

Ralf Spenneberg

Linux-Firewalls mit iptables & Co.

Sicherheit mit Kernel 2.4 und 2.6
für Linux-Server und -Netzwerke



 ADDISON-WESLEY

An imprint of Pearson Education

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

Teil VII

Protokolle und Applikationen





32 Behandlung einzelner Protokolle

Dieses Kapitel bespricht einzelne IP-Protokolle bei ihrer Filterung durch Iptables und gibt 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, so dass Sie selbst entscheiden können, ob Sie diese nutzen möchten.

32.1 DHCP

Das Dynamic Host Configuration Protocol 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 Zurordnung 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 ein Client mit der MAC-Adresse sich beim 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.

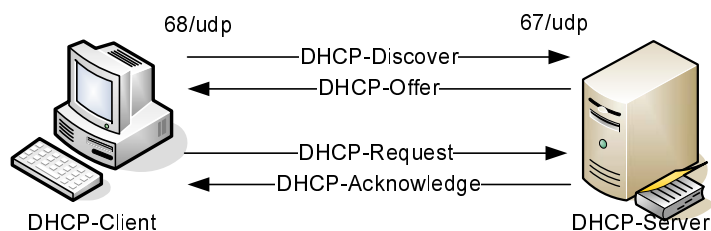


Abbildung 32.1: Das DHCP-Protokoll

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.

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 kann 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:

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 -j ACCEPT
$IPTABLES -A FORWARD -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, bei 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 -j ACCEPT
$IPTABLES -A FORWARD -s 0.0.0.0 -d 255.255.255.255 -p udp --sport bootpc --dport bootps -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.



Achtung

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.

Ein Betriebssystem, das jetzt bereits diese Funktion nutzt, ist Microsoft Windows 2003 Server. 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), wird aber im täglichen Einsatz meistens nicht verwendet.

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.



Achtung

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://nstx.dereference.de/nstx/>). Eine Beschreibung dieses Programmes finden Sie hier: <http://www.heise.de/security/artikel/print/43716/>.

Ein weiteres mächtigeres Programm wurde von Dan Kaminsky auf der Blackhat 2005 in Amsterdam vorgestellt. Sein Fragile-Router-Protocol ist in der Lage, 65 Kbit/s Daten über das DNS-Protokoll zu streamen.

Da DNS-Server Anfragen, die sie nicht beantworten können, weiterleiten, ist die Einschränkung der Nutzung auf bestimmte Nameserver nicht ausreichend.

Im Folgenden stelle ich die Regeln für drei Szenarien vor.

1. Ein LAN greift auf einen 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. Hier können Sie den Client der DNS-Verbindung zusätzlich einschränken. Dies erhöht die Sicherheit vor einem eventuellen DNS-Tunnel.

```
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.

```
# 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. Etwa 70-80% des gesamten Internetverkehrs wird von diesem Protokoll transportiert. Es dient dem 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 die 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

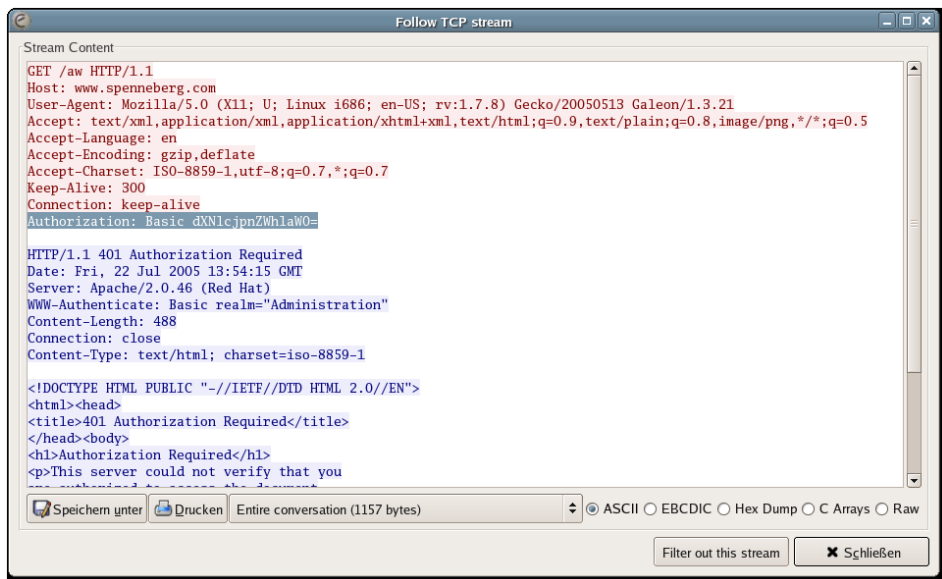


Abbildung 32.2: Ethereal kann die BASIC-HTTP-Authentifizierung anzeigen.

Cookies, die von dem Server an den Browser übertragen werden und beim 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 Ethereal können Sie sich diese Zeichenkette ansehen (Abbildung 32.2)

Die Zeichenkette lässt sich dann auf jedem Linux-System leicht dekodieren:

```
[spenneb@bibo screenshots]$ echo "dXNlcjpnZWlhYW0=" | 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: Hier erfolgt die Anmeldung analog einer Anmeldung an einem Microsoft Windows-Server. 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 eine Authentifizierung verlangen. Diese Phase wird mit SSL geschützt. Anschließend erzeugt die Webapplikation eine Zufallszahl, die in einem Cookie gespeichert wird, der anschließend für die Authentifizierung genutzt wird. Die weiteren Verbindungen können dann wieder in Klartext übertragen werden, da kein Kennwort mehr übertragen wird. Es besteht nur die Gefahr, dass der Cookie geklaut wird. Daher wird häufig dieser Cookie mit einer maximalen Lebensdauer von wenigen Minuten versehen.

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 zu den Regeln.

Zunächst werden Beispielregeln demonstriert, so dass ein 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 --port 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 --port 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
- und 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. Ab Mitte September 2005 wird mit ElsterOnline ein zusätzlicher Weg eröffnet, der nur noch einen Browser benötigt. Die Übertragung erfolgt dann über HTTPS. Ab 2006 wird dies wahrscheinlich das ElsterFormular ablösen. ElsterFT ist schließlich eine Software für Behörden, um verfahrensabhängige Formulare herunterzuladen, Send- und Abholaufträge zu erstellen und Monitor- und Protokollfunktionen zu übernehmen. Eine Übertragung von Steuererklärungen oder deren Daten ist nicht möglich.

Das ElsterFormular benötigt für die Übertragung der Daten Zugriff auf den TCP-Port 8000. Alternativ ist auch die Übertragung über einen HTTP-Proxy möglich. Dann muss dieser aber den Zugriff auf den TCP-Port 8000 ermöglichen.

ElsterFT benötigt Zugriff auf den TCP-Port 80. Ein Zugriff über einen Proxy ist möglich. Elster selbst benötigt je nach Version Zugriff auf den TCP-Port 8000 oder 80.

32.4.1 Iptables-Regeln

Im Wesentlichen benötigt Elster Zugriff auf die Server der Finanzverwaltung. Hier muss der Zugriff auf den TCP-Port 80 und 8000 möglich sein. Die aktuelle Liste der Elster-Server (Stand Oktober 2005): 62.157.211.58, 62.157.211.59, 62.157.211.60, 193.109.238.26, 193.109.238.27 und 213.182.157.55. Folgende Regeln erlauben den Zugriff:

```
ELSTER_SERVER="62.157.211.58 62.157.211.59 62.157.211.60 193.109.238.26 193.109.238.27 213.182.157.55"
```

```

for SRV in $ELSTER_SERVER
do
  $IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -d $SRV -p tcp -m multiport --dport 80,8000 ↵
  -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 elster iphash
$IPSET -A elster 62.157.211.58
$IPSET -A elster 62.157.211.59
$IPSET -A elster 62.157.211.60
$IPSET -A elster 193.109.238.26
$IPSET -A elster 193.109.238.27
$IPSET -A elster 213.182.157.55
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -m set --set elster dst -p tcp -m multiport ↵
--dport 80,8000 -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 Netzwerkkomponenten wie Router, Switches oder Drucker per `telnet` konfiguriert. Eine Aufrüstung auf SSH ist teilweise zwar möglich, kostet aber 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 bössartiger 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. 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.

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:

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. Der Administrator in einer Firma möchte möglichst jede Anwendung verhindern, 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 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://community.sidestep.pt/~filipe/Iptables-p2p/>). Leider wird das Letztere seit September 2004 nicht mehr aktiv weiterentwickelt. Die aktuellste Version befindet sich im CVS. Diese Module versuchen, den Netzwerkverkehr durch eine Inhaltsanalyse zu erkennen. Während das Iptables-p2p-Modul nur Peer2Peer-Verkehr erkennen kann (FastTrack, eDonkey, Gnutella, BitTorrent und OpenFT), erkennt L7-Filter bereits eine Vielzahl von Protokollen und wird aktiv weitergepflegt.

Der Vorteil dieser Module ist, dass sie das P2P-Protokoll unabhängig von dem verwendeten Port erkennen. So kann der P2P-Verkehr auch bei Modifikation der

Ports noch sicher erkannt werden. Um Edonkey-Verbindungen mit dem Iptables-p2p-Modul zu verhindern, müssen Sie Ihren Kernel entsprechend patchen und die folgenden Befehle in Ihrem Skript eintragen:

```
$IPTABLES -A FORWARD -m p2p --p2p-protocol edonkey -j REJECT
```

Tipp



Dieses Modul kann auch sehr gut genutzt werden, um den Verkehr zu erkennen, zu markieren und anschließend zu priorisieren. So kann die Bandbreite, die der P2P-Verkehr einnimmt, beschränkt werden.

Hinweis



Der Versuch, das Modul auf einer Fedora Core-4-Distribution zu übersetzen, schlug fehl. Dieses Modul kann wahrscheinlich nur auf älteren Distributionen, die die Iptables-Version 1.2.9 verwenden, eingesetzt werden.

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 -j
-m state --state NEW -j ACCEPT
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp --dport $KAZAA_PORT -j DNAT -j
--to-destination $KAZAA_CLIENT
$IPTABLES -A FORWARD -i $EXTDEV -o $INTDEV -d $KAZAA_CLIENT -p tcp --dport $KAZAA_PORT -j
-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 FT-wall von Chris Lowth (<http://www.lowth.com/p2pwall/ftwall/>). 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

```

Mit dem iptables-p2p-Modul könnten Sie die folgende Regel verwenden:

```

$IPTABLES -A FORWARD -m p2p --p2p-protocol fasttrack -j REJECT

```

Außerdem gibt es noch das ipp2p-Modul aus Patch-O-Matic. Dieses Modul erlaubt ebenfalls den Test des Verkehrs:

```

iptables -A FORWARD -m ipp2p --edk --kazaa --bit -j DROP
iptables -A FORWARD -p tcp -m ipp2p --ares -j DROP
iptables -A FORWARD -p udp -m ipp2p --kazaa -j DROP

```

Dieses Modul aus Patch-O-Matic ist in der Lage die folgenden Protokolle (teilweise) zu erkennen: eDonkey/eMule, KaZaA, Gnutella, Direct Connect, BitTorrent, AppleJuice, SoulSeek, WinMX und Ares/AresLite.

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 Demo-datei eines Spiels anbieten möchte, wird die Download-Rate rapide in die Höhe schnellen. Die Anwender müssen sich mit langsameren Downloads zufrieden geben, 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.

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 tcp --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 `iptables-p2p` (siehe Abschnitt 32.7) sein oder das L7-Filter-Modul. Das Letztere wird in einem eigenen Kapitel besprochen. Das Ersthre kann sehr einfach eingesetzt werden:

```
$IPTABLES -A FORWARD -m p2p --p2p-protocol bittorrent -j REJECT
```

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 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.



Achtung

Es existieren kerberosierte Versionen, die einen Single-Sign-On ermöglichen und auch anschließend sämtliche Daten bei der Übertragung schützen. 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 beim 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 (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.

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

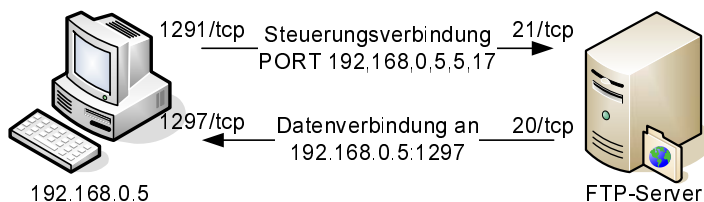


Abbildung 32.3: Der Server baut bei aktivem FTP die Datenverbindung auf.

Portnummer, auf der er den Aufbau der Datenverbindung erwartet. Hier verwendet der Server einen TCP-Port > 1023!



Achtung



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 (Abbildung 32.4).

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.

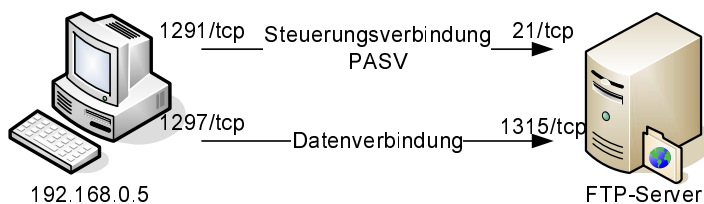


Abbildung 32.4: Der Server baut bei aktivem FTP die Datenverbindung auf.

- Bei dem aktiven FTP müssen Sie Verbindungen von dem Server zu Ihrem geschützten FTP-Client durch die Firewall erlauben. Dabei kennen Sie den Zielpport 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.
- 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 Die Stateful Inspection des FTP-Protokolls

Die Stateful Inspection des FTP-Protokolls wurde ursprünglich in kommerziellen Firewalls entwickelt. Sie wird inzwischen 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 Connection Tracking. Diese Verbindung und alle zu ihr gehörenden Pakete erhalten nun den Zustand `RELATED`, da sie mit einer anderen Verbindung (der Steuerungsverbindung) verwandt sind¹. Um nun die Stateful Inspection für das FTP-Protokoll in Ihrer Linux-Firewall anzuschalten, genügt es, das entsprechende Kernelmodul `ip_conntrack_ftp` zu laden. 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-Zielpport 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 beim Laden mit dem Befehl `modprobe` auch noch konfigurieren.

Der Befehl `modinfo` zeigt Ihnen die Möglichkeiten des Moduls `ip_conntrack_ftp`:

```
[root@bibo ~]# modinfo ip_conntrack_ftp
filename:      /lib/modules/2.6.12-1.1390_FC4/kernel/net/ipv4/netfilter/ip_conntrack_ftp.ko
license:      GPL
author:       Rusty Russell <rusty@rustcorp.com.au>
description:  ftp connection tracking helper
```

¹ Diese verwandten Verbindungen werden auch Expectation genannt, da es sich um erwartete Verbindungen handelt.

```
parmtyp:      ports:array of int
parmtyp:      loose:int
vermagic:     2.6.12-1.1390_FC4 686 REGPARAM 4KSTACKS gcc-4.0
depends:       ip_contrack
srcversion:   7A102256C83E32975182EEF
```

Fett hervorgehoben sind die Parameter, die beim Laden des Moduls mit angegeben werden können. 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 ip_contrack_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 kommunizieren 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 genettet werden. Hierzu gibt es ein weiteres Kernelmodul, das die korrekte Adressumsetzung garantiert. Auch dieses Modul, `ip_nat_ftp`, müssen Sie selbst laden, wenn Sie die Funktionalität wünschen. Es wird nicht automatisch von dem Kernel geladen.

```
$MODPROBE ip_nat_ftp ports=21,2121
```

Sie müssen bei diesem Modul genauso wie beim `ip_contrack_ftp`-Modul die zu überwachten Ports angeben. Erst ab Kernel 2.6.11 fällt diese Anforderung weg und das Modul liest die Ports aus dem `ip_contrack_ftp`-Modul aus.

32.10.3 Iptables-Regeln

Bei 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 ip_contrack_ftp # Aktiviert die Stateful Inspection
$MODPROBE ip_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 `ip_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.



Achtung

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, sind SMTP-Server und IRC-Server.

Das Laden des `ip_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 ip_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 im Klartext übertragen werden, sollte das Protokoll nicht in ungeschützten Netzwerken eingesetzt werden. Meistens baut der Client (Manager) die Verbindung zum Server (Agent) auf. Jedoch kann auch der Agent sich in wichtigen Fällen beim 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 laden:

```
$MODPROBE ip_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 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, denken Sie daran, 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 `ip_nat_snmp_basic` laden. Dieses Modul ist bei 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 Ammandad. Der Client forkt einen Ammandad-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) als auch in Kapitel 19 besprochen. Auch für Amanda gibt es zwei Kernel-Module, die die einfache Filterung des Protokolls erlauben:

```
$MODPROBE ip_conntrack_amanda
$MODPROBE ip_nat_amanda
```

Das Modul `ip_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 ip_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 ip_conntrack_amanda master_timeout=3600
$MODPROBE ip_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 PPTP-Protokoll wird 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 Antwortpakete der Verbindung auch an den richtigen Client zurückzusenden. Hierzu pflegt die Firewall eine Tabelle, in der alle genetteten Verbindungen eingetragen werden². 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.

² Bei Linux ist dies auch die Connection-Tracking-Tabelle.

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 auseinander halten und weist sie immer der zuletzt aktiven Verbindung zu.

Dieses Problem lässt sich mit der Stateful Inspection lösen.

32.13.1 Iptables-Regeln

Die Iptables-Regeln sind recht einfach. Möchten Sie mit Ihrer Firewall einen Client beim Zugriff auf einen PPTP-Server im Internet schützen, so können Sie folgende Regeln verwenden:

```
PPTP_SRV=5.0.0.1
GRE=47
```

```
$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. Leider sind die erforderlichen Kernelmodule nicht bei allen Distributionen enthalten. Dann müssen Sie Ihren Kernel bis einschließlich 2.6.13 selbst patchen und übersetzen (siehe Kapitel 18).



Achtung

Dieses Modul hat momentan noch einige Einschränkungen: Es funktioniert nicht mit den Linux-Kernen zwischen 2.6.10 und 2.6.13. Ab 2.6.14 ist es im Kernel enthalten! Außerdem ist auch eine Neukompilierung des Kommandos `iptables` erforderlich.

Sie benötigen die folgenden Module:

```
$MODPROBE ip_conntrack_proto_gre
$MODPROBE ip_conntrack_pptp
```

Dann genügen die folgende 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-Helfer-Modul, das ebenfalls nicht in allen Distributionen enthalten ist. Mit dem obigen Patch stehen diese Module aber zur Verfügung.

```
$MODPROBE ip_nat_proto_gre
$MODPROBE ip_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
GRE=47
```

```
$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 wird für den Transport von E-Mail zwischen den E-Mail-Servern im Internet genutzt. Auch E-Mail-Clients wie Evolution, Kontakt 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 Mailserver bei fehlenden Angaben ist von dem Produkt abhängig und kann nicht allgemein gültig 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. Es genügt der Standardport 25/tcp hierfür.

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 --dport 25 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

```



Achtung

Sie sollten zusätzlich auf Ihrer Firewall Identd-Anfragen ablehnen. Ansonsten kann es beim 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 --dport 25 -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
```



Achtung

Ein E-Mail-Server 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 bei Einsatz eines E-Mail-Servers in einer DMZ finden Sie im Kapitel 9.

32.15 IRC

Das Protokoll Internet Relay Chat 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, so genannten 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. Auf Grund 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. Auf Grund 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, Urdernet, 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 `ip_conntrack_irc` (siehe auch Abschnitt 32.10).

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 ip_conntrack_irc ports=6667
$MODPROBE ip_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 `ip_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 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.

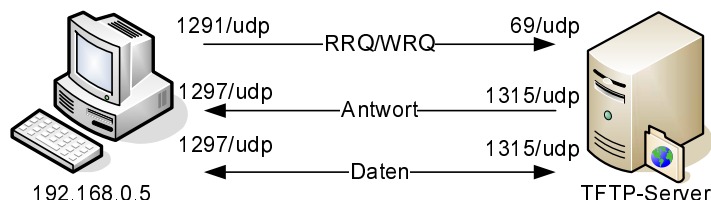


Abbildung 32.5: Das TFTP-Protokoll verwendet beliebige Ports.

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 enorme Löcher in der Firewall geöffnet werden, damit von einem beliebigen UDP-Port eine Verbindung zu einem beliebigen UDP-Port aufgebaut werden kann.

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 `ip_nat_tftp` laden. Dieses Modul benötigt in einer älteren Ausgabe auch die Angabe der Ports. Das können Sie aber selbst prüfen:

```
[root@bibo ~]# modinfo ip_nat_tftp
filename:      /lib/modules/2.6.12-1.1390_FC4/kernel/net/ipv4/netfilter/ip_nat_tftp.ko
author:       Magnus Boden <mb@ozaba.mine.nu>
description:  tftp NAT helper
license:      GPL
vermagic:     2.6.12-1.1390_FC4 686 REGPARM 4KSTACKS gcc-4.0
depends:       ip_conntrack_tftp,ip_conntrack,iptable_nat
srcversion:   F97654AA2851F38C31D9F7B
```

Dieses Modul benötigt die Angabe nicht mehr. Nun können Sie mit der folgenden Regel TFTP-Verbindungen erlauben:

```
$MODPROBE ip_conntrack_tftp
```

```
$MODPROBE ip_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, RFC 3501) erlaubt den Zugriff auf und die Verwaltung von E-Mails und Dateien auf dem Server. Im Gegensatz zum POP3-Protokoll verbleiben die E-Mails beim 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.

Achtung



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.
```

32.17.1 Iptables-Regeln

Die Iptables-Regeln sind angesichts des einfachen Protokolls sehr einfach. Zunächst die Regeln für ein Gateway:

```
IMAP_PORTS="143,993"  
  
$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp -m multiport --dport $IMAP_PORTS -j ACCEPT  
-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
```

32.18 POP3

Das Post Office Protocol 3 (RFC 1939) 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.



Achtung

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.

```

32.18.1 Iptables-Regeln

Die Iptables-Regeln für das POP3-Protokoll sind sehr einfach und entsprechen denen für das IMAP-Protokoll. 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

```

32.19 NTP

Das Network Time Protocol (NTP) dient zur Synchronisation der Systemzeit mit einer oder mehreren zentralen Zeitquellen (RFC 2030). 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 untereinander miteinander kommunizieren, verwenden sie auch als Client den Port 123. Alle anderen Clients, die zur Synchronisation eingesetzt werden, verwenden einen UDP-Port > 1023.

32.19.1 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
```

32.20 NNTP

Das Network News Transfer Protocol (NNTP, RFC 997) dient zum Transport von Nachrichten zu und von einem Newsserver. Das textbasierte Protokoll erlaubt das Auflisten der Newsgroups, das Abfordern 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.

32.20.1 Iptables-Regeln

Die Iptables-Regeln für das NNTP-Protokoll sind sehr einfach und entsprechen denen für das IMAP-Protokoll. Zunächst die Regeln für ein Gateway:

```
NNTP_PORTS="119,563"

$IPTABLES -A FORWARD -i $INTDEV -o $EXTDEV -p tcp -m multiport --dport $NNTP_PORTS -j ACCEPT
-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
```

32.21 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 erhält man 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 Streamin

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 auf <http://www.packetizer.com/voip/h323/> ausführliche Informationen über das Protokoll.

32.21.1 Iptables-Regeln

Die Komplexität des Protokolls ist auf einer Firewall durch Einsatz der Stateful Inspection handhabbar. Erfreulicherweise gibt es sowohl für den Linux-Kernel 2.4 als auch den Linux-Kernel 2.6 im Iptables-Patch-O-Matic Kernel-Module, die dies unterstützen. Ob Ihre Distribution diese Module enthält, können Sie sehr leicht mit dem folgenden Befehl prüfen:

```
[root@bibo ~]# modinfo ip_contrack_h323
modinfo: could not find module ip_contrack_h323
```

In diesem Fall müssen Sie das Modul mit Patch-O-Matic in Ihrem Kernel einführen, aktivieren und übersetzen.



Achtung

Ab Version 2.6.11 wird ein neuer Patch benötigt.

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 ip_conntrack_h323
$MODPROBE ip_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.



Tipp

Wenn Sie auch Verbindungen von außen entgegennehmen möchten, so ist ein DNAT des TCP-Ports 1720 nach innen erforderlich.

32.22 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.

Auf Grund 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-Proxys eingesetzt.

32.22.1 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 den Kernel > 2.6.11 einsetzen. Dann gibt es im Patch-O-Matic Stateful-Inspection-Module für SIP, die Sie in Ihren Kernel patchen, aktivieren (CONFIG_IP_NF_SIP, CONFIG_IP_NF_NAT_SIP) und kompilieren können. Diese erkennen die dynamischen RTP-Verbindungen und führen ein korrektes NAT durch.

```
$MODPROBE ip_conntrack_sip
$MODPROBE ip_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
```

