type time- ↵
        exceeded -m state --state RELATED -j ACCEPT

# Echo-Request von innen für ping
$IPTABLES -A MY_ICMP -i $INTDEV -o $EXTDEV -p icmp --icmp-type echo- ↵
        request -m state --state NEW -j ACCEPT

# Protokolliere Parameter-Problem und Source-Quench
$IPTABLES -A MY_ICMP -p icmp --icmp-type source-quench -j LOG --log- ↵
        prefix "Source-Quench: "
$IPTABLES -A MY_ICMP -p icmp --icmp-type parameter-problem -j LOG --log- ↵
        prefix "Parameter-Problem: "

# Alle weiteren ICMP-Nachrichten werden unterdrückt
$IPTABLES -A MY_ICMP -j DROP
```

Die letzte Regel verwirft alle ICMP-Pakete, die im Vorfeld nicht akzeptiert wurden. Damit ist sichergestellt, dass am Ende dieser benutzerdefinierten Kette auch kein Rücksprung in die FORWARD-Kette erfolgt und dort möglicherweise noch Pakete akzeptiert werden oder einfach nur Prozessorleistung für unnötige Tests vergeudet wird.

Teil VIII

IPv6

Dieser Teil des Buches widmet sich dem IPv6-Protokoll. Hierzu werde ich Ihnen zunächst die Besonderheiten des Protokolls (siehe Kapitel 36) erläutern. Dann stelle ich die erweiterten Möglichkeiten des Befehls `ip6tables` vor (siehe Kapitel 39). Schließlich (in Kapitel 42) werde ich Ihnen die Regeln demonstrieren und erläutern, die Sie bei dem Einsatz des IPv6-Protokolls zwingend benötigen.

Das IPv6-Protokoll wird bisher kaum eingesetzt. Dennoch unterstützen Netfilter und der Befehl `ip6tables` bereits das IPv6-Protokoll. Während die älteren Linux-Kernel bis einschließlich 2.6.13 das IPv6-Protokoll nur wenig unterstützen, ist die Unterstützung ab dem Kernel 2.6.14 und besonders ab Version 2.6.20 vorhanden. Diese modernen Kernel können nun auch IPv6 zustandsorientiert filtern und unterstützen auch hier die Stateful-Inspection von FTP-Verbindungen, ICMP-Fehlernachrichten etc.

Um nicht das ganze Connection Tracking für das IPv6-Protokoll erneut schreiben zu müssen, hat das Netfilter-Team diese Funktionalität komplett überarbeitet und neu geschrieben, sodass die Zustandsüberwachung nun unabhängig vom Internetprotokoll sowohl für IPv4 als auch für IPv6 komplett unterstützt wird.

Leider existiert aber zumindest mit Red Hat Enterprise Linux (RHEL) 5 eine Distribution, die zwar den Kernel 2.6.18 einsetzt, der diese Funktion bieten könnte, bei der sich aber der Distributor gegen die Nutzung entschieden hat. Bis heute hat es für RHEL 5 keine Aktualisierung gegeben, die dieses Manko behebt. RHEL 5 lässt sich daher nicht sinnvoll als IPv6-Firewall einsetzen. Dies müssen Sie bei älteren Distributionen immer noch prüfen.

36. Das IPv6-Protokoll

Bevor wir uns mit der Filterung beschäftigen, möchte ich Ihnen eine ganz kurze Einführung in IPv6 geben. Diese soll Sie in die Lage versetzen, die restlichen Kapitel in diesem Teil des Buches zu verstehen. Für IPv4 finden Sie einen entsprechenden Überblick im Anhang A. TS² Wenn Sie bereits Erfahrung mit dem IPv6-Protokoll besitzen, können Sie gerne dieses Kapitel überspringen.

Ich werde das IPv6-Protokoll nicht in allen Einzelheiten und Details vorstellen können. Dies füllt ganze Bücher.[1] An einigen Stellen werde ich auch verallgemeinern und vereinfachen. Ich bitte das zu entschuldigen.

36.1 Hintergründe und Geschichte

Als in den 1970er-Jahren das Internet Protocol (IP) entworfen wurde, hat keiner seinen Erfolg vorhergesehen. Das Internet und damit auch das Internet-Protokoll ist ein Opfer seines Erfolges geworden. Da immer mehr Unternehmen, Behörden, Institute und Endanwender Zugang zu dem Internet wünschten, sind heute kaum noch IPv4-Adressen verfügbar. Die Internet Assigned Numbers Authority (IANA), die für die Verteilung der IP-Adressen an die Local Registries (AfriNIC, APNIC, ARIN, LACNIC und RIPE) zuständig ist, hat am 31. Januar 2011 die letzten freien Netze abgegeben.[2] Das APNIC, zuständig für den Asiatisch-Pazifischen Raum, hat am 15. April 2011 die letzten IPv4-Adressen vergeben.[3]

Diese Entwicklung wurde früh vorhergesehen und bereits viele Jahre früher erwartet. Ursprünglich waren die IP-Adressen in Klasse-A-, -B- und -C-Netze aufgeteilt. Diese Klassen waren festen IP-Adressbereichen zugewiesen:

Klasse	Bereich	Netzmaske	IP-Adressen
A	0/8 - 127/8	255.0.0.0	16777216
B	128/8 - 191/8	255.255.0.0	65536
C	192/8 - 223/8	255.255.255.0	256

Die IP-Adressen wurden ursprünglich von der IANA auch in diesen Blöcken vergeben. Dies führte zu einem sehr schnellen Verbrauch der verfügbaren IP-Adressen. Bereits 1990 hat Frank Solensky auf einem Treffen der Internet Engineering Task Force (IETF) vorhergesagt, dass bei dieser Vergabepolitik die IPv4-Adressen im Jahr 1994 verbraucht sein würden.[4]

1 Ich empfehle *Migrating to IPv6* von Marc Blanchet
2 http://mailman.nanog.org/pipermail/nanog/2011-February/032107.html
3 http://www.apnic.net/publications/news/2011/final-8
4 http://www.ietf.org/proceedings/prior29/IETF18.pdf

Um dieses Horrorszenario abzuwenden, wurden in den folgenden Jahren mehrere Maßnahmen entwickelt:

» **Entwicklung eines neuen Internet-Protokolls (IPv6)**
Das Protokoll konnte übrigens nicht IPv5 genannt werden, da diese Versionsnummer bereits durch das Stream Protocol 2 (ST2), ein experimentelles Protokoll für die Resource-Reservierung einer Internetverbindung belegt war. Das ST2-Protokoll hat sich nicht durchgesetzt.

» **Abschaffung der festen Klassen und Einführung des Classless Internet Domain Routing (CIDR) und der Variable Length Subnet Mask (VLSM)**
Hiermit ist es nun möglich, auch IP-Adressen der ehemaligen Klasse A, als /24-Netze an Organisationen zu vergeben. Somit kann ein Klasse-A-Netz auf 65.536 Organisationen aufgeteilt werden. Dies war vorher nicht möglich.

» **Einführung von Network Address Translation (NAT)**
Nun benötigen Anwender mit mehreren Hundert oder Tausend IP-Adressen nicht mehr entsprechend viele offizielle IP-Adressen. Es genügen wenige IP-Adressen, um das Netzwerk an das Internet anzubinden. Für die internen Netze wurden die privaten IP-Adressen eingeführt (RFC1918).

» **Einführung des Name-Based Virtual Hosting mit HTTP/1.1**
Das Web trieb das Wachstum des Internets maßgeblich voran. Das HTTP/1.0-Protokoll benötigte jedoch für jede Web-Domäne zwingend eine einzigartige IP-Adresse. Mit der Version 1.1 ist das Protokoll in der Lage, viele verschiedene Domänen unter Verwendung derselben IP-Adresse anzusprechen. Hierzu überträgt das Protokoll in einem eigenen Client-Header den Domänennamen.

Insbesondere die ersten beiden Maßnahmen waren sehr erfolgreich, sodass das IPv4-Protokoll noch über viele Jahre bis heute genutzt werden konnte. Dies war ursprünglich nicht so vorgesehen. Die Einführung von IPv6 sollte wesentlich schneller erfolgen, und die Maßnahmen sollten nur die Übergangszeit ein wenig verlängern.

Neben der Knappheit der IP-Adressen gab es jedoch auch andere Gründe, den Wechsel auf IPv6 voranzutreiben. Da die Local Registries nun sehr kleine Netze (/24) an die Organisationen und Unternehmen vergaben, stieg die Anzahl der Routen in den Internet-Routern stark an. Betrug die Anzahl der Routen 1995 etwa 20.000, so müssen heute die Internet-Router mehr als 300.000 Routen speichern und verwalten. Daher wurden für die Vergabe der IPv6-Adressen neue Richtlinien aufgestellt, die eine Aggregation der Routen auf Provider-Ebene erlauben sollen.

Ein weiterer Grund für die Einführung der IPv6-Adressen ist der Verzicht auf die Network Address Translation. Jedes System erhält eine eindeutige offizielle IP-Adresse und ist über diese Adresse ansprechbar. Dies erlaubt die Entwicklung neuer Protokolle, die eine derartige Ende-zu-Ende-Verbindung benötigen, und vereinfacht die Verwendung bereits vorhandener Protokolle wie VoIP, FTP, IPsec etc. Umständliche Umsetzungen von IP-Adressen an Gateways sind nicht mehr erforderlich.

CE[Y] Bitte auch die nicht angesprochenen Kapitel in dieser Einleitung erwähnen.

TS[2] Kapitel "Netzwerkgrundlagen" als Anhang gesetzt. Korrekt?

Die Verwendung privater IP-Adressbereiche erzeugte bei einer später erwünschten Kommunikation zwischen den Netzen häufig Probleme, da die zu verbindenden Netze meist gleiche IP-Adressen (z.B. 10.0.0.0/8) verwendeten. Hier mussten die Netze neu adressiert oder aufwendige NAT-Mechanismen genutzt werden.

All dies ist mit IPv6 nicht mehr erforderlich. In dieser Beziehung ist die zügige Einführung von IPv6 sicherlich zu wünschen. Jedoch wird auch IPv6 im Betrieb sicherlich noch die eine oder andere Schwäche zeigen. Auch wird es noch einige Zeit dauern, bevor das komplette Internet auf das IPv6-Protokoll umgestellt worden ist.

36.2 Überblick über die Änderungen und Neuerungen

IPv6 besitzt viele neue Funktionen und weist einige Änderungen gegenüber dem alten IPv4-Protokoll auf. Dieser Abschnitt listet die wesentlichen Änderungen auf und erläutert die Auswirkungen.

- **Größerer Adressraum**
  IPv6 verwendet 128 Bit lange IP-Adressen. Damit stehen auf lange Sicht ausreichend IP-Adressen zur Verfügung.
- **Vergabe der Adressen**
  Die IPv6-Adressen werden im Wesentlichen durch die Provider vergeben und nicht direkt durch die Local Registries. Dies erlaubt eine wesentlich bessere Aggregation der Routen und damit kleinere Routing-Tabellen in den Internet-Routern.
- **Keine VLSM**
  Der Verzicht auf variable Netzmasken verhindert Fehlkonfigurationen der Netzmaske.
- **Autokonfiguration**
  IPv6 ermöglicht es einem System, mithilfe von Router Advertisements selbstständig seine IP-Adresse zu ermitteln. DHCP ist nicht zwingend erforderlich.
- **Verzicht auf NAT**
- **Verzicht auf Broadcast**
  Broadcastpakete müssen immer von sämtlichen Systemen eines Netzes verarbeitet werden. Dies erzeugt unnötigen Overhead. IPv6 nutzt Multicast, um den Broadcast komplett zu ersetzen.
- **Ersatz des Address Resolution Protocols (ARP) durch das ICMP Neighbor Discovery Protocol (NDP)**
- **Mehrere IP-Adressen je Link (Netzwerkkarte)**
- **Einfachere IP-Header mit fester Länge**
  Dies soll vor allem die Behandlung und Filterung der Header durch Hardware vereinfachen.
- **Scope**
  Um die Reichweite von IP-Paketen zu beschränken, konnte bei IPv4 nur der TTL-Wert genutzt werden. IPv6 bietet neben dem Hop-Limit nun einen echten Scope an.

» **Einführung von privaten, aber einzigartigen IP-Adressen**
Die Verwendung von privaten IP-Adressen hat die spätere Kommunikation zwischen derartigen Netzen meist kompliziert. IPv6 kennt ebenfalls private IP-Adressen, die nicht für eine Kommunikation mit dem Internet genutzt werden können.[5] Diese sind jedoch mit großer Wahrscheinlichkeit einzigartig.

36.3 IPv6-Adressen und ihre Notation

36.3.1 Adressen

IPv6-Adressen sind, wie bereits erwähnt, 128 Bit und damit 16 Bytes lang. Damit sind sie im Vergleich mit IPv4-Adressen viermal länger. Insgesamt können so theoretisch 340.282.366.920.938.463.463.374.607.431.768.211.456 IP-Adressen verwendet werden.

Die IPv6-Adressen werden im Gegensatz zu den IPv4-Adressen hexadezimal in Blöcken zu je zwei Bytes geschrieben. Eine typische IPv6-Adresse sieht so aus:

```
2001:db8:735f:3137:1218:8bff:fed2:313a
```

Da derartige Zahlenmonster nur schwer zu lesen und zu schreiben sind, wurden in den letzten Jahren einige Vereinfachungen der Schreibweise genutzt, die nun mit dem RFC5952 auch verpflichtend definiert wurden. Diese sollen ausgehend von der IPv6-Adresse 2001:db8:0000:0000:0000:000f:0000:003a demonstriert werden

» Führende Nullen in einzelnen Blöcken müssen weggelassen werden. Dies führt zu folgender Schreibweise:

```
2001:db8:0:0:0:f:0:3a
```

» Die maximale Anzahl aufeinanderfolgender Blöcke von Nullen müssen durch `::` ersetzt werden:

```
2001:db8::f:0:3a
```

» Die Abkürzung `::` darf nicht für ein einzelnes Feld `:0000:` genutzt werden.

» Bei zwei gleich langen Abfolgen von Null-Blöcken wird das erste Vorkommen durch `::` ersetzt.

» Sämtliche Buchstaben müssen in Kleinbuchstaben geschrieben werden.

Sobald Sie die IPv6-Adresse in Kombination mit einem Port angeben möchten, wird die folgende Schreibweise genutzt. Die IPv6-Adresse wird in eckigen Klammern gefasst und der Port durch einen Doppelpunkt abgetrennt:

```
[2001:db8::f:0:3a]:80
```

5 Es gibt kein IPv6-NAT. Die Kommunikation ist dann nur über Router möglich.

Eine besondere IP-Adresse ist die Loopback-Adresse. Während das IPv4-Protokoll die IP-Adresse `127.0.0.1` für diesen Zweck nutzt, verwendet IPv6 die einfacher zu schreibende IPv6-Adresse `::1`. Testen Sie es direkt auf Ihrem Rechner: Funktioniert ein `ping6 ::1`?

```
$ ping6 -c1 ::1
PING ::1(::1) 56 data bytes
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.058 ms

--- ::1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.058/0.058/0.058/0.000 ms
```

Funktioniert der Ping, ist Ihr System IPv6-fähig.

36.3.2 Netzmaske

Neben der IP-Adresse ist aber auch die Netzmaske wichtig. Da es bei dem IPv4-Protokoll immer wieder zu Fehlkonfigurationen aufgrund der variabel konfigurierbaren Netzmaske (Variable Length Subnet Mask, VLSM) gekommen ist, verwendet IPv6 eine feste Netzmaske von 64 Bit. Diese kann offiziell auch nicht weiter in Subnetze unterteilt werden. Somit können IPv6-Netze maximal $2^{64} - 2$ (18.446.744.073.709.551.614) Rechner enthalten. Dies sollte auch für die größten Netze ausreichend sein.

INFO

Das bedeutet natürlich gleichzeitig, dass sehr verschwenderisch mit IPv6-Adressen umgegangen wird. Wenngleich ich oben (siehe Abschnitt 36.3.1) gesagt habe, dass es theoretisch 340.282.366.920.938.463.463.374.607.431.768.211.456 IPv6-Adressen gibt, können diese nur in maximal 2^{64} = 18.446.744.073.709.551.616 Netzen verwaltet werden. Da IPv6 bisher keine Network Address Translation (NAT) unterstützt[6], benötigt jedes System, das mit anderen Systemen im Internet direkt ohne Proxy kommunizieren möchte, eine einzigartige IPv6-Adresse. Ein DSL-Anwender, der über einen Router kommunizieren möchte, benötigt daher mindestens eine IPv6-Adresse für den Router und ein ganzes /64-IPv6-Netz für die Systeme hinter dem Router.

Dennoch sollten Sie nun nicht in Panik verfallen. Immerhin hatten wir bisher nur 2^{32}=4.294.967.296 IPv4-Adressen. An deren Stelle treten immerhin 2^{64} = 18.446.744.073.709.551.616 theoretisch mögliche IPv6-Netze.

Die feste Netzmaske von /64 und das hiermit erzeugte sehr große Netz erlauben jedoch neue Ansätze der Autokonfiguration, die ich in Abschnitt 36.6 erläutere. Im Wesentlichen kann ein Rechner seine IPv6-Adresse selbst aus der Adresse des /64-Netzes und seiner Hardware-MAC-Adresse bilden.

Um praktisch an dem IPv6-Internet (auch Internet6 genannt) teilzunehmen, benötigen Sie nun IPv6-Adressen. Diese erhalten Sie von Ihrem Provider oder können sie auch direkt vom RIPE (`http://www.ripe.net`) anfordern. Auch hier erhalten Sie vergleichsweise verschwenderisch IPv6-Adressen. Die kleinste Einheit, die das RIPE verteilt, ist ein /48-Präfix.

6 ... und dies wahrscheinlich auch so bleibt

Ein Beispiel:

```
$ whois 2001:67c:24::/48
[Querying whois.ripe.net]
[whois.ripe.net]

inet6num:       2001:67c:24::/48
netname:        SPE6-NET
descr:          OpenSource Training Ralf Spenneberg
country:        DE
org:            ORG-OTRS1-RIPE
admin-c:        RS9110-RIPE
tech-c:         RS9110-RIPE
status:         ASSIGNED PI
```

Bei einem /48-Präfix stehen Ihnen 16 Bits für die Verwaltung der /64-Netze zur Verfügung. Das bedeutet, dass Sie $2^{16} = 65.536$ Netze verwalten können. Dies sollte auch für größere Organisationen genügen. Erhalten Sie Ihren Zugang über einen Tunnel-Provider wie gogoNET[7] oder SiXXS[8], dann bekommen Sie meist ein /56- (8 Bits = 256 Netze) oder auch ein /48-Präfix.

Wie Sie die Netze aufteilen, bleibt Ihnen überlassen. Da mit derartigen Präfixen in kleinen Netzwerkumgebungen häufig mehr Netze als genutzte IP-Adressen zur Verfügung stehen, können die Netzkomponenten tatsächlich logisch in unterschiedlichen Netzen getrennt werden. So können Drucker, einzelne Server, Clients und Netzwerkkomponenten IPv6-Adressen in unterschiedlichen Netzen erhalten. Wenn die eingesetzten Netzwerk-Switches VLANs unterstützen, können diese Netze auch physikalisch in VLANs aufgetrennt werden. Router oder Multi-Layer-Switches können dann die Kommunikation zwischen diesen Netzen wiederherstellen, damit der Client auch tatsächlich drucken kann. Diese Segmentierung kann die Sicherheit des Netzes stark erhöhen, da nun auch Access Control Lists (ACLs) die zu routenden Pakete betrachten und verwerfen können. Als Router zwischen den IPv6-Netzen eignet sich daher auch sehr gut eine Linux-`ip6tables`-Firewall.

36.4 Adressarten und Scopes

IPv6 kennt drei Arten von Adressen. Außerdem kennt IPv6 nun den *Scope*, was sich am einfachsten als „Reichweite der IP-Adresse“ übersetzen lässt. Auch IPv4 kennt nun IP-Adressen mit unterschiedlichen Scopes. Dies wird jedoch nicht so intensiv genutzt wie bei IPv6.

7 http://gogonet.gogo6.com/
8 http://www.sixxs.net/

36.4.1 Adressarten

Beginnen wir mit den Adressarten. IPv6 kennt:

Unicast

Dies sind die üblichen IP-Adressen. Die IP-Adresse kennzeichnet genau ein Interface. Je Interface sind jedoch mehrere IP-Adressen erlaubt. Bei Verwendung einer Unicast-Adresse kommuniziere ich immer nur mit genau einem Host.

Multicast

Mithilfe dieser IP-Adressen kommunizieren Sie mit einer Rechnergruppe. Die Rechner der Gruppe müssen sich vorher für diese Gruppe registrieren. Die Verwendung des Multicast betrachte ich genauer in Abschnitt 36.5. Um dies zu testen können Sie einfach mal einen Ping an die Adresse `ff02::1` senden. Dies ist die All-Nodes-Multicast-Adresse. Alle IPv6-fähigen Rechner eines lokalen Netzes registrieren diese Multicast-Adresse. Bei dem Ping müssen Sie die Netzwerkkarte angeben, die verwendet werden soll:

```
$ ping6 -c1 -I eth0 ff02::1
PING ff02::1(ff02::1) from fe80::221:70ff:feb4:317 eth0: 56 data ↵
        bytes
64 bytes from fe80::221:70ff:feb4:317: icmp_seq=1 ttl=64 time ↵
        =0.113 ms
64 bytes from fe80::230:48ff:fe79:668c: icmp_seq=1 ttl=64 time ↵
        =1.18 ms (DUP!)
64 bytes from fe80::230:48ff:fe86:1f44: icmp_seq=1 ttl=64 time ↵
        =1.62 ms (DUP!)
64 bytes from fe80::5652:ff:fe0a:685f: icmp_seq=1 ttl=64 time=2.10 ↵
         ms (DUP!)
64 bytes from fe80::21e:bff:fea4:c5ff: icmp_seq=1 ttl=64 time=2.16 ↵
         ms (DUP!)
64 bytes from fe80::20f:1fff:fe8d:2905: icmp_seq=1 ttl=64 time ↵
        =2.57 ms (DUP!)
64 bytes from fe80::216:3eff:fe4d:9453: icmp_seq=1 ttl=64 time ↵
        =2.60 ms (DUP!)
64 bytes from fe80::216:3eff:fe1a:889: icmp_seq=1 ttl=64 time=2.62 ↵
         ms (DUP!)
64 bytes from fe80::216:3eff:fe2c:c18a: icmp_seq=1 ttl=64 time ↵
        =2.63 ms (DUP!)
64 bytes from fe80::225:90ff:fe0a:685e: icmp_seq=1 ttl=64 time ↵
        =2.86 ms (DUP!)
64 bytes from fe80::5652:ff:fe0a:685e: icmp_seq=1 ttl=64 time=3.02 ↵
         ms (DUP!)
64 bytes from fe80::216:3eff:fe6c:54ab: icmp_seq=1 ttl=64 time ↵
        =3.35 ms (DUP!)
```

```
64 bytes from fe80::5652:ff:fe0a:685d: icmp_seq=1 ttl=64 time=4.87 ↵
         ms (DUP!)
^C
--- ff02::1 ping statistics ---
1 packets transmitted, 1 received, 12 duplicates, 0rtt
        min/avg/max/mdev = 0.113/2.442/4.877/1.084 ms
```

Wie Sie erkennen können, antworten viele verschiedene Rechner auf den Ping. Die Antworten werden von Ping als Duplikat (DUP!) angezeigt. Alle Multicast-Adressen beginnen mit `ff00::/8`.

Anycast

Bei Verwendung dieser Adresse kommunizieren Sie mit einem Rechner aus einer Gruppe. Mehrere Systeme registrieren diese IP-Adresse. Die Router stellen fest, welcher Rechner Ihnen am nächsten ist, und routen die Pakete zu diesem einen System. Die Adressen können nicht von Unicast-Adressen unterschieden werden.

IPv6 unterstützt keine Broadcast-Adressen mehr, bei deren Verwendung alle Rechner eines Netzes antworten. Angriffe, die auf Broadcast-Protokollen oder -Paketen basieren, wie zum Beispiel Smurf, können bei IPv6 daher so nicht mehr angewendet werden.

Einige Adressen haben eine besondere Bedeutung. Diese sind in dem RFC5156 zusammengefasst und werden hier kurz aufgeführt:

Addressen	Bedeutung
::1/128	Loopback-Adresse
::/128	Unspezifizierte Adresse
::ffff:0:0/96	IPv4-gemappte Adressen (RFC4291)
::<ipv4>/96	IPv4-kompatible Adressen (RFC4291)
fe80::/10	Link-Local IP-Adressen
fc00::/7	Unique-Local IP-Adressen (RFC4193)
2001:db8::/32	Reserviert für Dokumenation (RFC3849)
2002::/16	6to4-Adressen (RFC3056)
2001::/32	Teredo
5f00::/8	ehemals 6bone (RFC1897), wieder frei
3ffe::/16	ehemals 6bone (RFC2471), wieder frei
2001:10::/28	Orchid (RFC4843)
ff00::/8	Multicast

Jeder Rechner muss, wenn er IPv6 nutzen möchte, die folgenden Adressen registrieren:

- eine Link-Local Adresse für jede IPv6-fähige Netzwerkkarte
- eine Loopback-Adresse (::1)
- All-Nodes-Multicast-Adressen für jede IPv6-fähige Netzwerkkarte (siehe Abschnitt 36.5)
- Solicited-Nodes-Multicast-Adressen für jede Unicast- und Anycast-Adresse (siehe Abschnitt 36.5)

36.4.2 Scope

Neben den drei Adressarten spielt bei IPv6 nun auch der Scope einer IP-Adresse eine Rolle.

Grundsätzlich unterscheidet IPv6 zwei Scopes: den Global Scope und den Local Scope. Global-Scope-IP-Adressen sind im gesamten Internet6 gültig und werden hier auch geroutet. Dies sind die klassischen offiziellen eindeutigen IP-Adressen, die Sie von einem Provider zugeteilt bekommen. Local-Scope-IP-Adressen beschränken sich auf einen bestimmten Netzbereich. Wie groß dieser Bereich ist, hängt wieder von dem Scope ab. Es gibt die folgenden Local Scopes:

Interface-Local

Dieser Scope wird auch Host-Scope genannt. Diese IP-Adresse kann nur für die Kommunikation auf der Netzwerkkarte genutzt werden. Die entsprechenden Pakete verlassen die Netzwerkkarte nicht. Die IP-Adresse `::1` auf dem Loopback Interface (`lo`) verwendet diesen Scope. Dies können Sie zum Beispiel mit dem Befehl `ip` testen:

```
$ ip -6 addr show dev lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
```

Link-Local

Dieser Scope beschränkt die Gültigkeit der Adressen auf das lokale Netz. Adressen mit diesem Scope dürfen von einem Router nicht geroutet werden. Es gibt Link-Local-Unicast- und -Multicast-Adressen. Die Link-Local-Unicast-Adressen beginnen mit `fe80::/64`. Jeder Rechner mit IPv6-Unterstützung erzeugt entsprechende Unicast-Link-Local-Adressen für seine Netzwerkkarten:

```
$ ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 fe80::221:70ff:feb4:317/64 scope link
       valid_lft forever preferred_lft forever
```

Admin-Local

Dieser Scope ist nur für Multicast-Adressen sinnvoll und wird dort genauer beschrieben. Die Grenzen der Admin-Local-Zone werden auf den Routern konfiguriert, und die Zone[CE] kann mehrere physikalische Netze enthalten.

Site-Local

Dieser Scope wurde ursprünglich für Unicast- und Multicast-Adressen spezifiziert. Die Site-Local-Unicast-Adressen (`fec0::10`) wurden nach Anweisung des RFC3879[CE a] in den Deprecated-Zustand versetzt und sollten heute nicht mehr verwendet werden. Sie wurden durch die Unique-Local-Unicast-Adressen (`fd00::/8` und `fc00::/8`, RFC4193) abgelöst. Diese haben jedoch einen globalen Scope! Heute ist der Site-Local-Scope nur noch für Multicast-Adressen interessant. Die Definition der Site-Zone erfolgt administrativ auf den Routern.

CE Bitte diese Ergänzung bestätigen.

CE a Bitte prüfen, ob die neue Formulierung deutlicher ist, als die ursprüngliche: „mit dem RFC3879“.

Organization-Local

Dieser Scope ist Multicast-Adressen vorbehalten. Die Definition der Organization-Zone erfolgt administrativ auf den Routern.

INFO

Auch IPv4 kennt Link-Local-Adressen. Diese wurden im RFC3927 vor allem auf Betreiben von Microsoft und Apple definiert. Es handelt sich um den Adressraum 169.255.0.0/16, der von Systemen genutzt werden darf, wenn Sie per DHCP eine IP-Adresse anfordern, aber kein DHCP-Server zur Verfügung steht. Um dennoch eine Kommunikation über Broadcast oder Multicast in derartigen Netzen zu erlauben, verwenden die Systeme eine zufällige IP-Adresse aus diesem Bereich.

Die Scopes erlauben die Verwendung von IP-Adressen, die nur in bestimmten Bereichen erlaubt sind. Das ermöglicht komplett neue Anwendungen. Falls ein bestimmter Dienst nicht über Router erreichbar sein soll, genügt es, wenn sich dieser Dienst auf eine Link-Local-Adresse bindet. Die Adresse kann nur in dem lokalen Netz erreicht werden und ist auch hier nur eindeutig. Microsoft nutzt dies zum Beispiel ab Windows 7 für die Freigaben in dem Heimnetzwerk.

Besonders die Multicast-Adressen nutzen diese Funktion sehr stark (siehe nächster Abschnitt). So ist es zum Beispiel möglich, alle NTP-Server einer Site mit einem Multicast-Paket zu adressieren.

36.5 IPv6-Multicast

Die Multicast-Adressen und die Multicast-Kommunikation spielen eine wesentliche Rolle für die IPv6-Funktionalität. Hierbei verbessert Multicast die Leistungsfähigkeit gegenüber der Broadcast-Kommunikation bei IPv4 wesentlich. Mithilfe von Multicast-Adressen wählt der Absender genau die Gruppe der Systeme aus, mit denen er kommunizieren möchte. Damit eine entsprechende Gruppe ausgewählt werden kann, unterstützt Multicast viele verschiedene Scopes (siehe auch Abschnitt 36.4.2):

- Host
- Link
- Admin
- Site
- Organization
- Global

Multicast-Adressen beginnen immer mit dem Byte `ff`. Das anschließende Byte teilt sich auf in zwei Nibbles zu je vier Bits. Das erste Nibble definiert, ob es sich um eine well-known Multicast-Adresse oder um eine temporäre benutzerdefinierte Adresse handelt. Das zweite Nibble definiert den Scope (siehe Abbildung 36.1).

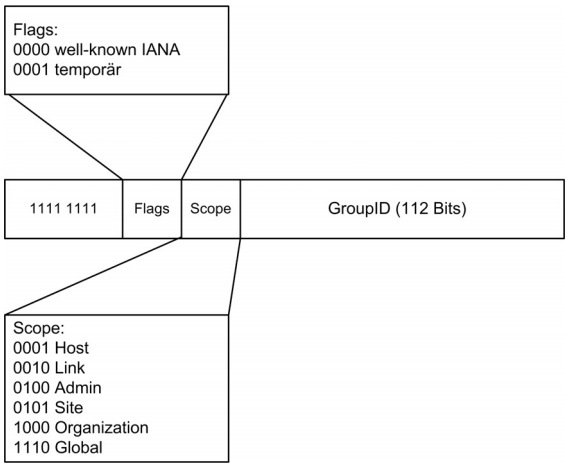


Abbildung 36.1: **Multicast-Adressen haben immer denselben Aufbau.**

Das IANA hat einige well-known Multicast-Adressen definiert. Die aktuellste Aufstellung finden Sie auf der Webseite der IANA.[9] Die wichtigsten möchte ich hier aufzählen:

Adresse	Gruppe
ff02::1	Alle Rechner des lokalen Netzes
ff02::2	Alle Router im lokalen Netz
ff02::5	Alle OSPFv3 SPF-Router im lokalen Netz
ff02::6	Alle OSPFv3 DR-Router im lokalen Netz
ff02::9	Alle RIPng-Router im lokalen Netz
ff02::A	Alle EIGRP-Router im lokalen Netz
ff02::1:2	Alle DHCP-Server und Relays im lokalen Netz
ff05::1:3	Alle DHCP-Server der Site
ff0x::fb[10]	Alle Multicast-DNS-Server
ff0x::101	Alle NTP-Server

36.6 Autokonfiguration

Eine der wesentlichen neuen Funktionen des IPv6-Protokolls ist die Autokonfiguration der IP-Adresse, die es jedem Host ermöglicht, sich selbst eine gültige IPv6-Adresse ohne einen DHCP-Server zuzuweisen. Voraussetzung für diese Funktion ist lediglich ein IPv6-Router, der per ICMPv6 Router-Advertisements versendet.

Natürlich ist es neben der Autokonfiguration auch möglich, statische IP-Adressen zu verwenden. Zusätzlich kann auch DHCP verwendet werden. DHCP kann sogar Vorteile gegenüber der

9 `http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml`

10 Das `x` wird durch das Scope-Bit ersetzt.

Autokonfiguration aufweisen. Dies wird in Abschnitt 36.6.1 genauer besprochen. Die Autokonfiguration kann auch abgeschaltet werden (siehe Abschnitt 23.3.12).

Die Autokonfiguration wird in RFC2462 beschrieben. Hiermit können die Rechner ihre IP-Adressen selbst konfigurieren, sobald sie mit einem Netz verbunden sind. Der Vorgang erfolgt in mehreren Schritten.

Zunächst ermitteln die Systeme für jede verbundene Netzwerkkarte den Interface Identifier. Dieser wird häufig auch EUI-64 Identifier genannt. Hierzu nimmt der Host die MAC-Adresse (48 Bits) und fügt die 16 Bits `fffe` in der Mitte ein. Anschließend wird das Bit 7 des höchstwertigsten Bytes gesetzt (siehe Abbildung 36.2).

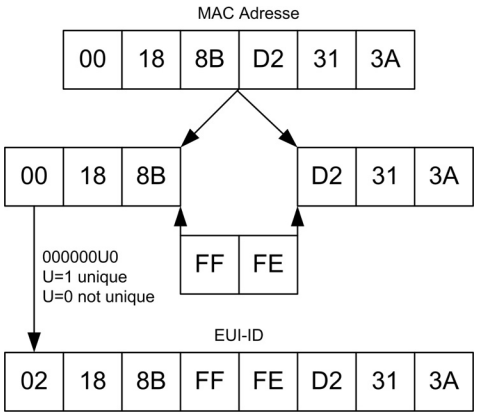


Abbildung 36.2: **Die Interface ID wird aus der MAC-Adresse gebildet.**

Mithilfe dieser ID erzeugt der Rechner eine Link-Local-Adresse. Hierzu hängt er den Interface Identifier an das Präfix `fe80/10` an (siehe Abbildung 36.3).

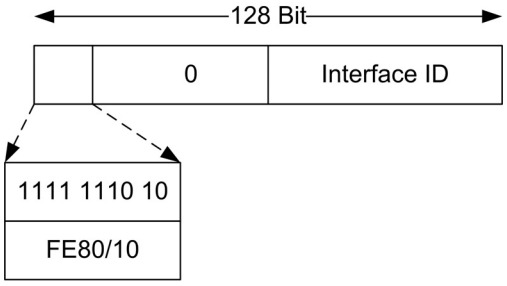


Abbildung 36.3: **Die Link-Local-Adresse verwendet den Interface Identifier.**

Da die MAC-Adresse eine im lokalen Netzwerk eindeutige Adresse sein muss, damit eine Kommunikation möglich ist, sollte auch die Link-Local-Adresse eindeutig sein. Dennoch führt der IPv6-Host eine Duplicate-Address-Detection (DAD) durch. Hierzu sucht er im Netz nach einem Rechner, der bereits diese IPv6-Adresse verwendet. Dies erfolgt mit dem Neighbor Discovery Protocol (NDP), das in Abschnitt 36.7 genauer betrachtet wird. Als eigene IP-Adresse

verwendet er in diesem Schritt die unspezifizierte IP-Adresse ::/128. Wird die IP-Adresse noch nicht verwendet, registriert der Rechner die IP-Adresse für die entsprechende Netzwerkkarte. Mit tcpdump kann die DAD verfolgt werden:

```
# tcpdump -vv
16:31:57.856122 IP6 (hlim 255, next-header ICMPv6 (58) payload length:  ↵
        24) :: > ff02::1:ff00:103: [icmp6 sum ok] ICMP6, neighbor  ↵
        solicitation, length 24, who has fe80::5650:75ff:fe00:103
```

Mit dieser Link-Local-IPv6-Adresse kann der Rechner nun mit anderen Rechnern im lokalen Netz kommunizieren. Eine Kommunikation mit dem Internet6 ist noch nicht möglich, da der Rechner noch nicht über eine globale IPv6-Adresse verfügt. Um eine derartige IPv6-Adresse zu erhalten, sendet der Rechner nun Router-Solicitation-Nachrichten aus. Hierbei handelt es sich um ICMPv6-Nachrichten.

```
16:31:58.856199 IP6 (hlim 255, next-header ICMPv6 (58) payload length:  ↵
        16) fe80::5650:75ff:fe00:103 > ff02::2: [icmp6 sum ok] ICMP6,  ↵
        router solicitation, length 16
          source link-address option (1), length 8 (1): 54:50:75:00:01:03
            0x0000:  5450 7500 0103
```

Ist ein Router im Netz verfügbar, so antwortet dieser mit einem Router Advertisement. Das Router Advertisement enthält die IP-Adresse des Routers und das Präfix des lokalen Netzes. Seit RFC5006 kann das Router Advertisement auch rekursive DNS-Server (RDNSS) enthalten. Jedoch unterstützen noch nicht alle Router und Clients die entsprechenden Optionen.

```
16:31:58.857034 IP6 (hlim 255, next-header ICMPv6 (58) payload length:  ↵
        80) fe80::5650:75ff:fe00:101 > ff02::1: [icmp6 sum ok] ICMP6,  ↵
        router advertisement, length 80
        hop limit 64, Flags [none], pref medium, router lifetime 1800s,  ↵
                reachable time 0s, retrans time 0s
          prefix info option (3), length 32 (4): 2001:db8::/64, Flags [ ↵
                  onlink, auto], valid time 86400s, pref. time 14400s
            0x0000:  40c0 0001 5180 0000 3840 0000 0000 2001
            0x0010:  0db8 0000 0000 0000 0000 0000 0000 0000
          rdnss option (25), length 24 (3):  lifetime 600s, addr: fe80 ↵
                  ::5650:75ff:fe00:101
            0x0000:  8000 0000 0258 fe80 0000 0000 0000 5650
            0x0010:  75ff fe00 0101
          source link-address option (1), length 8 (1): 54:50:75:00:01:01
            0x0000:  5450 7500 0101
```

Basierend auf dem Präfix, das der Client von dem Router erhalten hat, erzeugt er nun mit dem Interface Identifier eine weitere IP-Adresse. Anschließend prüft er mithilfe der Duplicate Address Detection erneut, ob diese IP-Adresse möglicherweise bereits verwendet wird:

```
16:31:59.628024 IP6 (hlim 255, next-header ICMPv6 (58) payload length:
        24) :: > ff02::1:ff00:103: [icmp6 sum ok] ICMP6, neighbor
        solicitation, length 24, who has 2001:db8::5650:75ff:fe00:103
```

Erhält er hier keine Antwort, registriert der Client die IP-Adresse auf der Netzwerkkarte.

Ein Router kann mehrere Präfixe gleichzeitig verteilen. Jedes Präfix wird mit zwei Lebensdauern verteilt:

Valid Lifetime: Nach Ablauf dieser Lebensdauer darf der Client die IP-Adresse nicht mehr nutzen.

Preferred Lifetime: Nach Ablauf dieser Lebensdauer darf der Client keine neuen Verbindungen mit dieser IP-Adresse aufbauen. Vorhandene Verbindungen laufen weiter.

Es können auch mehrere Router gleichzeitig antworten. Der Client konfiguriert für sämtliche erhaltene Präfixe entsprechende IP-Adressen. Diese werden mit der in dem Router Advertisement enthaltenen Lebensdauer konfiguriert. Dies können Sie auf dem Client auch mit dem Befehl `ip` kontrollieren:

```
# ip -6 addr show dev eth0
4: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8::5650:75ff:fe00:103/64 scope global dynamic
       valid_lft 86394sec preferred_lft 14394sec
    inet6 fe80::5650:75ff:fe00:103/64 scope link
       valid_lft forever preferred_lft forever
```

36.6.1 DHCP

Die ursprüngliche Autokonfiguration nach RFC2462 erlaubte nur die automatische Konfiguration der IP-Adresse und Route durch den Client. Mit dem RFC5006 wurden Optionen für die Konfiguration von DNS-Servern hinzugefügt. Nicht alle Clients und Router können mit diesen Optionen umgehen, und viele Clients benötigen weitere Informationen wie zum Beispiel die zu verwendenen NTP-Server.

Hierfür kann DHCPv6 nach RFC3315 verwendet werden. Ursprünglich sollte IPv6 komplett auf DHCP verzichten können. Jedoch hat DHCP bei IPv4 für die automatische Konfiguration zusätzlicher Informationen neben der IP-Adresse einen derart hohen Stellenwert erhalten, dass auch bei IPv6 diese Möglichkeiten wünschenswert sind.

Es gibt zwei Varianten, wie IPv6 DHCP nutzen kann:

Stateless: Hier nutzt der Client die Autokonfiguration wie in Abschnitt 36.6 beschrieben und verwendet DHCP nur, um zusätzliche Informationen, wie DNS-Server, Domäne, NTP-Server etc., anzufordern.

Stateful: Hier wird DHCP genauso genutzt wie bei IPv4. Der Client fordert auch seine IPv6-Adresse per DHCP an. Der Server muss daher Listen mit den bereits vergebenen IPv6-Adressen pflegen und speichern.

Stateless DHCP hat den Vorteil, dass der DHCP-Server keine Informationen speichern muss. Ein Neustart ist unproblematisch, und es können auch ohne Probleme mehrere DHCP-Server in einem Netz aufgesetzt werden, die sich gegenseitig ersetzen können. Stateful DHCP erfordert die Synchronisation dieser DHCP-Server, damit nicht zwei DHCP-Server unabhängig voneinander identische IP-Adressen an unterschiedliche Clients verteilen.

Unter Linux unterstützt neben vielen anderen Implementierungen ISC-DHCP ab der Version 4.1 DHCPv6. Hierbei werden sowohl stateless als auch stateful DHCPv6 unterstützt. Ob DHCPv6 verwendet wird, hängt hier von der Konfiguration ab.

Microsoft Windows unterstützt ab Vista DHCPv6. Das genaue Verhalten hängt hier von dem Router Advertisement ab. Das Router Advertisement kann zwei Flags enthalten: `M` (Managed-Flag) und `O` (OtherConfigFlag). Sind beide Flags gesetzt, so verwendet Windows den Stateful-Modus. Ist nur das `O`-Flag gesetzt, so verwendet der DHCPv6-Client den Stateless-Modus.

Stateful DHCPv6 bietet gegenüber der Autokonfiguration einige Vorteile. Daher wird es recht häufig eingesetzt:

» Der DHCPv6-Server kann die IP-Adressen protokollieren. Es ist später möglich, über die IP-Adresse auf die MAC-Adresse und damit auf das System zurückzuschließen und zu erkennen, wann das System online war.

» Der DHCPv6-Server kann den Namen des Clients in einem DNS-Server eintragen (dynamisches DNS). Der Client kann anschließend direkt über seinen DNS-Namen angesprochen werden.

36.7 Neighbor Discovery

Das Address Resolution Protocol (ARP) wird von IPv6 nicht mehr genutzt. Stattdessen nutzt IPv6 das ICMPv6-Protokoll um die Hardware-Adresse weiterer Systeme im lokalen Netz zu ermitteln. Das Protokoll heißt nun Neighbor Discovery Protocol (NDP).

INFO

Die wesentliche Neuerung ist jedoch nicht der Wechsel des Protokolls, sondern die Nutzung von Multicast-Adressen für die Funktion. Während ARP alle Rechner eines Netzes nach der MAC-Adresse für eine IPv4-Adresse fragt, schränkt NDP diese Frage bereits auf eine Multicast-Gruppe ein. In den meisten Fällen befindet sich in der entsprechenden Gruppe nur der tatsächlich betroffene Rechner.

Das Neighbor Discovery Protocol verwendet spezielle Multicast-Adressen, um die zu den IPv6 gehörenden Hardware-Adressen zu ermitteln. Es handelt sich um die Solicitated-Node-Multicast-Adresse.

Möchte ein Host die Hardware-Adresse zu einer Unicast-IPv6-Adresse ermitteln, so entnimmt er dieser die niedrigen 24 Bit. Diese fügt er in die Solictated-Node-Multicast-Adresse `ff02::01:ff00:0000/104` ein (siehe Abbildung 36.4).

Jeder Client muss sich entsprechend seiner IP-Adresse für die korrekte Multicast-Gruppe registriert haben. Dies erfolgt direkt nach der Konfiguration der IP-Adressen. Sie können dies auch mit dem Befehl `ip` prüfen. Hier hat Rechner eine bestimmte IPv6-Adresse:

```
# ip -6 addr show dev eth1
7: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 fe80::5650:75ff:fe00:101/64 scope link
       valid_lft forever preferred_lft forever
```

Dann registriert er sich auch für die entsprechende Multicast-Gruppe:

```
# ip -6 maddr show dev eth1
7:      eth1
        inet6 ff02::1:ff00:0
        inet6 ff02::2 users 2
        inet6 ff02::1:ff00:101
        inet6 ff02::1
```

Da üblicherweise die IPv6-Adressen durch Autokonfiguration basierend auf der MAC-Adresse erzeugt werden und die letzten drei Bytes dieser MAC-Adresse meist variieren, befindet sich in der entsprechenden Gruppe meist nur ein Zielsystem, das die entsprechende *neighbor-solicitation*-ICMPv6-Nachricht verarbeiten muss.

INFO

Die ersten drei Bytes der MAC-Adresse sind häufig identisch, da sie den Hersteller kennzeichnen. Wenn Sie mehrere Geräte desselben Herstellers verwenden, unterscheidet sich die MAC-Adresse nur in den letzten drei Bytes.

Alle Rechner in der Multicast-Gruppe verarbeiten die Neighbor-Solicitation-ICMPv6-Nachricht. Der Host mit der gefragten IPv6-Adresse antwortet und sendet seine MAC-Adresse in einer ICMPv6-*neighbor-advertisement*-Nachricht.

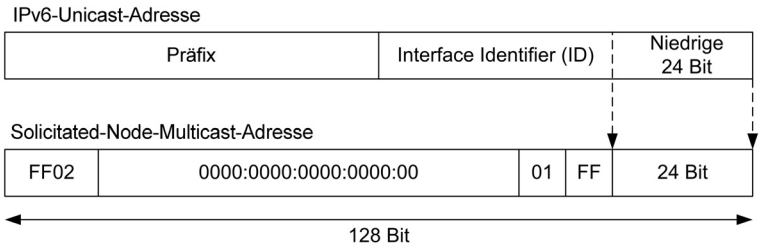


Abbildung 36.4: **Die Solicitated-Node-Multicast-Adresse enthält 24 Bit der gewünschten IP-Adresse.**

CE Bezug? (Welches Protokoll?)

STOP

Da dieses Protokoll das ICMPv6-Protokoll auf der Schicht drei nutzt, darf eine Firewall nicht einfach sämtliche ICMPv6-Pakete in der `INPUT`- und `OUTPUT`-Kette verwerfen. Bei IPv4 war dies ohne Weiteres möglich, da das ARP-Protokoll auf der Schicht Zwei nicht von `iptables`-Regeln betroffen war. Bei IPv6 würde das dazu führen, dass die Firewall selbst nicht mehr von den Systemen, auch nicht als Gateway, erreichbar wäre!

Bestandteil des Neighbor Discovery Protocol sind auch die *router-solicication*- und *router-advertisement*-Nachrichten. Diese wurden aber bereits in Abschnitt 36.6 beschrieben und werden daher hier nicht wiederholt.

36.8 Privacy Extensions

Die Autokonfiguration nutzt, wie in Abschnitt 36.6 beschrieben, die MAC-Adresse des Systems, um mithilfe der Router-Advertisements global gültige IPv6-Adressen zu erzeugen. Da die MAC-Adressen aktuell einzigartig sind, ist darüber eine Client-Erkennung möglich. Auch wenn ein Client sich in unterschiedlichen IPv6-Netzen bewegt, bleibt sein Interface Identifier immer gleich und einzigartig. Da die IP-Adressen nicht mehr durch NAT ausgetauscht werden, ist ein Tracking des Systems ohne Weiteres möglich. Führte bisher ein Reconnect Ihres DSL-Routers immer zu neuen IPv4-Adressen, erhalten Sie bei IPv6 zwar jedes Mal auch ein anderes IPv6-Netz, jedoch bleibt der Identifier Ihrer Clients immer identisch und damit nachvollziehbar.

Mit dem RFC4941 wurde die Möglichkeit geschaffen, dass Hosts zusätzlich zu Ihrer auf der MAC-Adresse basierenden IPv6-Adresse zusätzliche temporäre IPv6-Adressen erzeugen, die sie bevorzugt für Verbindungen einsetzen. Diese Erweiterungen werden als Privacy Extensions bezeichnet, da sie die Privatsphäre schützen sollen.

Die folgenden Betriebssysteme unterstützen bereits die Privacy Extension:

OS	Default
Windows XP	ein
Windows Vista	ein
Windows 7	ein
Windows Server 2003	aus
Windows Server 2008 R2	aus
Linux 2.6	aus
Mac OS X	aus

Bei allen Betriebssystemen lässt sich die Privacy Extension ein- bzw. ausschalten. Bei Windows erzeugen die Systeme sogar ihre statische (Haupt-)IPv6-Adresse mit einem zufälligen Identifier. Um diese Funktion unter Windows abzuschalten, verwenden Sie:

```
netsh interface ipv6 set global randomizeidentifiers=disabled
```

Um die Privacy Extensions komplett inklusive der temporären IPv6-Adressen abzuschalten, nutzen Sie:

```
netsh interface ipv6 set privacy state=disabled
```

Unter Linux konfigurieren Sie die temporären Adressen über die Proc-Schnittstelle (siehe auch Abschnitt 23.3.12) mit `use_tempaddr`.

Auch unter MacOSX werden die temporären Adressen mit `sysctl` verwaltet:

```
sysctl -w net.inet6.ip6.use_tempaddr=1
```

INFO

Ich empfehle in geschlossenen Netzen immer die Abschaltung dieser Funktion, da ansonsten die Fehlersuche basierend auf IP-Adressen fast unmöglich wird. Da die Clients zufällige IPv6-Adressen nutzen und deren Verwendung nicht protokollieren, können Sie später kaum feststellen, welcher Client welche IP-Adresse genutzt hat. Die Quelle eines Virusausbruchs kann so kaum zurückverfolgt werden.

37. Transitionsmechanismen

Ein großes Problem bei der Einführung von IPv6 ist die fehlende Kompatibilität mit IPv4. Hosts, die nur das IPv4-Protokoll verwenden, können nicht mit Hosts kommunizieren, die nur das IPv6-Protokoll unterstützen und umgekehrt. Da nun die IPv4-Adressen aufgebraucht sind bzw. in den nächsten Monaten aufgebraucht sein werden, können neue Systeme nur mithilfe von IPv6-Adressen angebunden werden. Diese können dann aber mit den IPv4-Hosts nicht kommunizieren. Die Wahrscheinlichkeit, dass alle IPv4-Hosts zusätzlich mit IPv6-Adressen ausgestattet werden, ist sehr gering, da ein Großteil der aktuell im Einsatz befindlichen Systeme kein IPv6 unterstützt. So ist nur ein Bruchteil der aktuell eingesetzten DSL-Router oder Netzwerkdrucker, um zwei Beispiele zu nennen, IPv6-fähig.

Aus diesem Grunde werden während des Übergangs für die Kommunikation Transitionsmechanismen benötigt, die zwischen den beiden Protokollen vermitteln können.

Der große Wurf sollte mit NAT-PT (Network Address Translation and Protocol Translation) und NAPT-PT (Network Address and Port Translation and Protocol Translation) gelingen. Hierbei handelt es sich um eine tatsächliche Protokollumsetzung, die in dem RFC2766 beschrieben wird. Eine Implementierung für Linux ist verfügbar.[1] Mithilfe von DNS-Application-Layer-Gateways wird eine bidirektionale Verbindung ermöglicht.

Da sich jedoch die Meinung durchgesetzt hat, dass NAT-PT und NAPT-PT die Entwicklung und Verwendung von IPv6 behindern, wurden diese Mechanismen durch den RFC4966 im Juli 2007 in den Deprecated- bzw. Historic-Zustand versetzt. Der Grund für diese Überlegungen waren die Vorausetzungen, um bei NAT-PT und NAPT-PT auch Verbindungen von IPv4-Hosts mit IPv6-Hosts zu ermöglichen.

Die häufigste Implementierung dieser Protokollumsetzung ist die Transport Relay Transition (TRT), die im RFC3142 genauer spezifiziert wird. Entsprechende Implementierungen gibt es auch für Linux.[2] TRT ist nicht von dem RFC4966 betroffen, da der Transport Relay Translator nur Verbindungen von IPv6-Hosts zu IPv4-Hosts umsetzt. Dabei arbeitet der TRT als transparenter Proxy auf TCP-Ebene.

Dennoch wird dieses Verfahren heute weniger eingesetzt und ist im Wesentlichen von Dualstack-Lite und NAT64 abgelöst worden. NAT64 wird in Abschnitt 37.1 besprochen. Bei Dual-

1 http://naptpt.sourceforge.net/
2 pTRTd: http://www.litech.org/ptrtd/

stack-Lite handelt es sich um eine spezielle Form eines Dualstack-Hosts. Als Dualstack bezeichnet man einen Host, der sowohl das IPv4- als auch das IPv6-Protokoll unterstützt. Verfügt der Host über offizielle IP-Adressen für beide Protokolle, so kann er ohne Probleme mit Systemen im Internet4 und Internet6 kommunizieren.

Da bald auch für die Internet Service Provider die IPv4-Adressen knapp werden, wurde Dualstack-Lite entwickelt. Hierbei erhält der Endanwender von seinem DSL-Provider zwei IP-Adressen:

» eine offiziell gültige globale IPv6-Adresse und ein globales IPv6-Präfix für Systeme hinter seinem DSL-Router und

» eine private IPv4-Adresse

Intern verteilt der DSL-Router des Endanwenders das globale IPv6-Präfix und private IP-Adressen, die sich nicht mit den vom Provider verwendeten IP-Adressen überschneiden.

Greift der Endanwender auf das Internet6 zu, so kann dies direkt mit den globalen IPv6-Adressen erfolgen. Möchte der Endanwender auf das Internet4 zugreifen, so nutzen die internen Hosts ihre privaten IP-Adressen. Diese werden von dem DSL-Router des Endanwenders auf die von dem Provider erhaltene private IP-Adresse genattet. Der Dualstack-Lite nimmt dann die IPv4-Pakete und verschickt diese durch einen Tunnel an ein CGN-Gerät bei dem Provider (siehe Abbildung 37.1). Bei dem Carrier-Grade-NAT-(CGN-)Device handelt es sich um ein besonders leistungsfähiges NAT-Gerät, das die Anfragen sämtlicher Endanwender wieder auf wenige global gültige IPv4-Adressen nattet. Dies ermöglicht den Zugriff auf das IPv4-Internet.

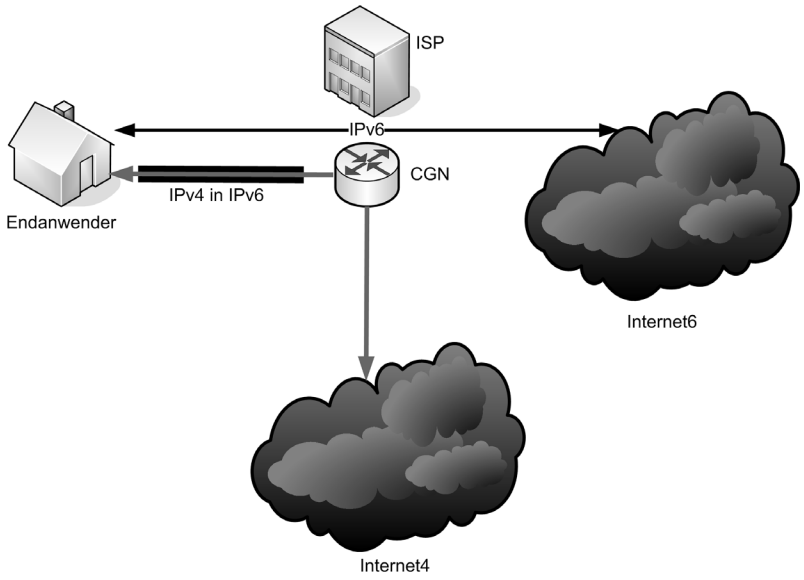


Abbildung 37.1: **Dualstack-Lite nutzt Carrier-Grade-NAT (CGN) um private IP-Adressen in globale IPv4-Adressen umzusetzen.**

Ein weiteres Problem beim Einsatz von IPv6 ist die Kommunikation von reinen IPv6-Inseln über ein IPv4-Internet. Hierfür wurden verschiedenste Tunnel-Mechanismen entwickelt. So gibt es mit 6in4, 6over4, ISATAP und 6to4 mehrere Mechanismen, die entsprechende Tunnel aufbauen können. Während die ersten drei Verfahren spezielle konfigurierte Gegenstellen verlangen, ist 6to4 ein Verfahren, das keine speziell konfigurierte Gegenstelle benötigt. Es kommuniziert mit im Internet verfügbaren Systemen, die sämtliche Tunnel annehmen. Diese Systeme sind über IPv4-Anycast-Adressen erreichbar.[3]

Da diese Tunnel nicht über genattete Verbindungen funktionieren, wurden mit AYIYA und Teredo entsprechende Verfahren für Tunnel zur Verfügung gestellt.

Diese Verfahren werden in den nächsten Jahren jedoch wegfallen, da hoffentlich immer mehr Provider nativ IPv6 anbieten werden. Dann sind entsprechende Tunnel für die Anbindung von IPv6-Inseln über das IPv4-Netz nicht mehr erforderlich.

37.1 NAT64

Das heute vielversprechendste Verfahren für die Anbindung von IPv6-Clients an IPv4 ist das NAT64. Dieses benötigt auch ein DNS-ALG, das als DNS64 bezeichnet wird. NAT64 ist im März 2011 nach vielen IETF-Drafts als RFC6146 veröffentlicht worden, und DNS64 erschien als RFC6147. Es existieren Implementierungen für die verschiedensten Betriebssysteme. Für Linux existieren mit Tayga[4] und Ecdysis[5] sowohl eine Userspace- als auch eine Kernelspace-Implementierung. Tayga nutzt den TUN-Treiber, um die Pakete zu empfangen und senden, während Ecdysis ein Netfilter-Modul verwendet. Praktische Tipps für den Einsatz von Ecdysis finden Sie in Kapitel 38. Mit Bind 9.8.0 ist auch der Bind als DNS64-Gateway nutzbar. Patches sind nicht mehr erforderlich.

Bei NAT64 werden IPv6-Adressen in IPv4-Adressen genattet. Hierzu fragt der Client zunächst einen DNS64-Server nach der entsprechenden IP-Adresse für eine DNS-Domäne. Anschließend verbindet er sich über ein NAT64-Gateway mit dem Server im IPv4-Internet (siehe Abbildung 37.2).

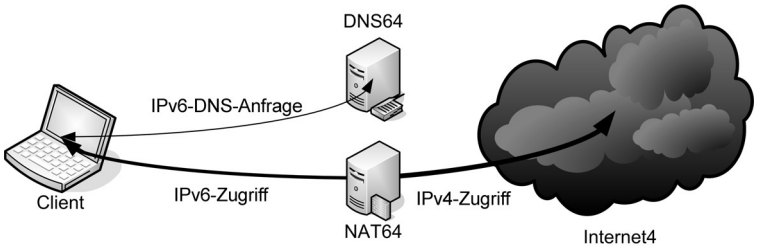


Abbildung 37.2: **Das NAT64 ermöglicht gemeinsam mit DNS64 die Kommunikation von IPv6-Clients mit IPv4-Servern.**

3 Auch IPv4 kennt bereits Anycast-Adressen!
4 http://www.litech.org/tayga/
5 http://ecdysis.viagenie.ca/

Wie funktioniert dies genau? Abbildung 37.3 zeigt das exakte Verfahren bei DNS64. Zunächst fragt der IPv6-Client den DNS64-Server nach der IPv6-Adresse (Quad-A Record, oder auch AAAA Record) für eine Domäne. Da er eine IPv4-Adresse nicht nutzen kann, schließt der Client diese in seine Anfrage nicht ein. Der DNS64-Server als rekursiver DNS-Server leitet die Anfrage weiter. Erhält er keinen Quad-A-Record zurück, so stellt der rekursive DNS-Server die Anfrage erneut und bittet um eine IPv4-Adresse (A-Record). Erhält er nun eine Antwort, so erzeugt er eine IPv6-Adresse, indem er die IPv4-Adresse in ein vorher definiertes /96-Präfix einbettet. Meist wird das Well Known Prefix (WKP) `64:ff9b::/96` genutzt. Es kann jedoch auch jedes andere Präfix genutzt werden. Dann spricht man von einem Network Specific Prefix (NSP). Das RFC6052 gibt weitere Hinweise.

Das NAT64-Gateway kennt ebenfalls das Präfix `64:ff9b::/96`. Sobald ein Paket eine IP-Adresse in diesem Präfix anspricht, extrahiert das NAT64-Gateway die IPv4-Adresse aus den niedrigsten 32 Bit, erzeugt einen NAT-Eintrag und tauscht die IP-Header und -Adressen in dem Paket gegen ihre IPv4-Pendants aus.

STOP

Problematisch ist DNS64 bei gleichzeitiger Nutzung von DNSSEC durch den Client. Meist wird das DNSSEC jedoch nicht durch den Client umgesetzt. Jedoch sind mit Fedora 15 erste Linux-Clients verfügbar, die DNSSEC selbst umsetzen. Da DNS64 die Antworten modifiziert, verwirft der Client dann die Antworten.

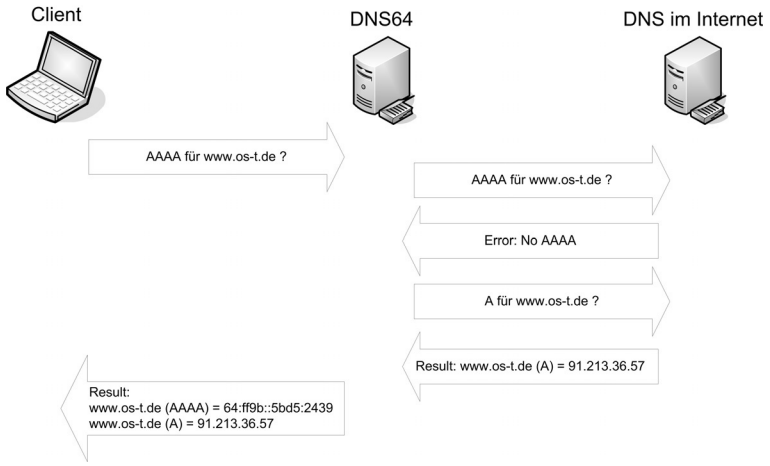


Abbildung 37.3: **DNS64 erzeugt aus IPv4-Adressen neue IPv6-Adressen mit einem bekannten Präfix.**

38. Kleines IPv6-Howto

Dieses Kapitel gibt Ihnen ein paar Hinweise, um schnell und einfach eine Verbindung zum IPv6-Internet herzustellen. Dabei gehe ich davon aus, dass Sie im Wesentlichen Linux-Systeme zur Verfügung haben. Auf der CD ist auch eine kleine virtuelle
Umgebung, mit der Sie das testen können. Diese enthält eine Linux-Firewall und den Client, der für viele weitere Szenarien auch verwendet wird.

Zunächst benötigen Sie eine Anbindung an das Internet6. Ich gehe davon aus, dass Ihr Provider dies bisher nicht unterstützt. Ich werde Ihnen zwei Wege über Tunnelbroker aufzeigen: Freenet6 und SixxS. Anschließend müssen Sie die von dem Tunnelbroker erhaltenen IPv6-Adressen an interne Clients verteilen. Hierzu nutzen Sie den Router Advertisement Daemon `radvd`. Um auch zusätzliche Informationen zu verteilen, werden wir auch DHCPv6 nutzen. Um später mit reinen IPv6-Clients auf IPv4-Systeme zuzugreifen, setzen wir auch noch NAT64 mit DNS64 in diesem Kapitel um.

STOP

Achtung: Ich habe auf den verschiedensten Distributionen zu Beginn immer wieder Probleme bei der Umsetzung von IPv6-Szenarien gehabt, da einige Distributionen automatisch IPv6-Firewallregeln aktiviert haben. Stellen Sie für Ihre Experimente sicher, dass Ihnen zunächst keine derartigen Regeln Probleme bereiten.

38.1 Anbindung über einen Tunnelbroker

Um auf das Internet6 über einen Provider zuzugreifen, der nur IPv4 unterstützt, benötigen Sie einen Tunnel. Neben vielen anderen Tunnelbrokern werden Freenet6 von GogoNET und SixXS meines Wissens am häufigsten verwendet. Beide bieten kostenlose Tunnel in das Internet6 an und verteilen dann IPv6-Präfixe für die Nutzung. Hier beschreibe ich beide Varianten.

38.1.1 Freenet6

Der Client für die Verbindung mit dem Tunnelbroker Freenet6 kann von der Homepage heruntergeladen werden. Verbinden Sie sich hierzu mit der URL `http://gogonet.gogo6.com/profile/gogoCLIENT`. Dort finden Sie den Client für verschiedene Betriebssysteme. Bei Debian Squeeze ist der Client bereits in der Distribution enthalten. Sein Name lautet `gogoc`. Installieren Sie auch gleich das Paket `radvd`.

Direkt nach der Installation kann der Client gestartet werden. Er baut dann auch schon eine Verbindung zum Tunnelbroker auf und erhält eine globale IPv6-Adresse.

```
root@ipv6-fw:~# ifconfig tun
tun       Link encap:UNSPEC  Hardware Adresse ↵
        00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet6-Adresse: 2001:5c0:1400:a::43b/128 Gültigkeitsbereich: ↵
                  Global
          UP PUNKTZUPUNKT RUNNING NOARP MULTICAST  MTU:1280  Metrik:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:500
          RX bytes:480 (480.0 B)  TX bytes:64 (64.0 B)

root@ipv6-fw:~# ping6 -c 1 ipv6.google.com
PING ipv6.google.com(2a00:1450:8003::69) 56 data bytes
64 bytes from 2a00:1450:8003::69: icmp_seq=1 ttl=55 time=28.0 ms

--- ipv6.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 28.020/28.020/28.020/0.000 ms
```

Um auch ein Präfix zu erhalten, müssen Sie ein gültiges Konto bei Freenet6 besitzen. Dieses können Sie ganz einfach über das Web erzeugen. Hierzu gehen Sie auf die Seite `http://gogonet.gogo6.com/page/freenet6-registration` und legen einen Benutzer an. Anschließend modifizieren Sie die GogoClient-Konfiguration:

```
userid=accountname
passwd=password

server=authenticated.freenet6.net
auth_method=any

host_type=router

if_prefix=eth1
```

Wenn Sie das Debian-Paket mit dem Startskript verwenden, müssen Sie nun den Gogo-Client einmalig im Vordergrund starten. Sie werden nun gefragt, ob Sie einen Serverkey akzeptieren und hinzufügen möchten. Bestätigen Sie dies:

```
root@ipv6-fw:~# gogoc -n
amsterdam.freenet6.net is an unknown host, do you want to add its key?? ( ↵
        Y/N) Y
```

Nun müssen Sie bei Verwendung des Debian-Paketes die Server-Zeile entsprechend dem verwendeten Server (hier: `amsterdam.freenet6.net`) anpassen. Ansonsten weigert sich das Startskript, den Tunnel zu starten:

```
root@ipv6-fw:~# /etc/init.d/gogoc start
Not starting gogoc - no server key ... (warning).
```

Falls Sie nicht das Debian-Paket verwenden oder anderweitig Probleme bekommen, sollten Sie die folgenden Punkte beachten:

- Aktivieren Sie die Protokollierung. Setzen Sie den Parameter `log_stderr` bzw. `log_syslog` auf den Wert 2.
- Der GogoClient möchte bei Verwendung der Funktion als Router gleich den Router Advertisement Daemon `radvd` starten. Wird dieser auf dem System nicht gefunden, schlägt die Verbindung fehl.
- Der `radvd` benötigt für den Betrieb einen unprivilegierten Benutzer und einige Verzeichnisse. Diese müssen angelegt werden.
- Der Router Advertisement Daemon startet nicht, wenn das IPv6-Forwarding nicht eingeschaltet wurde.

Sobald die Verbindung steht, wird für Sie auch ein DNS-Eintrag unter Ihrem Konto-Namen angelegt:

```
# host <account>.broker.freenet6.net
<account>.broker.freenet6.net has address 81.171.72.11
<account>.broker.freenet6.net has IPv6 address 2001:5c0:1400:b::a91d
```

Ein Client, der sich hinter dem Router befindet, erhält nun automatisch das Präfix und erstellt sich eine eigene globale IPv6-Adresse. Anschließend kann er auf das Internet zugreifen:

```
root@squeezeClient:~# ifconfig eth0
eth0      Link encap:Ethernet  Hardware Adresse 54:50:75:00:01:03
          inet Adresse:192.168.2.10  Bcast:192.168.2.255  Maske ↵
                  :255.255.255.0
          inet6-Adresse: 2001:5c0:1511:2a00:5650:75ff:fe00:103/64  ↵
                  Gültigkeitsbereich:Global
          inet6-Adresse: fe80::5650:75ff:fe00:103/64 Gültigkeitsbereich: ↵
                  Verbindung
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metrik:1
          RX packets:160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:229 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:1000
          RX bytes:13668 (13.3 KiB)  TX bytes:20661 (20.1 KiB)

root@squeezeClient:~# ping6 -c 1 ipv6.google.com
PING ipv6.google.com(2a00:1450:8003::68) 56 data bytes
64 bytes from 2a00:1450:8003::68: icmp_seq=1 ttl=54 time=24.7 ms
```

```
--- ipv6.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 24.778/24.778/24.778/0.000 ms
```

38.1.2 SixXS

Mit SixXS ist der Aufbau eines Tunnels ähnlich einfach. Jedoch müssen Sie hier bei der Registrierung mehr Informationen über sich selbst preisgeben, da diese von SixXS im eigenen Whois veröffentlicht werden.

Zunächst müssen Sie sich einen eigenen Handle erzeugen. Hierzu gehen Sie zu `https://www.sixxs.net/signup/create/`. Dort tragen Sie ihre eigenen Informationen ein. Bei falschen Informationen behält sich SixXS die Ablehnung vor. Sie erhalten eine E-Mail mit Ihrem Handle und Kennwort.

Nun müssen Sie sich anmelden und einen Tunnel beantragen: `https://www.sixxs.net/home/requesttunnel/`. Um den Tunnel zu nutzen, benötigen Sie die Software AICCU. Diese ist Bestandteil der meisten Linux-Distributionen. In der virtuellen Test-Umgebung ist diese bereits installiert. Hier müssen Sie nach der Beantragung des Tunnels die entsprechenden Parameter eintragen:

```
username nichandle
password password
tunnel_id Txxxx
```

Über den Tunnel erhalten Sie eine IPv6-Adresse. Diese können Sie bereits für den Zugriff auf das IPv6-Internet nutzen.

Für die Beantragung eines Subnetzes benötigen Sie IP SixXS Kredit (ISK). Dies ist eine virtuelle Währung, die von SixXS verwendet wird, um erweiterte Funktionen nur wirklich interessierten Nutzern zur Verfügung zu stellen. Bei der Anmeldung als Benutzer an der SixXS-Webseite erhalten Sie üblicherweise 25 ISK. Die Beantragung eines Tunnels und dessen Genehmigung kosten 10 bzw. 5 ISK, sodass Ihnen 10 ISK verbleiben. Für ein Subnetz benötigen Sie 14 ISK, die Sie aktuell noch nicht besitzen.

Für jede Woche, die Ihr Tunnel aktiv ist, erhalten Sie 5 ISK gutgeschrieben. Wenn Sie Ihren Tunnel also eine Woche betreiben, können Sie unter `https://www.sixxs.net/home/requestsubnet/` ein Subnetz beantragen. Dieses Subnetz wird dann auch im Whois eingetragen:

```
#  whois 2001:4dd0:ff26::/48
[Querying whois.ripe.net]
[whois.ripe.net]
...
```

```
% Information related to '2001:4dd0:ff26::/48'

inet6num:       2001:4dd0:ff26::/48
netname:        SIXXS-NET-RSM3-SIXXS
descr:          SixXS assignment to end-user RSM3-SIXXS
descr:          This prefix is routed to 2001:4dd0:ff00:5a::2
```

Um Informationen über den Nutzer zu erhalten, verwenden Sie:

```
# whois RSM3-SIXXS
[Querying whois.sixxs.net]
[whois.sixxs.net]
% This is the SixXS Whois server.
% SixXS - http://www.sixxs.net.
...

person:       Ralf Spenneberg
organisation: OpenSource Training Ralf Spenneberg
address:      Am Bahnhof 3-5
address:      48565 Steinfurt
address:      Germany
country:      DE
phone:        492552638755e-mail: info@os-t.deurl:
        http://www.os-t.denic-hdl: RSM3-SIXXSremarks: State:
        Enabledremarks: This object is generated from the SixXS
        databaseremarks: Abuse must be reported to abuse@sixxs.netremarks:
        Information can be found at http://www.sixxs.net/changed:
        info@sixxs.net 20090921changed: info@sixxs.net 20100226mnt-by:
        SIXXS-MNTsource: SIXXS
```

Um das Netz zu verteilen, verwenden Sie den Router Advertisement Daemon `radvd`, der im nächsten Abschnitt genauer betrachtet wird. Die SixXS-Software AICCU startet diesen nicht automatisch.

38.2 Verteilung der IP-Adressen mit radvd

Der Router Advertisement Daemon ist für die Verteilung des IPv6-Präfix im lokalen Netz zur Autokonfiguration der Clients verantwortlich.

Bei Verwendung des GogoClient wird der `radvd` automatisch gestartet. Bei allen anderen Verfahren zur Anbindung an das Internet6 müssen Sie sich selbst um den Start des Daemon kümmern.

Der `radvd` hat eine einfache Konfigurationsdatei und wird bei den meisten Distributionen über ein eigenes Startskript gestartet.

Die wesentlichen Parameter in der Konfigurationsdatei sind:

```
interface eth0
{
   AdvSendAdvert on;
   prefix 2001:db8::/64
   {
   };
};
```

Wichtig ist die Angabe der Netzwerkkarte, auf der die Advertisements versendet werden sollen, und die Angabe des Präfix. Hierbei kann der Router Advertisement Daemon auch mehrere Präfixe auf derselben Netzwerkkarte versenden. Dies vereinfacht ein Renumbering im laufenden Betrieb. Durch sinnvolle Wahl der Lebensdauern der einzelnen Präfixe kann ein Wechsel im laufenden Betrieb erfolgen. Um derartige Anpassungen vorzunehmen, gibt es noch eine ganze Anzahl weiterer Parameter, die in der Konfigurationsdatei angegeben werden können. Die wichtigsten möchte ich kurz vorstellen:

- `IgnoreIfMissing`: Hiermit startet der `radvd` auch, wenn die entsprechende Netzwerkkarte noch nicht gefunden wird.
- `AdvSendAdvert`: Hiermit sendet der Daemon in regelmäßigen Abständen automatisch die Advertisements.
- `AdvManagedFlag`: Einige Betriebssysteme erwarten dieses Flag bei der Verwendung von DHCPv6. RFC4862 erläutert die Verwendung.
- `AdvOtherConfigFlag`: Auch dieses Flag wird für die Nutzung von DHCPv6 benötigt.
- `AdvValidLifetime`: Dies ist die maximale Gültigkeit des Präfix (Default: 1 Tag).
- `AdvPreferredLifetime`: Dies ist die bevorzugte Lebensdauer. Während dieser Zeit wird die IP-Adresse für den Aufbau neuer Verbindungen bevorzugt (Default: 4 Stunden).
- `RDNSS`: Hiermit können zusätzliche DNS-Server angegeben werden.

38.3 Nutzung von DHCPv6

Ein großer Vorteil von IPv6 ist die automatische Konfiguration der IP-Adressen durch Router Advertisements. Jedoch hat diese Methode auch ihre Nachteile:

- Viele Systeme erwarten heute auch eine automatische Konfiguration weiterer Informationen, wie DNS-Server und -Domäne, NTP-Server und Zeitzone, SLP-Server etc. Das DHCP-Protokoll stellt neben vielen weiteren Informationen bei IPv4 eine einfache Verteilung dieser Informationen zur Verfügung.
- Für die Namensauflösung im Netz wird DNS genutzt. Wenn Systeme ihre eigenen IP-Adressen dynamisch erzeugen, müssten sie sich selbst anschließend im DNS eintragen, um gefunden zu werden. Windows-Active-Directory-Mitglieder können sich mit Kerberos am DNS-Server authentifizieren und selbst eintragen. Viele andere Systeme, z. B. Linux, sind dazu nicht automatisch in der Lage. In der Vergangenheit wurde häufig der Umweg

über DHCP genutzt. Nach der Vergabe der IP-Adresse trägt der DHCP-Server den Client im DNS ein.

Das erste Problem lässt sich mit stateless DHCPv6 lösen. Für das zweite Problem wird jedoch ein stateful DHCPv6 benötigt. Im ersten Fall ermittelt der Client seine IPv6-Adresse mithilfe der Autokonfiguration basierend auf den erhaltenen Router Advertisements. Dann erkundigt er sich bei dem DHCPv6-Server nach weiteren Konfigurationseinstellungen. Im zweiten Fall fordert der Client von dem DHCPv6-Server sowohl die IPv6-Adresse als auch weitere Informationen an. Er kann, muss aber nicht, dann auf die Autokonfiguration verzichten.

Im Folgenden möchte ich Ihnen erläutern, wie Sie eine oder beide Varianten einsetzen können.

Hierbei setze ich die ISC-DHCP-Software ein. Diese ist ab der Version 4.1 in der Lage mit gewissen Einschränkungen sowohl als stateless wie auch als stateful DHCP-Client und -Server zu arbeiten. Daher werden die anderen Implementierungen, wie die Wide-DHCP-Softwaresuite, in Zukunft weniger Bedeutung erhalten.

38.3.1 Stateless DHCPv6

Um einen Stateless-DHCP-Server aufzusetzen, benötigen Sie zunächst die Software. Bei der Debian-Squeeze-Distribution ist der ISC-DHCP-Server in der Version 4.1.1 enthalten (`isc-dhcp-server`).

INFO

Die Squeeze-Distribution bereitet das Paket nur für die Verwendung mit IPv4 vor. Für IPv6 müssen Sie zwei Änderungen in dem Startskript vornehmen. Ändern Sie die Zeilen 40 und 69 wie folgt ab, indem Sie ein `-6` einfügen:

```
40          if ! /usr/sbin/dhcpd -6 -t -q > /dev/null 2>&1; then
...
69                          --exec /usr/sbin/dhcpd -- -6 -q $INTERFACES
```

Zusätzlich müssen Sie die Datei `/var/lib/dhcp/dhcpd6.leases` als leere Datei anlegen.

Nun benötigen Sie noch eine Konfigurationsdatei für den DHCPv6-Server. Diese enthält im einfachsten Fall die folgenden Angaben:

```
default-lease-time 600;
max-lease-time 7200;
log-facility local7;
subnet6 2001:5c0:1511:2a00::/64 {
        # Additional options
        option dhcp6.name-servers fec0:0:0:1::1;
        option dhcp6.domain-search "domain.example";
}
```

Nach dem Start bindet sich der DHCPv6-Server auf den Port 547/udp. Dieser Port wird von dem DHCPv6-Protokoll genutzt.

Der Client `dhclient` ist ebenfalls IPv6-fähig. Bei Debian Squeeze lautet der Paketname `isc-dhcp-client`. Leider enthält dieses Paket ein `dhclient-script`, das nicht IPv6-fähig ist. Dies wird in dem Bug 591589[1] beschrieben. In diesem Bugreport finden Sie auch den Link zu einem Skript, das anstelle des orginalen Skripts die Anpassungen vornehmen kann. In der virtuellen Umgebung wurde dieses Skript bereits auf dem Squeeze-Client eingerichtet.

Um nun DHCPv6 als Client zu nutzen, rufen Sie den Befehl `dhclient` auf. Damit dieser auch IPv6 nutzt und nur Stateless DHCPv6 macht, geben Sie die Optionen `-6 -S` an:

```
root@squeezeClient:~# dhclient -v -6 -S eth0
Internet Systems Consortium DHCP Client 4.1.1-P1
Copyright 2004-2010 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Bound to *:546
Listening on Socket/eth0
Sending on   Socket/eth0
PRC: Requesting information (INIT).
XMT: Forming Info-Request, 0 ms elapsed.
XMT: Info-Request on eth0, interval 930ms.
RCV: Reply message on eth0 from fe80::5650:75ff:fe00:101.
PRC: Done.
root@squeezeClient:~# cat /etc/resolv.conf
search domain.example.
nameserver fec0:0:0:1::1
```

Die Option `-v` erzwingt die ausführliche Ausgabe auf dem Bildschirm.

38.3.2 Stateful DHCPv6

Um einen Stateful-DHCP-Server aufzusetzen, sind nur wenige zusätzliche Schritte erforderlich. Zunächst müssen Sie sich überlegen, welchen Range Sie in Ihrem Präfix verteilen wollen. Diesen tragen Sie dann zusätzlich in der Konfigurationsdatei des DHCP-Servers ein:

```
default-lease-time 600;
max-lease-time 7200;
log-facility local7;
subnet6 2001:5c0:1511:2a00::/64 {
        range6 2001:5c0:1511:2a00:0001::/96;
        option dhcp6.name-servers fec0:0:0:1::1;
        option dhcp6.domain-search "domain.example";
}
```

1 http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=619308

Um den Client im Stateful-Modus zu starten, verzichten Sie auf die Angabe der Option `-S`:

```
root@squeezeClient:~# dhclient -v -6 eth0
Internet Systems Consortium DHCP Client 4.1.1-P1
Copyright 2004-2010 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Bound to *:546
Listening on Socket/eth0
Sending on   Socket/eth0
PRC: Soliciting for leases (INIT).
XMT: Forming Solicit, 0 ms elapsed.
XMT:  X-- IA_NA 75:00:01:03
XMT:  | X-- Request renew in  3600XMT: | X- Request rebind in 5400
XMT: Solicit on eth0, interval 1090ms.
RCV: Advertise message on eth0 from fe80::5650:75ff:fe00:101.
RCV:  X-- IA_NA 75:00:01:03
RCV:  | X-- starts 1303820674
RCV:  | X-- t1 - renew  ORCV: | X- t2 - rebind 0
RCV:  | X-- [Options]
RCV:  | | X-- IAADDR 2001:5c0:1511:2a00:1:0:9e9d:7212
RCV:  | | | X-- Preferred lifetime 375.
RCV:  | | | X-- Max lifetime 600.
RCV:  X-- Server ID: 00:01:00:01:15:49:56:3d:54:50:75:00:01:01
RCV:  Advertisement recorded.
PRC: Selecting best advertised lease.
PRC: Considering best lease.
PRC:  X-- Initial candidate 00:01:00:01:15:49:56:3d:54:50:75:00:01:01 (s: ↵
         153, p: 0).
XMT: Forming Request, 0 ms elapsed.
XMT:  X-- IA_NA 75:00:01:03
XMT:  | X-- Requested renew  3600XMT: | X- Requested rebind 5400
XMT:  | | X-- IAADDR 2001:5c0:1511:2a00:1:0:9e9d:7212
XMT:  | | | X-- Preferred lifetime 7200XMT: | | | X- Max lifetime 7500
XMT:  V IA_NA appended.
XMT: Request on eth0, interval 1010ms.
RCV: Reply message on eth0 from fe80::5650:75ff:fe00:101.
RCV:  X-- IA_NA 75:00:01:03
RCV:  | X-- starts 1303820675
RCV:  | X-- t1 - renew  ORCV: | X- t2 - rebind 0
RCV:  | X-- [Options]
RCV:  | | X-- IAADDR 2001:5c0:1511:2a00:1:0:9e9d:7212
RCV:  | | | X-- Preferred lifetime 375.
```

```
RCV:  | | | X-- Max lifetime 600.
RCV:  X-- Server ID: 00:01:00:01:15:49:56:3d:54:50:75:00:01:01
PRC: Bound to lease 00:01:00:01:15:49:56:3d:54:50:75:00:01:01.

root@squeezeClient:~# ip -6 addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:5c0:1511:2a00:1:0:9e9d:7212/64 scope global
       valid_lft forever preferred_lft forever
    inet6 2001:5c0:1511:2a00:5650:75ff:fe00:103/64 scope global dynamic
       valid_lft 86081sec preferred_lft 14081sec
    inet6 fe80::5650:75ff:fe00:103/64 scope link
       valid_lft forever preferred_lft forever
```

Leider hat Debian Squeeze noch keine Unterstützung für den automatischen Aufruf des Kommandos `dhclient` für DHCPv6 in der Datei `/etc/networkinterfaces`. Andere Distributionen (Fedora, OpenSUSE etc.) sind dort bereits weiter in ihrer Unterstützung.

38.4 NAT64 und DNS64

In naher Zukunft wird die Anbindung von reinen IPv6-Clients an das IPv4-Internet ein Problem werden. Während es für bisherige Anwender und Nutzer mit IPv4-Adressen ein Leichtes ist, zusätzliche IPv6-Adressen zu erhalten und zu nutzen, werden bald viele Anwender nur noch IPv6-Adressen erhalten und nicht über zusätzliche IPv4-Adressen für den Zugriff auf Internet4-Inhalte verfügen. Mit `http://tradeipv4.com/` versuchen auch bereits erste Anbieter, IPv4-Adressen meistbietend zu verkaufen.

Um dennoch den IPv6-Anwendern Zugriff auf das Internet4 zu geben, wurde neben anderen Mechanismen NAT64 und DNS64 entworfen. Hiermit kann ein Dual-Stack-System mit je einer globalen IPv4- und IPv6-Adresse reinen IPv6-Clients Zugang zum Internet4 bieten. Ich halte dies mit für eine der erfolgversprechendsten Varianten und will kurz die Implementierung unter Linux erläutern.

Zunächst gibt es zwei verschiedene Implementierungen von NAT64 für Linux:

» TAYGA[2] arbeitet im Userspace und erhält die Pakete per TUN-Netzwerkkarten.

» Ecdysis[3] ist ein Netfilter-Modul. Leider wird diese Implementierung wahrscheinlich nicht fester Bestandteil des Linux-Kernels werden, da unterschiedliche Vorstellungen über die Implementierung existieren.[4]

Im Folgenden werde ich die Übersetzung und Installation des Netfilter-Moduls beschreiben. Die letzte Version vom 17.11.2010 ist auf der CD enthalten.

2 `http://www.litech.org/tayga/`
3 `http://ecdysis.viagenie.ca/`
4 `http://seclists.org/nanog/2011/Mar/178`

Zunächst müssen Sie den Quelltext auspacken. Anschließend wechseln Sie in das neu entstandene Verzeichnis und rufen die folgenden Befehle auf:

```
make
make install
depmod -a
```

Schlägt die Übersetzung fehl, müssen Sie wahrscheinlich eine Build-Umgebung und das Kernel-Headers-Paket installieren. Unter Debian erreichen Sie dies mit folgenden Aufruf:

```
aptitude install build-essentials linux-headers-2.6.32-5-amd64
```

Passen Sie dabei die Kernel-Version an Ihren installierten Kernel an.

Bestandteil des Netfilter-Modulpakets ist auch ein Skript namens `nat64-config.sh`, das Sie nun editieren müssen:

```
#!/bin/bash

#IPV4_ADDR="x.x.x.x"
PREFIX_ADDR="dead:beef::"
PREFIX_LEN="96"
...
```

Hiermit weisen Sie das Modul an, für alle IPv6-Adressen, die mit `dead:beef` beginnen, ein NAT64 durchzuführen. Sie können hier auch ein anderes Präfix wählen. Nachdem Sie das Skript editiert haben, rufen Sie es unter Angabe Ihrer globalen IPv4-Adresse auf:

```
# ./nat64-config.sh 192.168.1.1
**************
nf_nat64 setup
**************

Info: Using 192.168.1.1 as the NAT64 IPv4 address.
Info: Using dead:beef::/96 as the NAT64 Prefix.


+ modprobe -r nf_nat64
+ modprobe nf_nat64 nat64_ipv4_addr=192.168.1.1 nat64_prefix_addr=dead: ↵
        beef:: nat64_prefix_len=96
+ ifconfig nat64 up
+ ip -6 route add dead:beef::/96 dev nat64
+ sysctl -w net.ipv4.conf.all.forwarding=1
net.ipv4.conf.all.forwarding = 1
+ sysctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding = 1
```

Nun müssen Sie noch die DNS64-Funktion bereitstellen. Ab der Version 9.8 unterstützt der Bind-DNS-Server die DNS64-Funktion.

Hierzu wurde eine neue Option im Bind eingeführt:

```
dns64 IPv6_prefix {
[clients {address_match_list };]
[mapped {address_match_list };]
[exclude {address_match_list };]
[suffix IPv6_addr;]
[recursive-only (yes|no);]
[break-dnssec (yes|no);]
};
```

Der `IPv6_prefix` ist das Präfix, an das die IPv4-Adressen angehängt werden, wenn nur eine IPv4- und keine IPv6-Adresse für den angefragten DNS-Namen bekannt ist. Diese Angabe ist erforderlich. Hier tragen Sie die Präfix-Adresse ein, die Sie oben in dem Skript `nat64-config.sh` angepasst haben (`PREFIX_ADDR`).

Mit dem `clients`-Parameter können Sie die Funktion auf eine bestimmte Menge von Clients einschränken. Der Default ist `any`.

Mit dem `mapped`- und dem `exclude`-Parameter können Sie die Menge der IPv4-Adressen, für die eine Umschreibung erfolgen soll, einschränken.

Das `suffix` definiert weitere Bits in der endgültigen IPv6-Adresse. Wenn Sie ein /64-Präfix definieren, werden für die IPv4-Umsetzung nur 32 Bits benötigt. Die restlichen 32-Bits definieren Sie in diesem Suffix.

Der `recursive-only`-Parameter entscheidet, ob die Umschreibung nur für rekursiv aufgelöste Anfragen erfolgt.

Mit `break-dnssec` können Sie schließlich entscheiden, wie Bind bei DNSSEC-signierten Antworten reagieren soll.

39. Filterung mit ip6tables

Der Befehl `ip6tables` funktioniert genauso wie der Befehl `iptables`. Im Wesentlichen können Sie alle Funktionen, die Sie mit dem Befehl `iptables` nutzen, auch mit dem Befehl `ip6tables` nutzen. Einige IPv6-spezifische Targets und Matches sind aber hinzugekommen.

STOP

Verzichten müssen Sie auf die Network Address Translation (NAT) von IPv6-Adressen auf IPv6-Adressen, da diese für IPv6 nicht in den Standards erlaubt und spezifiziert wurde. Dies bezeichnet man häufig auch als NAT66. Ein NAT64 von IPv6-Adressen auf IPv4-Adressen ist möglich, und wir haben es bereits in Abschnitt 38.4 betrachtet. Ein NAT46 von IPv4 auf IPv6 ist nicht ganz einfach, und sämtliche RFCs, die dieses NAT beschreiben, sind in den Zustand Deprecated und Historic versetzt worden. Dies wurde in Kapitel 37 zu den Übergangstechniken von IPv4 zu IPv6 genauer erläutert.

Der Befehl `ip6tables` unterstützt die folgenden Ziele:

» `ACCEPT`, `DROP`, `RETURN` und `QUEUE`, da es sich hier um fest eingebaute Ziele handelt. Alle weiteren Ziele werden über ladbare Kernelmodule realisiert.

» `LOG`, `MARK`, `NFQUEUE`, `REJECT`, `CLASSIFY`, `CONNMARK`, `CONNSECMARK`, `CT`, `DSCP`, `NFLOG`, `NOTRACK`, `RATEEST`, `SECMARK`, `SET`, `TCPMSS`, `TCPOPTSTRIP`, `TEE`, `TOS` und `TRACE` stehen, wie für das IPv4-Protokoll, zur Verfügung.

» `HL`: Dies ist ein Ziel, das nur für das IPv6-Protokoll zur Verfügung steht. Es wird weiter unten erläutert.

Diese Ziele können Sie in den drei Tabellen `raw`, `filter` und `mangle` einsetzen. Die Tabelle `nat` steht nicht zur Verfügung.

Um die Pakete zu prüfen, können Sie auf die folgenden Tests zurückgreifen:

» `-p, --protocol, -s, --source, -d, --destination, -i, --in-interface, -o, --out-interface` haben dieselbe Funktion wie bei IPv4.

» `-p icmpv6 --icmpv6-type`: Um das spezielle ICMPv6-Protokoll zu unterstützen, haben Sie hier einen eigenen Test.

» Erweiterungen, die genauso wie die Iptables-Erweiterungen funktionieren, sind:

- `ah`
- `cluster`
- `comment`
- `connbytes`
- `connlimit`

- connmark
- conntrack
- dccp
- dscp
- esp
- hashlimit
- helper
- iprange
- length
- limit
- mac
- mark
- multiport
- owner
- pkttype
- policy
- quota
- rateest
- sctp
- set
- state
- statistic
- string
- tcpmss
- time
- tos
- u32

» Neu hinzugekommen für das Protokoll IPv6 sind:

- dst: Hiermit können Sie Optionen im IPv6-Destination-Header prüfen.
- eui64: Dies prüft, ob der EUI64-Teil einer autokonfigurierten IPv6-Adresse stimmt.
- frag: Hiermit können Sie Optionen im IPv6-Fragmentation-Header prüfen.
- hbh: Dies prüft den IPv6-Hop-by-Hop-Header.
- hl: Dieser Test prüft das Hop-Limit-Feld im IPv6-Header.
- ipv6header: Hiermit können Sie Optionen im IPv6-Header prüfen.
- mh: Dieser Test prüft den Mobility-Header im IPv6-Header.
- rt: Hiermit prüfen Sie den IPv6-Routing-Header.

40. Neue IPv6-Targets

40.1 HL

Mit diesem Ziel können Sie das IPv6-Hoplimit-Feld modifizieren. Der Hoplimit ist mit dem TTL-Feld des IPv4-Protokolls vergleichbar. Dieses Target darf nur in der Mangle-Tabelle eingesetzt werden. Die Verwendung des Targets ist gefährlich, da eine Erhöhung des Hoplimits Routing-Loops erzeugen kann.

Das Target hat die folgenden Optionen:

» `--hl-set <hops>`: Hiermit setzen Sie den Wert absolut.

» `--hl-dec <wert>`: Hiermit reduzieren Sie das Hoplimit um den angegebenen Wert.

» `--hl-inc <wert>`: Hiermit erhöhen Sie das Hoplimit um den angegebenen Wert.

CE

41. Neue IPv6-Matches

41.1 dst

Mit diesem Test können Sie den IPv6-Destination-Header prüfen. Der Test hat zwei zusätzliche Optionen:

» `--dst-len <länge>`: Dies prüft die totale Länge des Headers.

» `--dst-opts <TYPE>[:<LEN>], []`: Hiermit prüfen Sie einzelne Optionen und ihre Länge.

41.2 eui64

Wenn die IPv6-Adresse per Autokonfiguration zugewiesen wurde, enthalten die niedrigen 64 Bit der IP-Adresse die MAC-Adresse der Netzwerkkarte. Dieser Test prüft, ob das tatsächlich der Fall ist.

41.3 frag

Bei IPv6 sind lediglich der Absender und der Empfänger für die Fragmentierung und Defragmentierung zuständig. Router dürfen Pakete nicht fragmentieren. Die Path Maximum Transmission Unit Discovery (PMTUD) ist für alle Systeme Pflicht. Da der originale Absender jedoch fragmentierte Pakete versenden darf und kann, ist es in gewissen Umgebungen auch erforderlich, diese zu prüfen.

STOP

Sobald Sie die Zustandsüberwachung des Linux-Kernels nutzen (Connection Tracking), werden alle Pakete automatisch von dem Linux-System defragmentiert. Anschließend werden diese Pakete unter Umständen, wenn erforderlich, auch wieder fragmentiert. Der Linux-Netfilter-Code betrachtet jedoch die defragmentierten Pakete!

Sobald IPv6-Pakete fragmentiert werden, enthält das Paket einen zusätzlichen Fragment-Header. Für die Prüfung der Informationen in dem Fragment-Header gibt es viele verschiedene Optionen:

» `--fragid`: Hiermit können Sie die Identifikationsnummer des Pakets prüfen. Dabei können Sie auch einen Bereich (`id:id`) angeben.

» `--fraglen`: Diese Option existiert in den modernen Kerneln nicht mehr, da die Länge des Fragment-Headers statisch ist und sich nicht ändert. Bis zur Kernel-Version 2.6.10 ist diese Option jedoch vorhanden.

» `--fragres`: Hiermit prüfen Sie, ob die reservierten Felder im Fragment-Header mit 0 gefüllt sind.

» `--fragfirst`: Dieser Test erkennt das erste Fragment eines Pakets.

» `--fragmore`: Hiermit prüfen Sie, ob weitere Fragmente folgen.

» `--fraglast`: Diese Option erkennt das letzte Fragment eines Pakets.

41.4 hbh

Hiermit können Sie die IPv6-Hop-by-Hop-Optionen testen. Die Option `--hbh-len <länge>` prüft die totale Länge, während Sie mit `--hbh-opts <Type>[:<länge>]` die einzelnen Optionen und ihre Länge prüfen können.

41.5 hl

Dieses Modul prüft das Hoplimit-Feld. Sie können prüfen, ob der Wert mit einem bestimmten Wert übereinstimmt, kleiner oder größer ist. Dieser Test ist mit dem `ttl`-Test vergleichbar.

» `--hl-eq <hops>`: gleich

» `--hl-lt <hops>`: kleiner

» `--hl-gt <hops>`: größer

41.6 ipv6header

Mit diesem Test können Sie die Optionen im IPv6-Header testen. Mit der Option `--header <headers>` können Sie die Header angeben, die in dem Paket enthalten oder nicht (`!`) enthalten sein müssen. Dabei müssen Sie alle Header angeben, damit der Test zutrifft. Wenn Sie nur einen Teil angeben möchten, müssen Sie zusätzlich die Option `--soft` definieren.

Sie können die folgenden Header testen: `hop`, `dst`, `route`, `frag`, `auth`, `esp`, `none`, `proto`.

41.7 mh

IPv6 unterstützt auch Mobile-IPv6 mit dem Mobility-Header. Diese Erweiterung erlaubt es, den Header zu analysieren und seinen Typ zu erkennen. Hierzu unterstützt die Erweiterung den Match `--mh-type`. Gültige Typen, die Sie damit prüfen können, sind:

» binding-refresh-request (`brr`)

» home-test-init (`hoti`)

» careof-test-init (`coti`)

» home-test (`hot`)

» careof-test (`cot`)

» binding-update (`bu`)

» binding-acknowledgement (`ba`)

» binding-error (`be`)

CE Ist das nicht etwas wenig für ein eigenes Kapitel?

Die Verwendung der Erweiterung ist nur erlaubt, wenn das Paket auch tatsächlich das entsprechende Protokoll nutzt:

```
ip6tables -A FORWARD --protocol ipv6-mh -m mh --mh-type binding-error -j ↵
        REJECT
```

41.8 rt

Dieser Test prüft den Routing-Header. Hierfür haben Sie die folgenden Optionen:

» `--rt-type <type>`: Hiermit prüfen Sie den numerischen Typ des Routing-Headers.

» `--rt-segsleft <num>[<num>]`: Dies prüft das Segments-Left-Feld.

» `--rt-len <länge>`: Hiermit prüfen Sie die Länge des Headers.

» `--rt-0-res`: Dies prüft, ob das reservierte Feld gesetzt ist.

» `--rt-0-addrs <ip>[,<ip>]`: Dies prüft, ob im reservierten Feld (Typ 0) eine Adresse vorhanden ist.

» `--rt-0-not-strict`: Die letzte Option muss alle IP-Adressen im Routing-Header definieren. Wenn Sie lediglich eine prüfen möchten, müssen Sie zusätzlich diese Option angeben.

42. Empfohlene IPv6-Regeln

In diesem Kapitel möchte ich Ihnen einige Hinweise an die Hand geben, welche Regeln Sie in Ihrer Firewall berücksichtigen sollten, damit IPv6 bei Ihnen funktioniert. Da IPv6, wie ich bereits in Kapitel 36 erwähnt habe, kein Address Resolution Protocol mehr nutzt, sondern auch hierfür mit ICMPv6 ein IP-Protokoll nutzt, ist es erforderlich, dies in den Firewall-Regeln zu berücksichtigen. Daher werde ich mit den empfohlenen Regeln für ICMPv6 beginnen. Anschließend werde ich noch einige Hinweise für die Sicherheit des lokalen Netzes ohne Network Address Translation geben.

42.1 ICMPv6-Regeln

Dieses Thema ist so wichtig, dass sogar ein eigenes RFC-Dokument hierzu existiert. Ich werde versuchen, die wichtigsten Informationen aus dem RFC4890 für Sie leicht verständlich zusammenzufassen. Wenn Sie sich die Mühen, die in diesem Abschnitt erläutert werden, sparen möchten, ist es natürlich auch möglich, sämtlichen ICMPv6-Verkehr zu erlauben. Dann sind jedoch in Abhängigkeit von der Implementierung unter Umständen Angriffe mit dem Protokoll möglich. Um den gesamten ICMPv6-Verkehr zu akzeptieren und auf die Robustheit der entsprechenden Betriebssysteme zu setzen, verwenden Sie:

```
$IPT6=/usr/sbin/ip6tables

$IPT6 -A INPUT -p icmpv6 -j ACCEPT
$IPT6 -A OUTPUT -p icmpv6 -j ACCEPT
$IPT6 -A FORWARD -p icmpv6 -j ACCEPT
```

42.1.1 ICMPv6 – eine kurze Wiederholung

Das ICMPv6-Protokoll spielt eine wichtige Rolle für die Funktion des Netzwerks. Es ist für die Neighbor Discovery, Duplicate Address Detection, Path MTU Discovery, Benachrichtigung bei Fehlern und Prüfung der Erreichbarkeit erforderlich. Fehler in der Filterung der Meldungen können sowohl eine Beeinträchtigung der Leistung als auch einen scheinbaren Totalausfall des Netzes nach sich ziehen.

Im Wesentlichen gehören die ICMPv6-Nachrichten einer der folgenden Gruppen an:

» Fehlermeldungen, z. B. Host-Unreachable
» Prüfung der Erreichbarkeit, z. B. Echo-Request
» Entdeckung von Netzwerkkomponenten oder -eigenschaften, z. B. Neighbor oder Router Solicitation, Packet To Big (PMTU)
» experimentelle Erweiterungen

Wie und ob diese Meldungen gefiltert oder zugelassen werden, hängt von dem Einsatzort der Firewallregeln ab. Viele dieser Meldungen sind nur im Kontext einer Link-Local-Umgebung sinnvoll. Grundsätzlich müssen hier vier verschiedene Einsatzszenarien betrachtet werden:

» einzelner Host
» Bridge innerhalb eines Netzes
» Router innerhalb eines Netzes
» Router am Übergang zum Internet6

Tatsächlich ist ein Router jedoch auch ein Host, der an dem lokalen Netzwerkverkehr teilnimmt. Daher ist es sinnvoller, den Verkehr in zwei andere Gruppen einzuteilen:

weiterzuleitender Verkehr. Hierbei handelt es sich um Pakete, die von einem Router weitergeleitet werden.

lokal bestimmter Verkehr. Dies ist der Netzwerkverkehr, der nur lokal ausgewertet oder von einer Bridge weitergeleitet wird.

ICMPv6-Nachrichten können IPv6-Adressen mit unterschiedlichen Scopes nutzen. So können sowohl Link-Local- als auch globale IPv6-Adressen verwendet werden. In einigen Fällen (Address-Autokonfiguration) wird auch die unspezifizierte IPv6-Adresse (`::/128`) verwendet.

42.1.2 ICMPv6 aus Sicht der Firewall

Um nun die ICMPv6-Nachrichten zu filtern, können diese wieder in Gruppen eingeteilt werden:

» Nachrichten, die nicht gefiltert werden dürfen
» Nachrichten, die nicht gefiltert werden sollten
» Nachrichten, die gefiltert werden dürfen
» Nachrichten, die gefiltert werden sollten

Im Folgenden werde ich Ihnen für die verschiedenen Einsatzorte die Zuordnung der ICMPv6-Filter zu diesen Gruppen erläutern. Gleichzeitig werde ich Ihnen die sinnvollen `ip6tables`-Regeln vorstellen. Hierbei sollten Sie, wenn möglich, von dem Connection Tracking und der Erkennung der Nachrichten Gebrauch machen!

42.1.3 ICMPv6-Filter für weiterzuleitenden Verkehr

Auf einem Router sollten Sie die Firewall-Regeln so aufsetzen, dass die ICMPv6-Nachrichten entsprechend den folgenden Vorgaben gefiltert werden:

» Nachrichten, die nicht gefiltert werden dürfen
  - Fehlermeldungen
    - *destination-unreachable*
    - *packet-too-big*
    - *time-exceeded* (ttl-zero-during-transit)
    - *parameter-problem* (unknown-header-type, unknown-option)
  - Prüfung der Konnektivität
    - *echo-request*[1]

» Nachrichten, die nicht gefiltert werden sollten
  - *time-exceeded* (ttl-zero-during-reassembly)
  - *parameter-problem* (bad-header)
  - mobile IPv6-Nachrichten, wenn Sie davon Gebrauch machen möchten[2]
    - *home-agent-address-discovery-request* und Reply
    - *mobile-prefix-solicitation* und Advertisement

» Nachrichten, die gefiltert werden sollten
  - sämtliche weiteren Pakete

Um dieses Ziel zu erreichen, können die folgenden Regeln verwendet werden:

```
IPT6=/usr/sbin/ip6tables

MOBILE="0"

$IPT6 -P FORWARD DROP
$IPT6 -N myicmp6forward
$IPT6 -N myicmp6mobile

$IPT6 -A myicmp6forward -m state --state INVALID -j DROP
$IPT6 -A myicmp6forward -m state --state ESTABLISHED -j ACCEPT

$IPT6 -A myicmp6forward -p icmpv6 --icmpv6-type destination-unreachable ↵
        -m state --state RELATED -j ACCEPT
$IPT6 -A myicmp6forward -p icmpv6 --icmpv6-type packet-too-big -m state ↵
        --state RELATED -j ACCEPT
```

1 Die Wahrscheinlichkeit eines Host-Scans ist bei den großen IPv6-Netzen sehr gering.
2 Ich persönlich möchte dies nicht nutzen. Aktuell erwarte ich durch diese Funktion mehr Probleme als Nutzen. Daher würde ich auf die Funktion verzichten und die entsprechenden Pakete verwerfen.

```
$IPT6 -A myicmp6forward -p icmpv6 --icmpv6-type time-exceeded -m state
        --state RELATED -j ACCEPT
$IPT6 -A myicmp6forward -p icmpv6 --icmpv6-type parameter-problem -m
        state --state RELATED -j ACCEPT

$IPT6 -A myicmp6forward -p icmpv6 --icmpv6-type echo-request -m state
        --state NEW -j ACCEPT

   $IPT6 -A myicmp6mobile -p icmpv6 --icmpv6-type 144 -m state --state
           NEW -j ACCEPT
   $IPT6 -A myicmp6mobile -p icmpv6 --icmpv6-type 146 -m state --state
           NEW -j ACCEPT


$IPT6 -A FORWARD -p icmpv6 -j myicmp6forward
if [ $MOBILE -eq 1 ]; then
$IPT6 -A FORWARD -p icmpv6 -j myicmp6mobile
fi
```

42.1.4 ICMPv6-Filter für lokalen Verkehr

Die Filterung des lokalen Verkehrs ist sehr schwierig, da das IPv6-Protokoll im Allgmeinen und das ICMPv6-Protokoll im Speziellen sehr stark Multicast-Kommunikation nutzt. Dadurch kann die Zustandsüberwachung im Linux-Kernel häufig keine Verbindung zwischen den verschiedenen Paketen herstellen. Außerdem werden Router Advertisements häufig auch von den Routern automatisch gesendet, ohne vorher durch eine Router-Solicitation-Nachricht angefordert worden zu sein.

Speziell ältere Linux-Kernel haben hiermit Probleme und kennzeichnen entsprechende ICMPv6-Nachrichten häufig als `INVALID`. Neuere Kernel (ab 2.6.29) haben einen Patch[3] erhalten, der diesen Zustand in `UNTRACKED` ändert.

Die einzelnen Nachrichten fallen nun in die folgenden Gruppen:

» Nachrichten, die nicht gefiltert werden dürfen
- Fehlermeldungen
  - *destination-unreachable*
  - *packet-too-big*
  - *time-exceeded* (ttl-zero-during-transit)
  - *parameter-problem* (unknown-header-type, unknown-option)

3 http://www.spinics.net/lists/netfilter-devel/msg07182.html

- Prüfung der Konnektivität
  - *echo-request*[4]
- Konfiguration der Adressen und Router
  - *router-solicitation* und *router-advertisement*
  - *neighbor-nolicitation* und *neighbor-advertisement*
  - *inverse-neighbor-discovery-solicitation* und Advertisement
- Multicast-Gruppen-Verwaltung
  - *listener-query*
  - *listener-report*
  - *listener-done*
  - *listener-report-v2*
- SEND Certificate Patch Notification[5]
  - *certificate-path-solicitation* and Advertisement
- Multicast Router Discovery
  - *multicast-router-solicitation*, Advertisement und Termination

» Nachrichten, die nicht gefiltert werden sollten

» Nachrichten, die nicht gefiltert werden sollten

- *time-exceeded* (ttl-zero-during-reassembly)
- *parameter-problem* (bad-header)

» Nachrichten, die gesondert betrachtet werden müssen

- *redirect*: In Abhängigkeit von der Umgebung sollten Redirect-Meldungen gefiltert werden. Hiermit ist ein Router-Spoofing-Angriff möglich (siehe auch Abschnitt A.16.4).

Da sich die Aufstellung der zu erlaubenden ICMPv6-Regeln mit den in dem vorherigen Abschnitt aufgezählten Regeln deckt, können diese Regeln hier wiederverwendet werden. Für die weiteren Nachrichten werde ich zusätzlich Regeln aufführen. Leider ist es nicht möglich, die Nachrichten als Bereich anzugeben. Für viele Nachrichten gibt es auch noch keine Text-Repräsentation, sodass die Listener-Query-, Report-, Done- und Reportv2-Nachrichten als 130–132 und 143 dargestellt werden müssen.

```
IPT6=/usr/sbin/ip6tables

$IPT6 -P INPUT DROP
$IPT6 -P OUTPUT DROP
$IPT6 -N myicmp6local
```

4 Die Wahrscheinlichkeit eines Host-Scans ist bei den großen IPv6-Netzen sehr gering.

5 SEND ist die Secure Neighbor Discovery, die ein Spoofing mithilfe von Zertifikaten verhindert. Nur wenige Betriebssysteme unterstützen diesen Mechanismus.

```
$IPT6 -A myicmp6local -p icmpv6 --icmp-type router-solicitation -j ACCEPT
$IPT6 -A myicmp6local -p icmpv6 --icmp-type router-advertisement -j ↵
        ACCEPT
$IPT6 -A myicmp6local -p icmpv6 --icmp-type neighbor-solicitation -j ↵
        ACCEPT
$IPT6 -A myicmp6local -p icmpv6 --icmp-type neighbor-advertisement -j ↵
        ACCEPT
# Inverse Neighbor Discovery
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 141 -j ACCEPT
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 142 -j ACCEPT
# Multicast Gruppen Verwaltung
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 130 -j ACCEPT
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 131 -j ACCEPT
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 132-j ACCEPT
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 143 -j ACCEPT
# SEND
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 148 -j ACCEPT
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 149 -j ACCEPT
# Multicast Router Discovery
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 151-j ACCEPT
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 152-j ACCEPT
$IPT6 -A myicmp6local -p icmpv6 --icmp-type 153-j ACCEPT
$IPT6 -A myicmp6local -j DROP

# Verwende die Forward-Regeln auch für eingehende und ausgehende Pakete
$IPT6 -A INPUT -p icmpv6 -j myicmp6forward
$IPT6 -A OUTPUT -p icmpv6 -j myicmp6forward

$IPT6 -A INPUT -p icmpv6 -j myicmp6local
$IPT6 -A OUTPUT -p icmpv6 -j myicmp6local
```

42.2 Weitere IPv6-spezifische Regeln

Mit IPv6 ändern sich viele Aspekte des Protokolls. Im Grunde genügen diese Änderungen und die Möglichkeiten des Protokolls, um ein eigenes Buch nur über IPv6-Firewalling zu schreiben. Jedoch ist meines Erachtens angesichts der Tatsache, dass IPv6 noch sehr wenig eingesetzt wird, der Markt für ein derartiges Buch noch nicht vorhanden. Interessierte Leser möchte ich daher auf die folgenden RFCs verweisen:

- RFC4864 – Local Network Protection for IPv6
- RFC4942 – IPv6-Transition/Coexistence Security Considerations
- RFC5722 – Handling of overlapping IPv6-Fragments

Grundsätzlich ist bei der Verwendung von IPv6 die Nutzung der Zustandsüberwachung von Linux wesentlich wichtiger. Da nun die Network Address Translation (NAT) keine scheinbare Sicherheit durch die Verschleierung des internen Netzes erzeugt, muss die Perimeter-Firewall sicherstellen, dass keine unerwünschten Zugriffe auf die Systeme möglich sind. Scheinbar wird dadurch die Konfiguration der Firewallregeln komplizierter. Tatsächlich wird sie in Wirklichkeit aber viel einfacher, da umständliche Port-Forwarding-Regeln, um interne durch NAT verborgene Systeme öffentlich erreichbar zu machen, nun nicht mehr erforderlich sind.

Ein typisches Grundmuster für eine `ip6tables`-Firewall könnte daher wie folgt aussehen:

```
IPT6=/usr/sbin/ip6tables

INTDEV=eth0
EXTDEV=eth1

$IPT6 -P INPUT DROP
$IPT6 -P OUTPUT DROP
$IPT6 -P FORWARD DROP

# Ist die Unterstützung von MobileIPv6 erwünscht?
$MOBILE="0"

# Erzeuge neue Ketten für ICMPv6
$IPT6 -N myicmp6local
$IPT6 -N myicmp6forward
$IPT6 -N myicmp6mobile

# Aus Platzgründen wurde auf die Wiederholung der Ketten myicmp6* ↵
        verzichtet

### Lokaler Verkehr
$IPT6 -A INPUT -i lo -j ACCEPT
$IPT6 -A INPUT -m state --state ESTABLISHED -j ACCEPT
$IPT6 -A INPUT -p icmpv6 -j myicmp6forward
$IPT6 -A INPUT -p icmpv6 -j myicmp6local
$IPT6 -A OUTPUT -o lo -j ACCEPT
$IPT6 -A OUTPUT -m state --state ESTABLISHED -j ACCEPT
$IPT6 -A OUTPUT -p icmpv6 -j myicmp6forward
$IPT6 -A OUTPUT -p icmpv6 -j myicmp6local

# Erlaube Zugriff auf die folgenden lokalen Dienste
$IPT6 -A INPUT -p tcp -m state --state NEW -m multiport --dport 22,80,443 ↵
         -j ACCEPT
```

```
# Erlaube Zugriff auf die folgenden Dienste im Netz
$IPT6 -A OUTPUT -p tcp -m state --state NEW -m multiport --dport ↵
        53,22,80,443 -j ACCEPT
$IPT6 -A OUTPUT -p udp -m state --state NEW --dport 53 -j ACCEPT


### Als Router erlaube die folgenden Pakete
$IPT6 -A FORWARD -m state --state ESTABLISHED -j ACCEPT

$IPT6 -A FORWARD -p icmpv6 -j myicmp6forward
if [ $MOBILE -eq 1 ]; then
$IPT6 -A FORWARD -p icmpv6 -j myicmp6mobile
fi

$IPT6 -A FORWARD -i $INTDEV -o $EXTDEV -p tcp -m state --state NEW -m ↵
        multiport --dport 53,22,80,443 -j ACCEPT
$IPT6 -A FORWARD -i $INTDEV -o $EXTDEV -p udp -m state --state NEW -- ↵
        dport 53 -j ACCEPT
```

Basierend auf diesem Beispielskript sollte Ihrer IPv6-Firewall nichts mehr im Wege stehen.

A. Netzwerk grundlagen

Dieses Kapitel soll eine kurze Einführung in die im Internet verwendeten Protokolle geben. Es soll sowohl als Einführung wie auch als Wiederauffrischung und als Nachschlagekapitel dienen. Die wesentlichen Punkte, um die Netzwerkprotokolle im Zusammenhang mit Firewalls zu verstehen, sollen erklärt werden. Ausführliche Darstellungen finden Sie in [12] und [13] (siehe die Bibliografie in Abschnitt B).

A.1 TCP/IP

Die TCP/IP-Protokollfamilie wird seit 1973 entwickelt. 1978 wurde die Version IPv4 fertiggestellt.
Diese Version findet heute die meiste Verwendung. Ab 1982 wurde das damalige ARPANET auf das neue Protokoll IP umgestellt. Heutzutage ist TCP/IP *das* im Internet verwendete Protokoll. Viele Firmen haben in den vergangenen Jahren ihre Netze ebenfalls auf TCP/IP umgestellt, um so intern Internet-ähnliche Dienste anbieten zu können und eine einfache Kommunikation auch mit dem Internet zu ermöglichen.

Das OSI-Referenzmodell wird verwendet, um die Netzwerkprotokolle in sieben verschiedene Schichten aufzuteilen. Hierbei handelt es sich um die folgenden Schichten: Physical, Data-Link, Network, Transport, Session, Presentation und Application. Das TCP/IP-Protokoll wurde vor dem OSI-Modell entwickelt. Es besitzt daher nicht diese exakte Unterteilung in sieben Schichten, sondern verwendet nur vier Schichten:

1. **Netzzugang:** Diese Schicht entspricht im OSI-Modell der Schicht 1 und 2.
2. **Internet:** Diese zweite Schicht des TCP-Modells entspricht im OSI-Modell der Schicht 3. Hier arbeitet das IP-Protokoll. Daher wird häufig umgangsprachlich auch von der Schicht 3 gesprochen, wenn das IP-Protokoll gemeint ist.
3. **Transport:** Dies entspricht der Schicht 4 des OSI-Modells. Hier arbeiten die Protokolle TCP und UDP.
4. **Anwendung:** Diese Schicht bildet die Schichten 5–7 des OSI-Modells ab. Hier arbeiten die Applikationsprotokolle wie HTTP, Telnet, FTP etc.

A.2 IPv4

Das IPv4-Protokoll (RFC791, STD 5) ist dafür verantwortlich, IP-Datagramme in Paketen von einer Quelle zu einem Zielrechner zu übertragen. Hierbei kümmert sich das IP-Protokoll um

die Zustellung des Pakets zu diesem Zielrechner. Die revolutionäre Neuerung bei der Einführung des IP-Protokolls war die Tatsache, dass eine IP-Kommunikation keine dedizierte Verbindung (*Circuit Switched Network*) mehr benötigte, sondern dass die Daten in einzelnen Paketen (*Packet Switched Network*) unabhängig ihr Ziel erreichten (siehe Abbildung A.1).

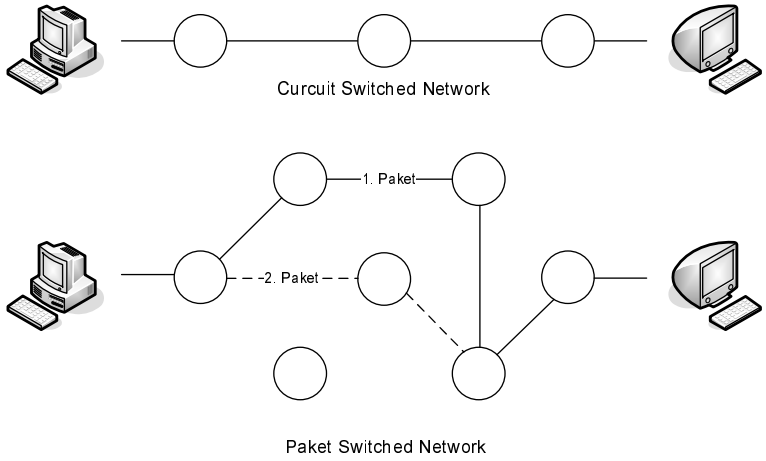


Abbildung A.1: **Circuit Switched Network vs. Packet Switched Network**

Dies ermöglicht es, die Kommunikation beim Ausfall redundanter Netzwerkkomponenten durch dynamische Routing-Protokolle aufrechtzuerhalten. Diese sind in der Lage, den Ausfall zu erkennen und die Daten über andere Knoten weiterhin zu übermitteln. Die hierzu benötigten Informationen werden im IP-Header des Pakets abgespeichert (siehe Abbildung A.2). Dennoch ist IP ein Protokoll, das die Zustellung des Pakets nicht garantiert und überprüft. Es wird auch als Best-Effort-Protokoll bezeichnet. Die höheren Protokolle oder die Anwendungen müssen die erfolgreiche Übertragung prüfen, wenn dies erforderlich ist.

Die Felder und ihre Bedeutung sollen nun kurz vorgestellt werden.

A.2.1 Version

Dieses Feld ist vier Bits lang. Es enthält die IP-Version. Üblicherweise enthält dieses Feld im Moment die Zahl 4. Jedoch werden in der nahen Zukunft sicherlich vermehrt auch IPv6-Pakete auftreten. Diese enthalten dann hier die Zahl 6.

A.2.2 Header-Länge

Dieses Feld enthält die Länge des Headers. Es ist selbst 4 Bits lang. Jedoch wird die Länge nicht in Bits oder Bytes gemessen, sondern in Doppelworten. Ein Doppelwort entspricht 4 Bytes oder 32 Bits. Ein üblicher IP-Header ohne Optionen ist 20 Bytes lang. Daher befindet sich bei den meisten Paketen hier eine 5. Weist der Header weitere IP-Optionen auf (z. B. Source Routing, siehe auch Abschnitt A.2.13), so befindet sich hier entsprechend eine größere Zahl. Der Header kann maximal eine Länge von 15 (4 Bits) Doppelworten, also 60 Bytes, einnehmen.

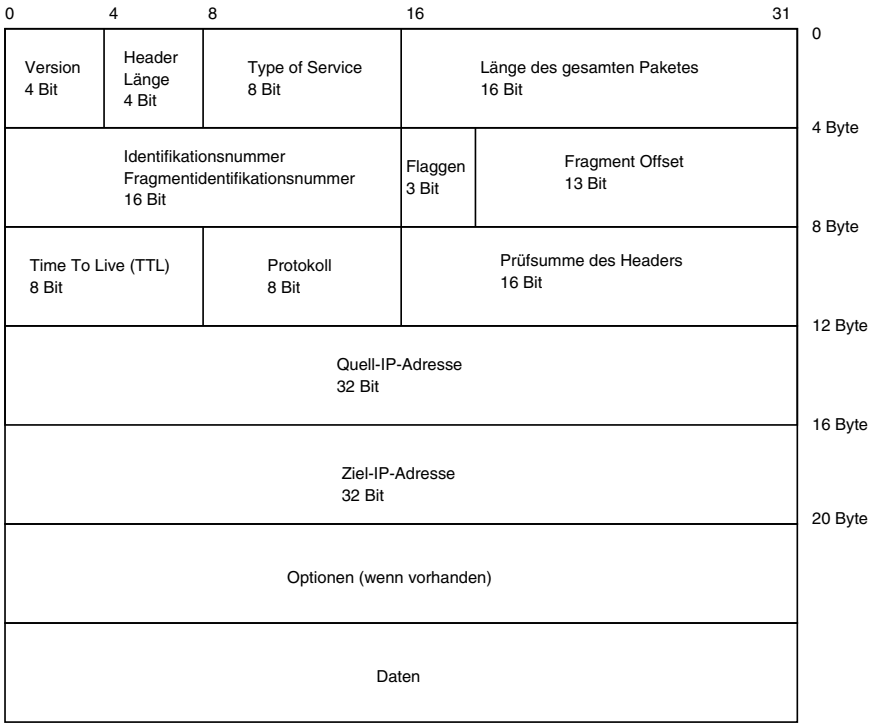


Abbildung A.2: **IP-Header Version 4**

A.2.3 Type of Service

Das Type-of-Service-Feld ist 8 Bits lang. Es steht seit der Entwicklung des IPv4-Protokolls zur Verfügung. Es wurde ursprünglich implementiert, um eine Art Quality-of-Service zu bieten. Es wird heutzutage nur von wenigen Anwendungen gesetzt. Die Anwendungen unter Linux nutzen es jedoch recht häufig. Dennoch unterstützen nur wenige Router im Internet seine Auswertung. Eine Verwendung durch die Anwendungen hat daher nur wenig Auswirkung auf den tatsächlichen Transport des Pakets. Es existieren die folgenden Werte: Minimize-Delay 16 (0x10), Maximize-Throughput 8 (0x08), Maximize-Reliability 4 (0x04), Minimize-Cost 2 (0x02) und Normal-Service 0 (0x00). Sie werden heute durch DiffServ abgelöst.

A.2.4 Paketlänge

Dieses Feld definiert die Gesamtpaketlänge in Bytes. Das Feld ist 16 Bit lang, daher kann ein Paket maximal 65.535 Byte lang sein. Üblicherweise sind die übertragenen Pakete wesentlich kleiner, da die Übertragungsmedien nicht in der Lage sind, derart große Pakete zu transportieren.

A.2.5 Identifikationsnummer

Jedes Paket wird üblicherweise mit einer eindeutigen 16 Bit langen Identifikationsnummer verschickt. Dies geschieht, damit im Falle einer Fragmentierung der Empfänger die Fragmente eines Pakets zuordnen kann. Daher wird in der Literatur häufig bei einem nicht fragmentierten Paket von der IP-Identifikationsnummer und bei einem fragmentierten Paket von der Fragment-Identifikationsnummer gesprochen.

Viele Systeme inkrementieren diese Zahl für jedes versandte Paket um 1. Hiermit sind jedoch gespoofte Portscans möglich (siehe Abschnitt 15.6). Daher existieren einige Betriebssysteme (z. B. OpenBSD), die diese Zahl zufällig vergeben. Der aktuelle Linux-Kernel setzt diesen Wert seit der Version 2.4 auf null, wenn das Paket gleichzeitig das DF-Flag (siehe auch Abschnitt A.2.6) gesetzt hat. In diesem Fall darf das Paket nicht fragmentiert werden, daher hat diese Zahl auch keinen Sinn.

A.2.6 Flaggen

Der IP-Header enthält drei Bits, die Informationen über die Fragmentierung des Pakets enthalten.

Das erste dieser drei Bits wird momentan nicht verwendet und muss immer null sein.

Das folgende Bit ist das Don't-Fragment-(DF-)Bit. Ist dieses Bit gesetzt (1), so darf das Paket nicht fragmentiert werden. Ein Router, der dieses Paket aufgrund seiner Größe nicht zustellen kann, muss das Paket verwerfen und eine ICMP-Fehlermeldung an den Absender schicken.

Das dritte und letzte Bit ist das More-Fragments-follow-(MF-)Bit. Dieses Bit zeigt an, dass weitere Fragmente folgen. Beim letzten Fragment und allen nicht fragmentierten Paketen ist dieses Bit gelöscht.

A.2.7 Fragment-Offset

Dieses Feld gibt bei einem Fragment dessen Beginn im Gesamtpaket an. Hierbei erfolgt die Angabe in Vielfachen von 8. Das bedeutet, dass ein Fragment (außer dem letzten) immer eine Länge aufweisen muss, die durch 8 ohne Rest teilbar ist. Das Feld ist 13 Bit lang und kann damit den ganzen Bereich von maximal 65.535 Bytes eines Pakets abdecken.

Der Empfänger verwendet diese Information, um das Paket in der richtigen Reihenfolge zusammenzusetzen.

Der Ping of Death nutzte Fragmente, um den TCP/IP-Stack einiger Betriebssysteme mit einem Bufferoverflow zum Absturz zu bringen. Es ist möglich, Fragmente so zu konstruieren, dass bei der Defragmentierung ein Paket entsteht, das größer 65.535 Bytes ist. Dies ist möglich, da das erste Fragment eine Größe von 1500 Bytes aufweist. Jedes weitere Fragment ist 1480 Byte lang. 1500 Bytes plus 43 mal 1480 Bytes ergeben 65.140 Bytes. Nun kann ein weiteres Fragment erzeugt werden, das erneut 1480 Byte lang ist. Erlaubt wären jedoch nur noch 395. Bei der Defragmentierung kommt es daher zum Bufferoverflow. Dieser Angriff erhielt den

Namen Ping of Death, da es besonders einfach war, mit dem Kommando `ping` diese Pakete zu erzeugen.

A.2.8 Time To Live (TTL)

Bei dem Feld Time To Live handelt es sich um ein 8 Bit langes Feld. Es kann somit Werte von 0 bis 255 aufnehmen. Dieser Wert wird von jedem Router, der das Paket weiterleitet, gelesen und pro Hop um 1 dekrementiert. Erreicht hierbei das Feld den Wert null, so muss der Router das Paket verwerfen und eine Fehlermeldung an den Absender zurücksenden. Ursprünglich sahen die Spezifikationen vor, dass der Wert je Sekunde Transport um 1 dekrementiert wird. Hieraus erklärt sich der Name „Time To Live".

INFO

Diese Funktion erlaubt es, mit dem Werkzeug `traceroute` die Route eines Pakets zu ermitteln (siehe auch Abschnitt 35.12). Hierzu werden Pakete mit steigenden TTL-Werten an den Zielrechner gesendet. Jeder Router wird entsprechende Pakete verwerfen müssen und eine Fehlermeldung an den Absender schicken. So wird die IP-Adresse eines jeden Routers ermittelt.

Verschiedene Betriebssysteme verwenden üblicherweise unterschiedliche Standard-TTL-Werte. Linux nutzt 64.

A.2.9 Protokoll

IP ist in der Lage, eine große Anzahl von Protokollen zu übertragen. Dies sind zum Beispiel ICMP (1), TCP (6) und UDP (17). Dieses Feld gibt die Nummer des enthaltenen Protokolls an. Unter Linux werden alle Protokolle und ihre Nummern in der Datei `/etc/protocols` aufgeführt.

Werden ungewöhnliche Protokolle im Netzwerk entdeckt, so kann es sich hierbei auch um einen Tunnel handeln. Das Honeynet Project hat ein Werkzeug entdeckt, das zum Beispiel einen NVP-(11-)Tunnel aufbaut, um hierüber die Verbindungen eines Angreifers zu verschleiern.

A.2.10 Prüfsumme

Dies ist die Prüfsumme des IP-Headers des Pakets. Hiermit können Netzwerkgeräte und auch der Empfänger die Validität des Paket-Headers bestimmen. Die Prüfsumme enthält nicht den Datenanteil des Pakets. Da einige Werte des Headers sich während der Übertragung ändern (z. B. TTL), wird diese Prüfsumme von jedem Netzwerkgerät (z. B. Router), das Änderungen durchführt, neu berechnet. Pakete mit fehlerhaften Header-Prüfsummen können daher nur im lokalen Netzwerk entstanden sein. Der Empfänger verwirft Pakete mit fehlerhaften Prüfsummen.

A.2.11 Quell-IP-Adresse

Dieses Feld enthält die IP-Adresse des Absenders. Wenn beim Transport des Pakets ein Source-NAT (Network Address Translation) durchgeführt wurde, befindet sich hier die entsprechende Adresse. Ein Source-NAT ist im Header nicht erkennbar.

A.2.12 Ziel-IP-Adresse

Dieses Feld enthält die IP-Adresse des Empfängers. Wenn beim Transport des Pakets ein Destination-NAT (Network Address Translation) durchgeführt wurde, befindet sich hier die entsprechende Adresse. Ein Destination-NAT ist im Header nicht erkennbar.

A.2.13 IP-Optionen

Sämtliche für den Transport des IP-Pakets erforderlichen Informationen werden in den 20 Bytes des IP-Headers gespeichert. Jedoch besteht gelegentlich die Notwendigkeit, zusätzliche Informationen zu senden, die das Verhalten des IP-Protokolls modifizieren. Diese werden als Option übertragen. Insgesamt können 40 Bytes Optionen übertragen werden. Standardmäßig werden keine Optionen verwendet.

Die Optionen bestehen aus einem Byte, das den Typ definiert, einem Byte für die Länge und entsprechenden Bytes für die Daten. Abbildung A.3 veranschaulicht das.

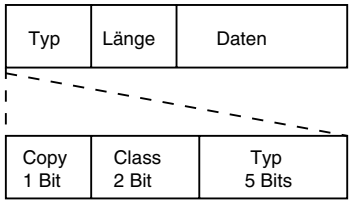


Abbildung A.3: **IP-Optionen-Aufbau**

End of Optionlist

Dies definiert das Ende der Optionenliste. Es ist eine Option der Klasse 0 und des Typs 0. Diese Option besitzt kein Längenfeld und kein Datenfeld.

No Operation

No Operation ist eine Option der Klasse 0 mit Typ 1. Sie wird verwendet, um die Optionenliste aufzufüllen. Die Länge des IP-Headers kann nur in Doppelworten angegeben werden. Wenn die Länge der Optionen nicht durch 4 teilbar ist, werden sie mit dieser Option aufgefüllt. Diese Option weist weder ein Längenfeld noch ein Datenfeld auf.

Security Options

Diese Option der Klasse 0 mit Typ 2 ist 88 Bits lang und wird für militärische Zwecke genutzt. Üblicherweise wird diese Option nicht im Internet beobachtet. Sie hat keine Verwandtschaft mit IPsec.

Record Route

Diese Option verwendet die Klasse 0 und den Typ 7. Router, die diese Option im Paket erkennen, sollen ihre IP-Adresse im IP-Header aufzeichnen. Da die Größe des IP-Headers beschränkt ist, können hier nur maximal acht Router ihre IP-Adressen eintragen. Daher wird diese Funktion nur selten im Internet genutzt. Stattdessen wird auf `traceroute` zurückgegriffen. Während diese Funktion jedoch für jedes Paket tatsächlich die Route protokolliert, zeigt `traceroute` nur die wahrscheinliche Route an.

Loose Source Routing

Loose Source Routing ist eine Option der Klasse 0 und mit Typ 3. Die Größe dieser Option hängt von der Anzahl der angegebenen Router ab. Loose Source Routing erlaubt es dem Absender, eine Liste von Routern im IP-Header anzugeben, über die das Paket neben anderen transportiert werden soll. Es können aus Platzgründen maximal acht Router angegeben werden. Verschiedene Befehle unterstützen diese Funktion: `traceroute`, `netcat` etc.

INFO

Loose Source Routing erlaubt es, Pakete zu routen, die ansonsten nicht geroutet werden würden. Es erlaubt zum Beispiel, auf den meisten Internet-Routern Pakete an RFC-1918-Netze (z. B. 192.168.0.0/24) zu routen. So kann ein Angreifer von außen ein Paket an diese Adresse senden und eine Loose Source Route angeben, damit die Internet-Router auch wissen, wohin das Paket gesendet werden soll. Die meisten Internet Router verwerfen heute Pakete, die diese Option nutzen.

Strict Source Routing

Strict Source Routing ist eine Option mit der Klasse 0 und mit Typ 9. Hier definiert die Liste die exakte Abfolge der zu verwendenen Router. Es dürfen keine weiteren Router verwendet werden.

Router Alert

Diese Option (Klasse 0, Typ 20) definiert, dass der Router das Paket modifizieren muss, bevor er es weiterroutet. Es wird für experimentelle Erweiterungen genutzt.

TimeStamp

Diese Option (Klasse 2, Typ 4) verlangt, dass der Router ähnlich der Record-Route-Option seine IP-Adresse im Header ablegt, aber zusätzlich noch die Zeit hinterlegt, zu der das Paket verarbeitet wurde.

Es besteht die Möglichkeit, nur die Timestamps oder Router-IP-Adressen und Timestamps anzufordern.

A.3 UDP

Das User Datagram Protocol (UDP, RFC 768, STD 6) stellt eines der beiden am häufigsten eingesetzten Protokolle auf Basis von IP dar. Die meisten Internetapplikationen verwenden TCP (siehe Abschnitt A.4). TCP garantiert die vollständige und korrekte Übertragung der Daten. Hierzu generiert es jedoch einen gewaltigen Overhead. Viele Anwendungen benötigen diese Garantie nicht oder können sie gar nicht nutzen, da es sich um Multicast- oder Broadcast-Anwendungen handelt, bei denen ein Paket gleichzeitig an mehrere Empfänger zugestellt wird. In diesen Fällen wird meist das UDP-Protokoll verwendet.

UDP ist ein unzuverlässiges und datagramm-orientiertes Protokoll. Es garantiert weder die Zustellung eines Datagramms noch bietet es Vorkehrungen gegen eine Duplizierung oder eine Vertauschung der Reihenfolge der Daten. Aufgrund seiner Unzuverlässigkeit bieten die höheren Protokolle meist eine gewisse Fehlerkontrolle. Dies äußert sich oft in einer gewissen Unempfindlichkeit gegenüber verlorenen Paketen. Wenn zum Beispiel ein Paket bei einer Videostreaming-Anwendung verloren geht, so macht sich das meistens nur durch ein leichtes Zittern der Darstellung bemerkbar. Eine TCP-ähnliche Fehlerkontrolle mit einer erneuten Sendung des Pakets würde meist zu einem Stottern und Anhalten des Streams führen.

UDP ist nur in der Lage, ein Datagramm gleichzeitig zu verarbeiten. Wenn UDP Informationen transportiert, so teilt es nicht die Daten auf mehrere Datagramme auf (TCP teilt die Daten entsprechend der MSS auf, siehe auch die Abschnitte A.14 und 35.3.1). Es erzeugt daher unter Umständen sehr große UDP-Datagramme, die anschließend von der darunterliegenden IP-Schicht auf IP-Fragmente aufgeteilt werden müssen.

Um mehreren Anwendungen die Verwendung des UDP-Protokolls zu ermöglichen, verwendet UDP einen Multiplexer. Client- wie Server-Anwendungen müssen sich vor der Verwendung des UDP-Protokolls registrieren. Während dieser Registrierung weist die UDP-Schicht diesen Anwendungen einen Port zu. Die Verwendung der Ports durch die verschiedenen Dienste ist grundsätzlich willkürlich, jedoch haben sich im Laufe der Jahre bestimmte Ports für bestimmte Dienste etabliert. Die Zuweisung der Ports zu den einzelnen Diensten erfolgt durch die Internet Assigned Numbers Authority (IANA), die die Liste der Well-Known Ports pflegt (`http://www.iana.org/assignments/port-numbers`).

Der UDP-Header (siehe Abbildung A.4) ist acht Byte lang. Er enthält den Quell- und den Zielport, die Datagrammlänge und eine Prüfsumme.

Der UDP-Header enthält keine IP-Adressen. Diese werden im IP-Header spezifiziert. Der UDP-Header stellt bei einem UDP-IP-Paket die ersten acht Bytes im Datenanteil des IP-Pakets dar.

Der UDP-Quellport (Source Port) ist wie der UDP-Zielport (Destination Port) 16 Bit oder 2 Byte lang. Die UDP-Protokollschicht verwendet diese Information, um die übertragenen Daten einzelnen Anwendungen zuzuordnen.

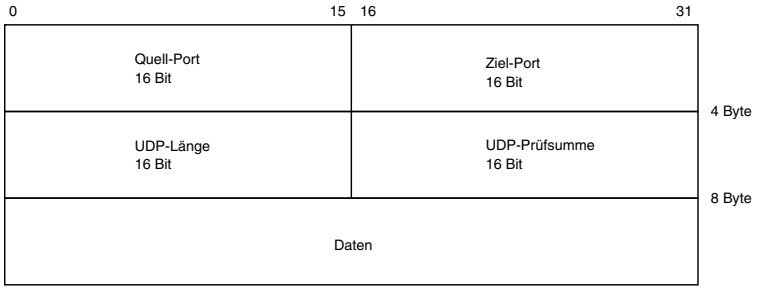


Abbildung A.4: **UDP-Header**

Die Länge des UDP-Datagramms wird ebenfalls im UDP-Header übertragen. Da ein UDP-Header mindestens acht Byte lang ist, ist die kleinste mögliche UDP-Nachricht acht Byte lang. Ein IP-Paket darf maximal 65.535 Byte lang sein. Abzüglich des IP-Headers von mindestens 20 Byte kann eine UDP-Nachricht maximal 65.515 Byte lang werden. Die Größe des UDP-Datagramms wird von der Anwendung bestimmt. Erzeugt diese Anwendung ein UDP-Datagramm, das größer ist als die MTU (Maximum Transmission Unit), so wird das Paket auf IP-Ebene fragmentiert.

Die UDP-Prüfsumme im UDP-Header ist optional. Das bedeutet, dass die Anwendung entscheiden kann, ob diese Prüfsumme berechnet werden soll oder nicht. Viele Anwendungen verzichten auf die Erzeugung einer Prüfsumme zugunsten der Performance. Wenn jedoch eine Prüfsumme erzeugt wurde, ist der Empfänger des Pakets laut RFC1122 verpflichtet, diese Prüfsumme zu überprüfen und im Zweifelsfall das Paket zu verwerfen. Bei der Berechnung der Prüfsumme wird der Inhalt des UDP-Datagramms zusammen mit einem Pseudo-Header aus Quell- und Zielport, der UDP-Protokollnummer 17 und der Größe als Eingabe verwendet.

Es existieren verschiedene Anwendungen, die das UDP-Protokoll nutzen. UDP wird zum Beispiel verwendet, um Namensauflösungen durchzuführen. Hier sendet der Client ein UDP-Paket an einen DNS-Server mit der Bitte, den enthaltenen Namen aufzulösen. Der DNS-Server sendet seine Antwort in einem UDP-Paket an den Client zurück. Da UDP kein verlässliches Protokoll darstellt, achtet der DNS-Server darauf, maximal 512 Bytes Daten in seinem UDP-Datagramm zurückzusenden. Dies garantiert, dass nicht durch eine mögliche Fragmentierung des Pakets Daten verloren gehen können. Ist die zu sendende Antwort jedoch größer, so sendet der DNS-Server eine trunkierte (*truncated*) Antwort. Der Client ist nun verpflichtet, den DNS-Server erneut zu kontaktieren und die Anfrage erneut mit dem TCP-Protokoll zu stellen. TCP ist in der Lage, die fehlerfreie und komplette Übertragung größerer Datenmengen zu garantieren.

A.4 TCP

Das Transmission Control Protocol (TCP, RFC793, STD 7) ist das im Internet hauptsächlich eingesetzte Protokoll. TCP garantiert die korrekte und vollständige Übertragung der Infor-

mationen. Hierzu setzt TCP unter anderem eine sehr intelligente Flussüberwachung und -steuerung ein. TCP ist ein verbindungsorientiertes Transportprotokoll für den Einsatz in paketvermittelten Netzen. Der häufigste Einsatz baut auf dem Internet-Protokoll IP auf. Es konzentriert sich auf die Verbindung und ihre Integrität. Hierzu bietet es die folgenden Dienste:

Virtuelle Verbindung. Die beiden TCP-Endpunkte kommunizieren über eine dedizierte virtuelle Verbindung. Diese ist für die Flusskontrolle, die garantierte Übertragung und das I/O-Management verantwortlich.

I/O-Management für die Anwendung. TCP bietet der Anwendung einen I/O-Puffer. Die Anwendung kann ihre Informationen als fortlaufende Daten in diesen Puffer schreiben beziehungsweise aus ihm lesen. TCP wandelt diesen fortlaufenden Strom anschließend in Pakete um. Nicht die Anwendung definiert die Paketgröße (wie bei UDP), sondern das TCP-Protokoll erzeugt die TCP-Segmente entsprechend der ausgehandelten Maximum Segment Size (MSS).

auf Netzwerkebene. TCP wandelt den Datenstrom der Anwendung in Segmente um. Hierbei werden die Segmente so erzeugt, dass sie effizient über das Netzwerk transportiert werden.

Flusskontrolle. TCP bietet eine fortgeschrittene Flusskontrolle, die die unterschiedlichen Sende- und Empfangsfähigkeiten der vielfältigen Geräte berücksichtigt. Es ist in der Lage, vollkommen transparent für die Anwendung die Geschwindigkeit der Verbindung optimal anzupassen.

Zuverlässigkeit. Jedes übertragene Byte wird mit einer Sequenznummer versehen. Dies ermöglicht eine Einordnung der übertragenen Daten in der richtigen Reihenfolge. Zusätzlich bestätigt der Empfänger mit dieser Nummer den Empfang der Daten. Stellt TCP fest, dass bestimmte Informationen nicht übertragen wurden, so werden sie transparent für die Anwendung automatisch erneut gesendet.

Zusätzlich bietet TCP wie UDP einen Multiplexer, sodass mehrere Anwendungen auf einem Rechner gleichzeitig das TCP-Protokoll verwenden können. Hier werden ebenfalls Ports eingesetzt. Jede Anwendung muss vor der Verwendung des Protokolls einen derartigen Port reservieren.

A.4.1 Auf- und Abbau einer TCP-Verbindung

Damit TCP die oben erwähnten Funktionen wahrnehmen kann, ist einiger Verwaltungsaufwand erforderlich. TCP muss zunächst eine Verbindung öffnen. Hierbei führen die beiden TCP-Kommunikationspartner eine Synchronisation ihrer Sequenznummern durch. Anschließend können die Daten ausgetauscht werden. Nach Nutzung der Verbindung sollte diese auch wieder korrekt abgebaut werden, sodass die beiden Kommunikationspartner die Verwaltungsstrukturen wieder freigeben können.

Abbildung A.5 zeigt schematisch den Auf- und Abbau einer TCP-Verbindung. Diese Abbildung führt bereits in einige Funktionen von TCP ein.

Der TCP-Handshake beginnt zunächst mit einem sogenannten SYN-Paket. TCP besitzt sechs Bits im TCP-Header (siehe Abschnitt A.10), die die Funktion des TCP-Pakets bestimmen. Es handelt sich hierbei um die Bits SYN, FIN, ACK, RST, URG und PSH. Ein SYN-Paket ist ein Paket, bei dem die eigene Sequenznummer an den Kommunikationspartner übermittelt wird. Dies ist für die Synchronisation der Verbindung erforderlich. Der Kommunikationspartner erhält hiermit die Information, dass die Datenzählung mit dieser Sequenznummer beginnt. Jedes übertragene Byte besitzt eine eigene Sequenznummer. In diesem Beispiel überträgt der Client die Sequenznummer 1234 an den Server. Diese initiale Sequenznummer wird für jede Verbindung neu bestimmt. Die Sequenznummern werden von dem Empfänger benötigt, um Pakete, die in unterschiedlicher Reihenfolge empfangen wurden, in der richtigen Reihenfolge zusammenzusetzen und den Verlust von Paketen zu erkennen.

Der Server beantwortet dieses Paket mit einem eigenen SYN/ACK-Paket. Das ACK- oder Acknowledge-Bit zeigt an, dass dieses Paket den Empfang von Daten bestätigt. Um der Gegenseite mitzuteilen, welche Daten bestätigt werden, übermittelt der Server eine Acknowledgement-Nummer: 1235. Diese Zahl definiert das nächste erwartete Byte von der Gegenstelle. Es entspricht also der Sequenznummer des letzten empfangenen Bytes plus 1. Mit dieser Bestätigung ist die Verbindung zwischen Client und Server in einer Richtung geöffnet worden. Man spricht von einer halb offenen Verbindung. In demselben Paket übermittelt jedoch der Server seinerseits seine Sequenznummer mit dem gesetzten SYN-Bit. Hiermit fordert er den Client auf, seine Seite mit dieser Nummer zu synchronisieren.

Eine Vollduplex-Verbindung, die in beide Richtungen Daten versenden kann, entsteht, wenn der Client diese Synchronisationsanfrage im dritten Paket bestätigt. Da der Client keine Daten übermittelt, erhöht er seine Sequenznummer nicht. Nun können Daten zwischen Client und Server ausgetauscht werden.

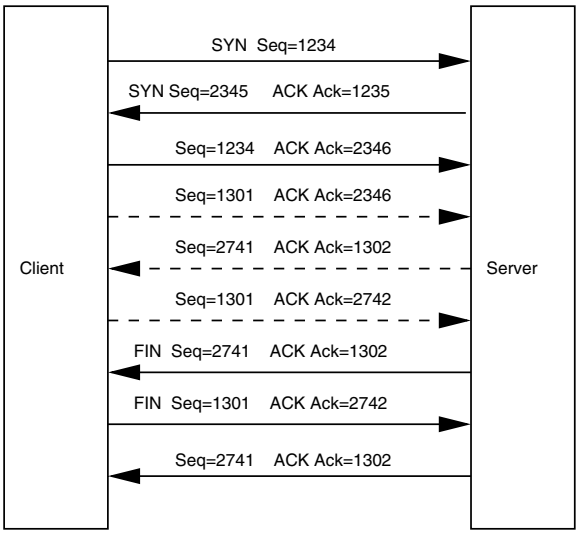


Abbildung A.5: **Der TCP-Handshake**

Der Client übermittelt nun zunächst 67 Bytes an den Server (4. Paket). Dies kann aus der Differenz der Sequenznummern des dritten und vierten Pakets errechnet werden. Anschließend bestätigt der Server den Empfang dieser Daten und sendet seinerseits 396 Bytes (5. Paket). Der Empfang wird erneut vom Client im 6. Paket bestätigt.

Nun beschließt der Server, dass die Verbindung beendet werden soll. Hierzu bestätigt er den letzten Empfang vom Client und setzt selbst das Bit `FIN`. Dieses Bit ist die Aufforderung, die Verbindung zu schließen. Hierbei werden keine Daten übermittelt. Daher wird die Sequenznummer nicht erhöht. Der Client bestätigt das `FIN`. Hiermit ist die Verbindung halb geschlossen. Um nun seine Richtung zu schließen, sendet der Client in demselben oder einem zusätzlichen Paket ebenfalls ein `FIN` an den Server. Dieser bestätigt dieses `FIN` ebenfalls in einem letzten Paket, und die Verbindung ist in beiden Richtungen geschlossen.

A.4.2 TCP-Header

Das TCP-Segment besteht ähnlich wie ein UDP-Datagramm aus einem Header und den Daten. Bevor nun das weitere Verhalten von TCP bezüglich der Flusskontrolle und der Zuverlässigkeit besprochen wird, soll kurz der TCP-Header mit seinen Informationen vorgestellt werden (siehe Abbildung A.6).

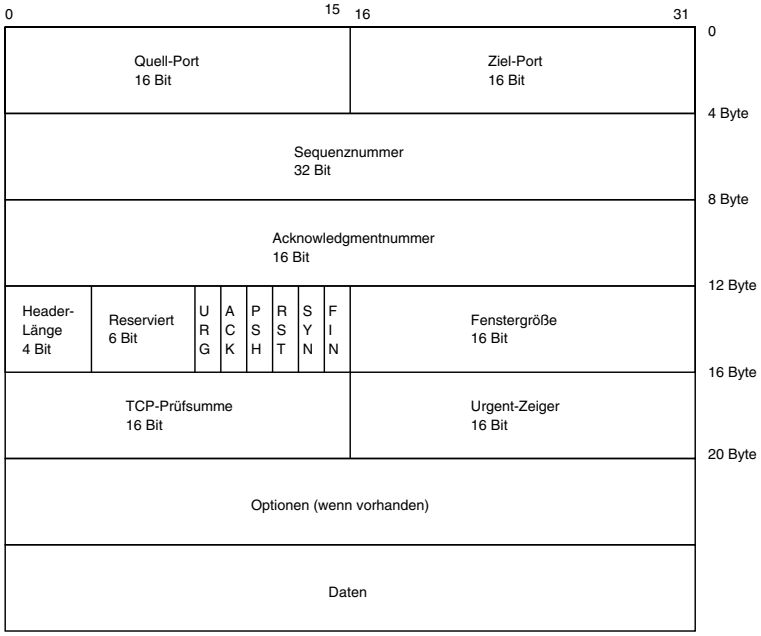


Abbildung A.6: **TCP-Header**

A.5 Quell- und Zielport

Die TCP-Ports werden vom TCP-Protokoll verwendet, um Multiplexer-Funktionalität zur Verfügung zu stellen. Das TCP-Protokoll ist hiermit in der Lage, mehrere Anwendungen gleichzeitig zu unterstützen. Hierzu bindet sich eine Anwendung zunächst an einen Port. Anschließend kann das TCP-Protokoll dieser Anwendung spezifisch ihre Daten zukommen lassen.

Das TCP-Protokoll unterstützt 65.536 Ports von 0 bis 65.535. Die Zuordnung der Ports zu den Applikationen ist grundsätzlich willkürlich. Jedoch wurden von der IANA einige Ports gewissen Diensten fest zugewiesen (s. UDP).

A.6 Sequenznummer

TCP garantiert die Zustellung der zu übertragenden Daten in der richtigen Reihenfolge. Damit die Kommunikationspartner diese Übertragung bestätigen und die Daten in der richtigen Reihenfolge zusammensetzen können, wird jedes übertragene Byte mit einer Sequenznummer versehen. Die Zuordnung der Daten und die Entscheidung über die Annahme der Daten erfolgt über diese Sequenznummer. Die initiale Sequenznummer sollte für jede Verbindung zufällig ermittelt werden. Besteht die Möglichkeit, die Sequenznummer von einer Verbindung zur nächsten vorherzusehen, so kann dies für gespoofte Angriffe auf das TCP-Protokoll genutzt werden.

Die Sequenznummer ist eine 32-Bit-Zahl.

A.7 Acknowledgement-Nummer

Wie bereits bei der Sequenznummer angesprochen, erhält jedes übertragene Byte eine eindeutige Sequenznummer. In dem Maße, in dem die Daten empfangen werden, übermittelt der Empfänger die Information an den Absender, dass er in der Lage ist, weitere Daten zu empfangen. Die Acknowledgement-Nummer enthält die Information, welches Byte als Nächstes erwartet wird. Alle Bytes mit einer kleineren Sequenznummer wurden bereits empfangen.

Sendet der Absender vier Pakete mit den Bytes 1–100, 101–200, 201–300 und 301–400, so antwortet der Empfänger entsprechend mit einer Acknowledgement-Nummer von 101, 201, 301 und 401. Geht das zweite Paket während der Übertragung verloren, so antwortet der Empfänger mit 101, 101, 101 und 101. Dies weist den Absender darauf hin, dass das zweite Paket erneut gesendet werden muss. Sendet der Absender anschließend das Paket 101–200 erneut, so bestätigt der Empfänger dies mit 401, da er die Daten 201–400 ebenfalls bereits erhalten hat.

Um dies noch zusätzlich zu optimieren, besteht die Möglichkeit, verzögerte oder selektive Bestätigungen des Empfangs zu versenden (siehe Abschnitt A.14).

A.8 Header-Länge

Dieses vier Bit lange Feld gibt die Länge des Headers in Doppelworten an. Die Länge in Bytes lässt sich aus dem Wert dieses Feld nach Multiplikation mit 4 ermitteln. Ein TCP-Header ist immer mindestens 20 Byte lang. Dieses Feld trägt also mindestens die Zahl 5. Maximal kann ein TCP-Header 60 Byte lang sein (15 mal 4). TCP definiert also nicht die Gesamtlänge des Pakets wie UDP, sondern nur die Länge des Headers. Die Gesamtlänge des Pakets lässt sich jedoch aus der Gesamtlänge des IP-Pakets minus der Länge des TCP-Headers ermitteln.

A.9 Reservierte Bits

Hierbei handelt es sich um sechs Bits, die bis vor Kurzem keine weitere Verwendung hatten. Diese Bits mussten daher bisher gelöscht sein. Viele Firewalls und Intrusion-Detection-Systeme lösen bei einer Verwendung dieser Bits einen Alarm aus. Viele Werkzeuge zur Erkennung von Betriebssystemen nutzen diese Bits, um entfernte Systeme zu untersuchen. Unterschiedliche Betriebssysteme reagieren meist unterschiedlich, wenn diese Bits gesetzt sind.

Seit einigen Jahren existieren jedoch Bemühungen, einige dieser Bits für die Explicit Congestion Notification (ECN) zu verwenden (RFC3168). Hierbei handelt es sich um einen Mechanismus, bei dem die Kommunikationspartner eine Verstopfung des Internets im Vorfeld erkennen und die Übertragungsrate automatisch anpassen, um eine tatsächliche Verstopfung zu vermeiden.

Da ECN sowohl den IP- als auch den TCP-Header betrifft, wird es in einem eigenen Abschnitt behandelt (siehe Abschnitt A.15.3).

A.10 TCP-Flags

Bei den TCP-Flags handelt es sich um sechs Bits. Diese Bits klassifizieren die übertragenen Daten. Es sind nur sehr wenige Kombinationen dieser Bits in einem gültigen Paket erlaubt. Sie haben die folgenden Funktionen:

- `URG`. Das `URG`-Bit (Urgent) wird verwendet, um der Gegenstelle mitzuteilen, dass das Paket wichtige Daten enthält, die sofort verarbeitet werden müssen. Um den Anteil der wichtigen Daten zu definieren, wird der Urgent-Zeiger verwendet (siehe Abschnitt A.13). Ist das URG-Bit nicht gesetzt, so ist der Wert des Zeigers zu ignorieren.
- `ACK`. Das `ACK`-Bit (Acknowledgement) wird verwendet, um den Empfang von Daten zu bestätigen. Jedes TCP-Segment einer Verbindung, außer dem ersten Segment und RST-Segmenten zum Abbruch einer Verbindung, muss dieses Bit gesetzt haben.
- `PSH`. Das `PSH`-Bit (Push) kennzeichnet Segmente, die von der sendenden Anwendung „gedrückt“ werden. Häufig müssen Anwendungen (z. B. telnet) nur sehr wenige Daten versenden. TCP würde zur Optimierung die Daten aber erst versenden, wenn ein komplettes Segment gefüllt wäre. Das PSH-Bit weist das TCP-Protokoll an, die Daten sofort zu versenden, da keine weiteren Daten in diesem Moment folgen.

» RST. Das RST-Bit (Reset) wird nur verwendet, wenn ein Fehler in einer Verbindung aufgetreten ist oder wenn der Wunsch nach einem Verbindungsaufbau abgelehnt wird.

» SYN. Das SYN-Bit (Synchronize) wird verwendet, um die eigene Sequenznummer zur Synchronisation an den Kommunikationspartner zu übermitteln. Dies erfolgt während des TCP-Handshakes (siehe Abschnitt A.4.1). Das SYN-Bit darf nur in diesem Moment gesetzt werden.

» FIN. Das FIN-Bit (Finish) wird gesetzt, wenn eine Seite der Kommunikation diese beenden will. Wenn die Gegenseite ebenfalls in der Lage ist, die Verbindung zu beenden, so antwortet sie ebenfalls mit einem FIN-Paket. Ein Paket mit gesetztem FIN-Bit ist nur in einer zuvor aufgebauten Verbindung gültig.

Grundsätzlich müssen alle TCP-Pakete das ACK-Bit gesetzt haben. Lediglich zwei Ausnahmen sind erlaubt. Hierbei handelt es sich um das erste Paket einer TCP-Verbindung, in dem lediglich das SYN-Bit gesetzt ist, und ein RST-Paket, das einen Fehler in einer TCP-Verbindung anzeigt und diese abbricht. Die weiteren Bits können in Kombinationen mit dem ACK-Bit vorkommen.

Kombinationen von SYN/FIN, SYN/RST oder RST/FIN sind jedoch nicht erlaubt. Netzwerk-Intrusion-Detection-Systeme ermitteln üblicherweise sämtliche Pakete mit fehlerhaften TCP-Flag-Kombinationen.

A.11 Receive Window

Das TCP-Fenster (Receive Window) gibt die Größe des Empfangsspeichers des sendenden Systems an. Diese Angabe wird von der TCP-Flusskontrolle verwendet. Die Größe dieses Feldes ist 16 Bit. Damit kann das Fenster maximal 64 Kbit groß werden. Da dies für Netze mit großer Bandbreite nicht ausreicht, existiert zusätzlich die Möglichkeit, die Window-Scale-Option zu verwenden. Hiermit können 30 Bits für die Angabe der Fenstergröße verwendet werden. Dies erlaubt Fenster bis zu 1 Gbyte Größe.

Wurden so viele Daten gesendet, dass dieses Empfangsfenster gefüllt ist, so muss der Sender zunächst auf eine Bestätigung dieser Daten warten, bevor weitere Daten gesendet werden dürfen.

A.12 TCP-Prüfsumme

Dieses Feld speichert die TCP-Prüfsumme. Diese Prüfsumme erstreckt sich sowohl auf den Header als auch auf die Daten des TCP-Segments. Zusätzlich wird ein Pseudo-Header mit aufgenommen, der die IP-Adressen und das Protokoll (TCP, 6) enthält.

Die TCP-Prüfsumme ist 16 Bit lang und nicht optional. Der Absender muss die TCP-Prüfsumme berechnen, und der Empfänger muss diese Prüfsumme kontrollieren. Wenn die Prüfsumme nicht gültig ist, wird das Paket verworfen. Der Absender erhält keinerlei Fehlermeldung.

A.13 Urgent-Zeiger

TCP ist in der Lage, bestimmte Teile der Nachricht als wichtig zu kennzeichnen. Hierzu wird das URG-Flag in den TCP-Flags gesetzt. Zusätzlich wird der Urgent-Zeiger gesetzt. Der Urgent-Zeiger (oder Pointer) zeigt auf das Ende der wichtigen Informationen im aktuellen Segment.

Pakete, die das URG-Bit gesetzt haben, müssen vom Empfänger sofort bearbeitet werden. Dieses Paket sollte Vorrang vor allen weiteren Paketen in der Empfangswarteschlange haben.

Wenn der Urgent-Zeiger gesetzt, das URG-Bit jedoch gelöscht ist, muss das Paket wie ein normales Paket behandelt werden.

A.14 TCP-Optionen

Die bisher im TCP-Header spezifizierten Angaben genügen für eine erfolgreiche TCP-Verbindung. Jedoch werden häufig zusätzliche Optionen genutzt, um eine Anpassung der Eigenschaften zu ermöglichen.

TCP unterstützt die folgenden sieben Optionen:

End of Option List. Diese Option markiert das Ende der Optionen im TCP-Header. Die Option ist acht Bit lang und hat den Typ 0.

No Operation. Diese Option wird verwendet, um Bereiche zwischen den TCP-Optionen aufzufüllen. Es ist sinnvoll, dass einige Optionen auf einer 32-Bit-Grenze beginnen. Dann wird der Bereich zwischen diesen Optionen mit NOP aufgefüllt. NOP ist acht Bit lang und vom Typ 1.

Maximum Segment Size. Die Maximum Segment Size (MSS) wird von den Endpunkten verwendet, um die Gegenseite über ihre MTU (Maximum Transmission Unit) beziehungsweise MRU (Maximum Receive Unit) zu informieren. Diese Funktion kann auch von Routern gesetzt werden. Netfilter besitzt zum Beispiel ein TCPMSS-Target. Dieser Austausch erfolgt lediglich bei der Synchronisation der Verbindung. Die MSS ist üblicherweise gleich der MTU minus 40 Bytes.

Es handelt sich um eine Option von 32 Bit Länge und Typ 2. Wird keine MSS angegeben, so verlangt RFC1122, dass als MSS 536 (IP-Default-Größe 576 – Header 40) verwendet wird.

Window-Scale. Hiermit ist es möglich, größere Empfangsfenster anzugeben als die üblichen 64 Kbyte. Maximal sind Fenster in der Größenordnung von 1 Gbyte möglich. Diese Option von Typ 3 ist drei Byte lang. Das dritte Byte definiert den Maßstab des im Header angegebenen Empfangsfensters. Die Window Scale-Option darf lediglich in den beiden ersten Paketen ausgetauscht werden. Ansonsten wird sie ignoriert.

Selective Acknowledgment Permitted. Bevor selektive Bestätigungen erlaubt sind, müssen beide Endpunkte sich darauf einigen. Die Option Selective Acknowledgment Permitted

vom Typ 4 ist 16 Bit lang. Sie muss ebenfalls in den ersten beiden TCP-Segmenten ausgetauscht werden. Später wird sie ignoriert.

Selective Acknowledgment Data. Hiermit besteht für den Empfänger die Möglichkeit, einen unterbrochenen Strom von Daten zu bestätigen. Normalerweise kann der Empfänger nur das letzte Byte eines ununterbrochenen Datenstroms bestätigen. Dass der Empfänger bereits weitere Pakete erhalten hat, kann er dem Absender nicht mitteilen. Mit dieser Option Selective Acknowledgment Data vom Typ 5 und einer variablen Länge ist der Empfänger in der Lage, exakt ein fehlendes Segment nachzufordern und die bereits empfangenen diskontinuierlichen Segmente dem Absender mitzuteilen.

Diese Option darf in jedem TCP-Segment verwendet werden.

Timestamp. Die Option Timestamp vom Typ 8 erlaubt den beiden Endpunkten, die Latenzzeit kontinuierlich zu messen. Diese Option mit einer Länge von 10 Byte enthält die Zeitstempel beider Endpunkte in jedem Paket.

Diese Option darf in jedem TCP-Segment verwendet werden.

A.15 Fortgeschrittene Eigenschaften von TCP

A.15.1 Flusskontrolle

Wenn eine Anwendung unter Verwendung des TCP-Protokolls Daten versendet, so schreibt sie die Daten in einen Sendepuffer. TCP wird in regelmäßigen Abständen die Daten dieses Sendepuffers in TCP-Segmenten versenden. Die Senderate wird hierbei ständig angepasst.

Ursprünglich wurde hierzu die Möglichkeit geschaffen, dass der Empfänger die Geschwindigkeit über sein Empfangsfenster (Receive Window Sizing) anpasst. Dazu übermittelt der Empfänger in jedem Paket die maximale Datenmenge, die er im Moment zu verarbeiten in der Lage ist. Der Sender darf nicht mehr Daten als die vorgeschriebene Menge übertragen. Anschließend muss der Sender zunächst auf eine Bestätigung des Empfangs und der Verarbeitung warten.

Das Empfangsschiebefenster (Sliding Receive Window) erlaubt es dann dennoch dem Sender, einige weitere Pakete zu versenden, da er eine Bestätigung der bereits gesendeten Pakete in Kürze erwartet. Hierdurch ist ein wesentlich reibungsärmerer Austausch der Daten möglich.

Dies stellt jedoch eine rein vom Empfänger gesteuerte Flusskontrolle dar. Benötigt der Empfänger mehr Zeit zur Verarbeitung der Daten, so kann er sein Empfangsfenster verkleinern. Ist er in der Lage, die Daten schnell zu verarbeiten, so kann er es derart vergrößern, dass die Geschwindigkeit nur noch durch das Netzwerk selbst gesteuert wird. Dies reicht jedoch nicht aus. Es ist auch eine durch den Sender gesteuerte Flusskontrolle sinnvoll.

Die vom Sender gesteuerte Flusskontrolle verwendet ein Congestion Window (Verstopfungsfenster), den langsamen Start (Slow Start) und die Verstopfungsvermeidung (Congestion Avoidance).

Lediglich der Sender ist in der Lage, Verstopfungen zu erkennen. Hierzu existieren drei Möglichkeiten:

1. Der Sender erhält eine ICMP-*source-quench*-Meldung eines Routers. Dies zeigt an, dass der Router nicht in der Lage ist, die Pakete schnell genug zu verarbeiten.
2. Der Sender erhält mehrfache Acknowledgements mit identischer Acknowledgement-Nummer. Man bezeichnet diese auch als doppelte Acknowledgements. Der Empfänger sendet ein Acknowledgement, wenn er ein weiteres Paket erhält. Wenn ihm jedoch ein Paket fehlt, versendet er alle diese Acknowledgements mit derselben Acknowledgement-Nummer. Die Acknowledgement-Nummer zeigt das nächste vom Empfänger erwartete Datenbyte an! Daher sind für den Sender doppelte Acknowledgements ein Hinweis darauf, dass wahrscheinlich ein Paket zwischendurch verloren gegangen ist.
3. Der Sender erhält kein Acknowledgement innerhalb einer bestimmten Zeit (Acknowledgment Timer). Dies weist ebenfalls auf verloren gegangene Pakete hin.

Wenn nun der Sender weiterhin die Pakete mit gleicher Geschwindigkeit sendet, wird die Verstopfung bestehen bleiben und möglicherweise schlimmer werden. Es ist also erforderlich, dass der Sender reagiert und die Rate senkt, um die Verstopfung zu beheben und schließlich den alten Paketdurchsatz wieder zu erreichen.

Hierzu wird ein Congestion Window verwendet. Dieses definiert, wie viele Daten der Sender ohne eine Bestätigung der Gegenseite versenden darf. Zu Beginn weist dieses Fenster die gleiche Größe auf wie das Empfangsfenster (Receive Window). Dieses Fenster wird nun in Abhängigkeit von der Verstopfung reduziert.

» Wurden mehr als drei doppelte Acknowledgements erkannt, so wird das Congestion Window halbiert. Anschließend wird die Congestion Avoidance aktiviert. Diese vergrößert das Fenster wieder in sehr kleinen Schritten.

» Wurde eine Source-Quench-Meldung erhalten oder fehlen Acknowledgements, so wird das Fenster so stark reduziert, dass jeweils nur ein Segment gesendet werden kann. Anschließend wird der Slow Start aktiviert.

Der Slow Start vergrößert das Congestion Window exponentiell. Würde das Congestion Window sofort wieder auf den Ausgangswert reinitialisiert, so würde die Verstopfung sofort wieder auftreten. Beim Slow Start wird das Congestion Window für jedes bestätigte Segment um ein weiteres Segment vergrößert. Diese Technik wird verwendet, wenn eine neue Verbindung aufgebaut wird und eine Verstopfung aufgetreten ist. Im Falle einer Verstopfung wird jedoch beim Erreichen des halb maximalen Congestion Windows auf Congestion Avoidance umgeschaltet.

Congestion Avoidance stellt eine langsamere, vorsichtigere Methode zur Vergrößerung des Congestion Windows dar. Wurden alle Pakete, die innerhalb eines Congestion Windows versandt wurden, bestätigt, so wird das Congestion Window um ein Segment vergrößert.

A.15.2 Zuverlässigkeit

Die Zuverlässigkeit ist eine der wichtigsten Eigenschaften des TCP-Protokolls. Daher soll hier kurz beschrieben werden, welche Mechanismen von TCP verwendet werden, um diese effizient garantieren zu können.

RFC 793 verlangt, dass TCP in der Lage ist, Daten, die beschädigt, verloren, dupliziert oder in falscher Reihenfolge übermittelt wurden, so zu handhaben, dass dies transparent für die Anwendung erfolgt. Um dieses Ziel zu erreichen, verwendet TCP Prüfsummen, Sequenznummern, Acknowledgement-Nummern und Zeitgeber.

Die TCP-Prüfsumme ähnelt der UDP-Prüfsumme. Im Gegensatz zu UDP ist sie bei TCP jedoch obligatorisch. Hierbei werden zusätzlich zum TCP-Header und zu den Daten die IP-Adressen und das IP-Protokoll zur Ermittlung der Prüfsumme verwendet. Diese Prüfsumme garantiert die fehlerfreie Übertragung der Daten.

Die Sequenznummern ermöglichen es dem Empfänger, die Daten in der richtigen Reihenfolge zu verarbeiten, auch wenn die Daten möglicherweise in einer anderen Reihenfolge erhalten wurden. Jedes übertragene Byte besitzt seine eigene eindeutige Sequenznummer. Diese Sequenznummern erlauben auch die Erkennung duplizierter Informationen, da diese identische Sequenznummern aufweisen. Um eine Störung durch veraltete Pakete beim Empfang zu vermeiden, sollte der Empfänger nur Pakete mit Sequenznummern verarbeiten, die seinem Empfangsfenster (Receive Window) entsprechen.

Die Acknowledgement-Nummern erlauben es dem Absender, den korrekten Empfang der gesendeten Daten zu prüfen. Der Empfänger bestätigt den Empfang der Daten und bestätigt, dass er in der Lage ist, weitere Daten zu verarbeiten. Wurden gesendete Daten nicht mit der entsprechenden Acknowledgement-Nummer bestätigt, so werden diese Daten nach Ablauf des entsprechenden Zeitgebers erneut versendet.

Da häufig nur wenige Pakete verloren gehen, wurde mit dem RFC1072 die Möglichkeit geschaffen, selektive Bestätigungen (Selective Acknowledgements) zu versenden. Meist hat der Empfänger bereits zehn Segmente erhalten, jedoch fehlt das dritte Segment. Ein klassisches Verhalten des Empfängers, bei dem dieser mehrfach nur dieselbe Acknowledgement-Nummer versendet, führt häufig dazu, dass sämtliche Pakete ab dem dritten erneut versendet werden. Dies ist jedoch nicht erforderlich und beinträchtigt die Bandbreite. Die Selective Acknowledgements erlauben es mithilfe der TCP-Option Selective Acknowledgement, trotz doppelter Acknowledgement-Nummern weitere Segmente selektiv zu bestätigen.

Wenn der Datendurchsatz sehr hoch ist, ist es sinnvoller, nicht jedes Paket zu bestätigen, sondern die Bestätigung verzögert zu versenden (Delayed Acknowledgements). Hierbei wird nur der Empfang jedes zweiten oder dritten Pakets bestätigt. Dies stellt kein Problem dar, da ein Acknowledgement bedeutet, dass alle Daten bis zu dieser Acknowledgement-Nummer empfangen wurden.

A.15.3 Explicit Congestion Notification

TCP ist in der Lage, bereits sehr gut mit einer Netzwerkverstopfung umzugehen und trotz Verstopfung die Datenübertragung zuverlässig zu garantieren. Jedoch kann es weiterhin zu einer Verstopfung und damit auch zu einer Verzögerung der Übertragung kommen. Die Explicit Congestion Notification (ECN) versucht nun, dies zu verhindern, indem die Kommunikationspartner bereits vor dem Auftreten der Verstopfung gewarnt werden und dem Entstehen entgegenwirken können.

Die Erweiterung des IP-Protokolls um die Explicit Congestion Notification wird in RFC3168 beschrieben. Linux ist eines der ersten Betriebssysteme, die dieses RFC umgesetzt haben (siehe Abschnitt 23.3.6). Auch Windows Server 2008, Windows 7 und Windows Vista unterstützen ECN für TCP, aber es ist auch hier per Default abgeschaltet. Unter Windows können Sie es mit dem folgenden Befehl einschalten:

```
netsh interface tcp set global ecncapability=enabled
```

Die bisher besprochenen Verfahren des TCP-Protokolls zur Vermeidung einer Netzwerkverstopfung gehen davon aus, dass es sich beim Netzwerk um eine Black Box handelt. Das Netzwerk selbst ist nicht in der Lage, eine Verstopfung anzuzeigen. Die Verstopfung zeichnet sich dadurch aus, dass keine Pakete mehr transportiert werden.

Moderne Netzwerke und ihre Komponenten sind jedoch wesentlich intelligenter und sehr wohl in der Lage, eine Verstopfung bereits in ihrem Entstehen zu erkennen. Ein Router ist bei aktiver Verwaltung (Random Early Detection, RED) seiner Warteschlangen in der Lage, eine Verstopfung zu erkennen, bevor die Warteschlange überläuft und der Router die Pakete verwerfen muss. Er ist in der Lage, diese Informationen an die Endpunkte einer Kommunikation zu übermitteln, wenn die Protokolle dies vorsehen und verstehen. Dies erfolgt durch die Verwendung des Congestion-Experienced-Bits (CE).

Damit dies möglich ist, müssen das IP-Protokoll und das Transport-Protokoll (TCP) dies auch unterstützen. Hierzu wird im IP-Header ein ECN-Feld definiert und in dem TCP-Header zwei neue ECN-Flags. Diese Definition erlaubt eine fließende Migration, da diese Felder von Systemen, die nicht ECN-fähig sind, ignoriert werden. Leider werden Pakete, die diese bisher reservierten Felder (siehe Abschnitt A.9) verwenden, von vielen Firewalls und Network-Intrusion-Detection-Systemen noch als gefährlich eingestuft.

Das ECN-Feld im IP-Header ist zwei Bits lang. Ist dieses Feld gelöscht, so ist der Absender des Pakets nicht ECN-fähig. Tragen die beiden Bits den Wert 01 oder 10, so ist der Absender des Pakets ECN-fähig. Trägt dieses Feld den Wert 11, so hat ein Router dies gesetzt, da er eine Congestion bemerkt hat: Congestion Experienced. Das ECN-Feld entspricht den bisher ungenutzten Bits 6 und 7 des Type-of-Service-Feldes (siehe Abschnitt A.2.3). Die ehemalige Verwendung des Type-of-Service-Feldes wird nun durch die Differentiated Services ersetzt. Diese nutzen die Bits 0 bis 5 dieses Feldes.

Erhält ein Endpunkt ein Paket, bei dem Congestion Experienced (CE) gesetzt ist, so muss dieser Endpunkt sich so verhalten, als ob das Paket verloren gegangen wäre. Im Falle von TCP muss das TCP-Protokoll das Congestion Window halbieren.

ECN benötigt eine Unterstützung durch das Transport-Protokoll. Dieses muss zunächst die ECN-Fähigkeit der Endpunkte aushandeln. Anschließend sollten die Endpunkte jeweils Informationen über CE-Pakete austauschen.

Insgesamt sind drei Funktionalitäten für ECN in TCP erforderlich. Hierbei handelt es sich erstens um die Aushandlung der ECN-Fähigkeit zwischen den beiden Endpunkten. Zusätzlich ist ein ECN-Echo erforderlich, mit dem der Empfänger eines CE-Pakets dem Absender diese

Tatsache mitteilt, und ein Congestion-Window-Reduced-(CWR-)Flag, mit dem der Absender dem Empfänger mitteilt, dass das Congestion Window reduziert wurde.

Diese Fähigkeiten werden in TCP mit zwei neuen Flags realisiert. Hierbei handelt es sich um zwei bisher reservierte Bits im TCP-Header. Das Bit 9 der Bytes 13 und 14 im TCP-Header ist das ECN-Echo-Bit (ECE). Das CWR-Bit ist das Bit 8 im TCP-Header. Abbildung A.7 zeigt den veränderten Header.

Während des Verbindungsaufbaus sendet ein ECN-fähiger Rechner nun ein TCP-`SYN`-Paket, bei dem das ECE- und das CWR-Bit gesetzt sind. Ein ECN-fähiger Rechner kann dies mit einem TCP-`SYN`/`ACK`-Paket beantworten, bei dem das ECE-Bit gesetzt und das CWR-Bit gelöscht ist. Anschließend kann ECN genutzt werden.

Weitere Informationen finden Sie im RFC3168.

Der Linux-Kernel bietet die Möglichkeit, die ECN-Funktionalität ein- oder abzuschalten. Um die Funktionalität einzuschalten, genügt:

```
# sysctl -w net.ipv4.tcp_ecn=1
```

Das Abschalten erfolgt über die Zuweisung einer 0.

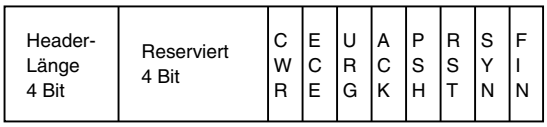


Abbildung A.7: **ECN-Header**

A.16 ICMP

IP ist ein Protokoll, das die Zustellung des Pakets nicht garantiert. Dies ist die Aufgabe der höheren Protokolle. Diese müssen beim Verlust eines Pakets diesen bemerken und das Paket erneut senden. Jedoch können Situationen auftreten, in denen kein Paket zum Ziel übertragen wird. In diesem Fall sollte der Absender informiert werden, um dauernde Neuübertragungen zu vermeiden oder um die Pakete modifiziert zu versenden. Es ist die Aufgabe des Internet Control Message Protocol (ICMP, RFC792, STD 5), diese Informationen zu übertragen.

INFO

ICMP kann eingesetzt werden, um das Betriebssystem eines Rechners zu bestimmen. Ofir Arkin und Fyodor Yarochkin haben das Werkzeug X entwickelt, das mit einigen wenigen ICMP-Paketen und den Antworten ermitteln kann, mit welchem Betriebssystem es sich gerade unterhält. Das Werkzeug und Whitepapers sind verfügbar unter `http://www.sys-security.com/html/projects/X.html`.

ICMP ist das IP-Protokoll Nummer eins. Es wird in einem IP-Datagramm übertragen. Der ICMP-Header besteht aus dem acht Bit langen ICMP-Type-Feld, dem acht Bit langen ICMP-Code-Feld und einer 16 Bit langen Prüfsumme. Anschließend können Daten in Abhängigkeit vom Typ und Code angehängt werden.

Die folgenden ICMP-Typen und -Codes sind definiert:

Nachricht	Type	Code
Echo reply	0	0
Destination unrechable	3	
Network unreachable	3	0
Host unreachable	3	1
Protocol unreachable	3	2
Port unreachable	3	3
Fragmentation needed but DF set	3	4
Source route failed	3	5
Destination network unkown	3	6
Destination host unkown	3	7
Source host isolated (obsolete)	3	8
Destination network admin. prohibited	3	9
Destination host admin prohibited	3	10
Network unreachable for TOS	3	11
Host unreachable for TOS	3	12
Communication admin prohibited	3	13
Host precedence violation	3	14
Precedence cutoff in effect	3	15
Source quench	4	0
Redirect	5	
Redirect for network	5	0
Redirect for host	5	1
Redirect for TOS and network	5	2
Redirect for TOS and host	5	3
Echo request	8	0
Router advertisment	9	0
Router solicitation	10	0
Time exceeded	11	
Time exceeded during transit	11	0
Time exceeded during reassembly	11	1
Parameter problem	12	
IP header bad	12	0
Required option missing	12	1
Timestamp request	13	0
Timestamp reply	14	0
Information request (obsolete)	15	0
Information reply (obsolete)	16	0
Address mask request	17	0
Address mask reply	18	0

Die wichtigsten dieser Nachrichten sollen im Weiteren erläutert werden.

A.16.1 *destination-unreachable*

destination-unreachable ist eine der wichtigsten ICMP-Nachrichten. Sie wird verwendet, um dem Absender mitzuteilen, dass der Empfänger nicht erreichbar ist. Die Nachricht *destination-unreachable* verwendet verschiedene Subtypen, die über den ICMP-Code unterschieden werden.

Die häufigsten Subtypen sind: *network-unreachable*, *host-unreachable* und *port-unreachable*. Die Nachricht *network-unreachable* wird versendet, wenn ein Router keine Route für das Zielnetzwerk kennt. Die Nachricht *host-unreachable* wird vom letzten Router versendet, wenn er den Zielrechner nicht erreichen kann (z. B. weil er ausgestellt ist). *port-unreachable* wird vom Zielrechner verwendet, wenn das Paket versucht, einen nicht existenten UDP-Dienst auf dem Zielrechner anzusprechen. Handelt es sich um einen TCP-Dienst, so versendet der Zielrechner ein TCP-`RST`-Paket.

Firewalls in Form eines Paketfilters verwenden ebenfalls häufig diese Meldungen, um einen Zugriff auf bestimmte Rechner und Dienste abzulehnen. Einfache Paketfilter können so auch erkannt werden, da sie häufig TCP-Anfragen mit einem ICMP-*port-unreachable* beantworten. Das Zielsystem hätte ein TCP-Reset geschickt.

Die Meldung „Fragmentation Needed but DF Bit set" (*fragmentation-needed*) wird verwendet, wenn ein Router das Paket nicht weiter senden kann, da es für das nächste Netzwerk zu groß ist. Zusätzlich wird dann die MTU des nächsten Netzwerks mit der Fehlermeldung übertragen. Diese Fehlermeldung wird von der Path Maximum Transmission Unit Discovery (Path MTU Discovery) eingesetzt. Hierbei versucht das Betriebssystem, eine Fragmentierung der Pakete zu vermeiden, indem es zunächst die MTU für den gesamten Pfad ermittelt. Anschließende Pakete werden dann mit dieser PMTU versandt.

Damit der Empfänger der Fehlermeldung *destination-unreachable* erfährt, auf welches Paket sich die Fehlermeldung bezieht, enthält diese den IP-Header des originalen Pakets mit den ersten acht folgenden Bytes des Pakets. Dies erlaubt eine eindeutige Zuordnung des Pakets durch den Empfänger.

STOP

Viele Paketfilter erlauben die Definition einer Network Address Translation (NAT). Das bedeutet, dass die Adressen der Pakete im IP-Header modifiziert werden. Einige Paketfilter kontrollieren jedoch nicht die IP-Adressen, die im IP-Header enthalten sind, der in der ICMP-Meldung eingebettet ist. So können private IP-Adressen nach außen gelangen!

A.16.2 *source-quench*

Dies ist eine sehr einfache Fehlermeldung. Mit ihr teilt der Absender mit, dass er die Pakete nicht schnell genug verarbeiten kann und einige Pakete verwerfen muss. Diese Meldung wurde früher auch von Routern versendet. Diese verwenden jedoch heutzutage modernere

Quality-of-Service-Funktionen. *source-quench*-Meldungen tauchen daher nur noch recht selten auf.

STOP

source-quench-Meldungen können für einen Denial-of-Service-Angriff genutzt werden, wenn sie geeignet gespooft werden.

A.16.3 *time-exceeded*

Die *time-exceeded*-Meldungen werden in erster Linie heute vom Werkzeug `traceroute` verursacht. Dieses Werkzeug versendet Pakete mit steigendem Time-To-Live-(TTL-)Wert an den Empfänger. Die Router, über die die Pakete zum Ziel transportiert werden, werden die entsprechenden Pakete verwerfen und Fehlermeldungen mit ihrer Absender-IP-Adresse zurücksenden. So kann der Weg des Pakets rekonstruiert werden.

Es existieren zwei wesentliche Varianten des Programms Traceroute: `traceroute` (UNIX) und `tracert.exe` (Microsoft Windows). Diese Programme versenden unterschiedliche Pakete. `tracert.exe` versendet an die Ziel-IP-Adresse jeweils drei Echo-Request-Pakete mit identischer TTL und inkrementiert dann den TTL-Wert. Die Rücklaufzeit der Pakete wird gemessen und angezeigt. Das UNIX-Programm `traceroute` versendet UDP-Pakete an die Ports 33434 ff. Hierbei werden ebenfalls immer drei Pakete mit identischer TTL an einen Port gesendet. Jedes Mal, wenn das Programm die TTL inkrementiert, wird auch der Port inkrementiert (siehe auch Abschnitt 35.12).

A.16.4 *redirect*

Die Redirect-Nachrichten werden von einem Router versendet, der den Absender eines Pakets über einen kürzeren Pfad informieren möchte. Router können ihre Routing-Tabellen dynamisch untereinander zum Beispiel mit dem Routing Information Protocol (RIP) austauschen. Hierdurch kennt ein Router alle weiteren Router und sämtliche möglichen Routen. Erhält ein Router nun ein Paket und stellt fest, dass sich in demselben Netzwerk ein weiterer, besser geeigneter Router befindet, so übermittelt er diese Information an den Client. Der speichert die Information in seinem Routing-Cache ab und wird das nächste Paket direkt an diesen geeigneteren Router versenden. Der Routing-Cache kann unter Linux mit dem Befehl `route -Cn` betrachtet werden.

STOP

Redirect-Meldungen können verwendet werden, um Router zu spoofen. Wenn ein Netzwerk keine dynamischen Routing-Protokolle einsetzt, sollten derartige Meldungen starkes Misstrauen hervorrufen. Unter Linux kann die Annahme derartiger Meldungen mit der Kernel-Variablen `/proc/sys/net/ipv4/conf/*/accept_redirects` abgestellt werden.

A.16.5 *parameter-problem*

Die Fehlermeldung *parameter problem-ICMP* wird versendet, wenn das IP-Datagramm selbst Fehler aufweist. Meistens wurde eine IP-Option falsch verwendet. Da die meisten IP-Implementierungen inzwischen jedoch recht ausgereift sind, kommen diese Fehlermeldungen eigentlich nur noch gehäuft vor, wenn Pakete manuell fehlerhaft konstruiert werden.

A.16.6 *echo-request* und *echo-reply*

Echo-Request und -Reply sind die ICMP-Nachrichten, die vom Kommando `ping` verwendet werden. Hierbei sendet ein Rechner den ICMP-Echo-Request. Der Empfänger antwortet auf jedes Echo-Request-Paket mit einem Echo-Reply-Paket. Um die Pakete voneinander trennen zu können, enthalten sie eine eindeutige Identifikationsnummer und eine Sequenznummer. Die Identifikationsnummer wird verwendet, um die Pakete mehrerer gleichzeitiger `ping`-Aufrufe voneinander zu trennen. Die Sequenznummer wird für jedes versandte Paket inkrementiert und identifiziert die einzelnen Pakete. Dies erlaubt die eindeutige Zuordnung eines Reply-Pakets zu dem Request-Paket und die Ermittlung der Übertragungszeit.

Die meisten Implementierungen des `ping`-Kommandos erlauben es, die Anzahl der zu übertragenden Bytes und den Inhalt zu definieren. So kann unter Linux mit der Option `-p` ein Muster (Pattern) definiert und mit der Option `-s` die Größe angegeben werden. Werden diese Informationen nicht modifiziert, so erlauben häufig die Größe und der Inhalt des Pakets einen Rückschluss auf das sendende Betriebssystem.

STOP

Eine weitere Option, die vor allem in UNIX-Implementierungen des Befehls existiert, ist `-b`. Diese erlaubt ein Broadcast-Ping. Hierbei werden die Echo-Request-Pakete an eine Broadcast-Adresse gesendet. Üblicherweise antworten sämtliche UNIX- und Linux-Rechner auf eine derartige Anfrage. In der Vergangenheit konnten hiermit Denial-of-Service-Angriffe erzeugt werden. Der Angreifer spoofte ein Echo-Request-Paket und sendete es an eine Broadcast-Adresse. Sämtliche Rechner antworteten und schickten ihre Antwort an den gespooften Rechner. Wurden hierzu Netzwerke mit mehreren Hundert Rechnern verwendet, konnte der gespoofte Rechner häufig überflutet werden. Heute existieren kaum noch Netzwerke, die diese Pakete, die an die Broadcast-Adresse gerichtet sind, hineinlassen. Dieser Angriff ist unter dem Namen Smurf berühmt geworden. Die Netzwerke bezeichnet man als Smurf-Amplifier-Netzwerk (Verstärker). Informationen hierzu finden Sie zum Beispiel unter `http://www.powertech.no/smurf/`.

A.16.7 *address-mask-request* und *address-mask-reply*

Diese beiden Nachrichten können verwendet werden, um die Subnetzmaske eines Rechners zu ermitteln. Diese Nachrichten verwenden ähnlich dem Echo eine Identifikationsnummer und eine Sequenznummer, um die Nachrichten zuordnen zu können.

INFO

Diese Anfragen werden teilweise von Angreifern verwendet, um herauszufinden, ob ein bestimmter Rechner erreichbar ist und welche Adressmaske er verwendet. Leider existiert kein klassisches Kommandozeilenwerkzeug für die Erzeugung der Anfrage. Ein Werkzeug, das jedoch genutzt werden kann, ist `icmpquery`. Dieses Werkzeug ist unter `http://www.angio.net/security/` erhältlich. Hiermit können Rechner erreicht werden, bei denen ein Ping durch eine Firewall blockiert wird. Linux reagiert auf einen Address Mask Request nicht.

A.16.8 *timestamp-request* und *timestamp-reply*

Diese Meldungen sind in der Lage, die Latenz des Netzwerks zu messen. Hierzu ist es jedoch erforderlich, dass sowohl der Absender als auch der Empfänger synchrone Uhrzeiten verwen-

den. Ähnlich den Echo-Meldungen und den Address-Mask-Meldungen verwenden diese Meldungen auch eine Identifikationsnummer und eine Sequenznummer, um die Pakete zuordnen zu können. Ein Werkzeug, das in der Lage ist, diese Meldungen zu erzeugen, ist `icmpquery`. Es wurde bereits bei den Address-Mask-Meldungen erwähnt. Der Linux-Kernel reagiert ab der Version 2.4 auf eine derartige Anfrage nicht mehr.

A.16.9 *router-solicitation* und *router-advertisement*

Wenn ein Netzwerkgerät, das das Router-Discovery-Protokoll unterstützt, eingeschaltet wird, sendet es eine Router-Solicitation-Meldung. Alle weiteren Router in demselben Netzwerk antworten mit einer Router-Advertisement-Nachricht. Damit alle Router die Solicitation-Nachricht erhalten, wird diese entweder an die Broadcast-Adresse 255.255.255.255 oder die All-Routers-Multicast-Adresse 224.0.0.2 gesendet. Alle Router antworten auf diese Anfragen mit einem Unicast-Paket. Zusätzlich versenden die Router möglicherweise regelmäßig ohne Aufforderung Router-Advertisement-Meldungen an die Adresse 224.0.0.1.

INFO

Unter IPv4 wird diese Funktion selten bis gar nicht verwendet. Bei IPv6 basiert jedoch die Autokonfiguration der IP-Adressen auf diesem Mechanismus.

A.17 ARP

Wenn zwei IP-fähige Netzwerkgeräte sich in einem lokalen Netz unterhalten möchten, so müssen sie zunächst ihre Hardware-Adressen ermitteln. Die tatsächliche Kommunikation erfolgt nicht auf Basis der IP-Adressen, sondern anhand dieser Hardware-Adressen. Für das Medium Ethernet wurde das Address Resolution Protocol (ARP) entwickelt. Dieses Protokoll wurde inzwischen auf die meisten anderen Netzwerkmedien portiert. Es erlaubt einem Netzwerkgerät, eine Anfrage (ARP-Request) zu senden, die von der entsprechenden Gegenstelle mit einer Antwort (ARP-Reply) beantwortet wird.

ARP-Pakete werden auf der Data-Link-Schicht versendet. Das ist dieselbe Schicht, die von IP-Paketen genutzt wird. ARP-Pakete sind also unabhängig von IP-Paketen.

ARP-Anfragen werden üblicherweise an alle Rechner eines Netzes versandt. Die Zieladresse des Pakets ist daher ff:ff:ff:ff:ff:ff. Dies ist die Ethernet-Broadcast-Adresse. Alle Rechner des Netzes verarbeiten das Paket. Es antwortet aber nur derjenige Rechner, der die richtige IP-Adresse besitzt.

Die Ergebnisse dieser Anfragen werden von den Rechnern in einem ARP-Cache zwischengespeichert. Das Verhalten des ARP-Caches wird unter Linux über das `sysctl`-Interface in `/proc/sys/net/ipv4/neigh/*/` gesteuert. Die Manpage des ARP-Kernel-Moduls `arp (7)` gibt nähere Auskunft über die Werte. Der Inhalt des ARP-Caches kann mit dem Befehl `arp` angezeigt werden:

```
# tcpdump -np arp
tcpdump: listening on eth1
```

```
15:22:23.374171 0:10:a4:c3:26:cb Broadcast arp 42: arp who-has ↵
        192.168.0.101 tell 192.168.0.202
15:22:23.374625 0:e0:7d:7d:70:69 0:10:a4:c3:26:cb arp 60: arp reply ↵
        192.168.0.101 is-at 0:e0:7d:7d:70:69
Ctrl-C
# arp -an
? (192.168.0.1) auf 00:50:BF:11:23:DF [ether] auf eth1
? (192.168.0.101) auf 00:E0:7D:7D:70:69 [ether] auf eth1
```

Der Linux-Kernel kann maximal 1024 Einträge in seinem Cache verwalten.

INFO

Ab IPv6 wird die Funktion des ARP-Protokolls von ICMP übernommen. Hier dient das Neighbor-Discovery-Protokoll mit seinen entsprechenden ICMP-Nachrichten zu diesem Zweck.

Literaturverzeichnis

[1]. Klein, Tobias: Buffer Overflows und Format-String-Schwachstellen. Heidelberg: Dpunkt Verlag 2003.

[2]. Erickson, Jon: Hacking: Die Kunst des Exploits. Heidelberg: Dpunkt Verlag, Deutsche Ausgabe der 2. Auflage, 2008.

[3]. Spenneberg, Ralf: Intrusion Detection und Prevention mit Snort 2 und Co. München u.a.: Addison-Wesley 2005.

[4]. Spenneberg, Ralf: VPN mit Linux. München u.a.: Addison-Wesley 2010.

[5]. Spenneberg, Ralf: SELinux und AppArmor. München u.a.: Addison-Wesley 2008.

[6]. Schneier, Bruce: Angewandte Kryptografie – Der Klassiker. München u.a.: Pearson Studium 2005

[7]. Stoll, Clifford: Kuckucksei. 5. Aufl. Frankfurt am Main: Fischer Taschenbuchverlag 1998.

[8]. Bellovin, William, Steven Cheswick: Firewalls und Sicherheit im Internet. 2., überarbeitete Aufl. Bonn u.a.: Addison-Wesley 1995.

[9]. Friedl, Jeffrey E.F.: Reguläre Ausdrücke. 1. Aufl. Köln: O'Reilly 1997.

[10]. Kahn, David: The Codebreakers. 2., überarbeitete Aufl. New York: Simon & Schuster Inc. 1997.

[11]. Barrett, Daniel J., Richard E. Silverman: SSH: Secure Shell – Ein umfassendes Handbuch. 1. Aufl. Köln: O'Reilly 2001.

[12]. Stevens, W. Richard: TCP/IP Illustrated. Bd. 1., 1. Aufl. Reading u.a.: Addison Wesley 1994.

[13]. Hall, Eric A.: Internet Core Protocols: The Definitive Guide. 1. Aufl. Sebastopol u.a.: O'Reilly 2000.

[14]. Wyk, Kenneth R. van, Richard Forno: Incident Response. 1. Aufl. Sebastopol u.a.: O'Reilly 2001.

[15]. Mandia, Kevin, Chris Prosise: Incident Response: Investigating Computer Crime. 1. Aufl. New York u.a.: Osborne/McGraw Hill 2001.

[16]. Proctor, Paul E.: The Practical Intrusion Detection Handbook. 1. Aufl. Upper Saddle River: Prentice Hall PTR 2001.

[17]. Bace, Rebecca Gurly: Intrusion Detection. 1. Aufl. Indianapolis: Newriders 2000.

[18]. Barman Scot: Writing Information Security Policies. 1. Aufl. Indianapolis: New Riders 2002.

[19]. Northcutt, Stephen, Judy Novak: Network Intrusion Detection: An Analyst's Handbook. 2. Aufl. Indianapolis: New Riders; 2001; ISBN 0-7357-1008-2

[20]. Schultz, E. Eugene, Russell Shumway: Incident Response. 1. Aufl. Indianapolis: New Riders 2001.

[21]. Kurtz, George, Stuart McClure, Joel Scambray: Das Anti-Hacker Buch. 3. Aufl. Bonn: MITP 2001.

[22]. Northcutt, Stephen, Mark Cooper, Matt Fearnow, Karen Frederick: Intrusion Signatures and Analysis. 1. Aufl. Indianapolis: New Riders 2001.

[23]. The Honeynet Project: Know Your Enemy. 1. Aufl. Reading u.a.: Addison Wesley 2002.

[24]. Blaze, Matt, Whitfield Diffie, Ronald l. Rivest, Bruce Schneier, Tsutomu Shimomura, Eric Thomson, Michael Wiener: Minimal Key Length of Symmetric Ciphers to Proved Adequate Commercial Security. 1996. http://www.counterpane.com/keylength.html

[25]. Lenstra, Arjen K., Eric R. Verheul: Selecting Cryptographic Key Sizes. In: Journal of Cryptology. Bd. 14(4) 2001. S. 255–293.

[26]. Cavallar, Stefania, Bruce Dodson, Arjen K. Lenstra, Walter Lioen, Peter L. Montgomery, Brian Murphy, Herman te Riele, Karen Aardal, Jeff Gilchrist, GÃ©rard Guillerm, Paul Leyland, et al.: Factorisation of a 512-bit RSA modulus. In: Theory and Application of Cryptographic Techniques, ftp://ftp.gage.polytechnique.fr/pub/publications/jma/rsa-155.ps.

Stichwortverzeichnis

T