

Ralf Spenneberg

# Linux-Firewalls mit iptables & Co.

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



---

An imprint of Pearson Education

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

# Teil III

## Typische Firewall-Konfigurationen







# 8 Eine lokale Firewall

Im letzten Kapitel haben wir uns mit einer typischen Firewall beschäftigt. Dabei handelte es sich um einen Router, der sämtliche Pakete, die durch ihn weitergeleitet wurden, gefiltert hat. 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 interessant, wenn es sich beim 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, so dass Ihre lokalen Dienste geschützt werden.

## 8.1 Wieso eine lokale Firewall?

Wieso sollten Sie eine lokale Firewall einrichten, wenn Sie doch schon Ihr Netzwerk mit einer Firewall gesichert haben? Wie bereits in der Einführung erwähnt, 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 vor den anderen Rechnern durch eine Firewall 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 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 auf diesen Betriebssystemen 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 Pop-up-Fenster, sobald ein beliebiger Zugriff auf den Rechner erfolgt. Bei einem Portscan öffnen manche Firewalls schneller die Pop-up-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<sup>1</sup>.

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.

---

<sup>1</sup> Wir vernachlässigen hier einmal die Gefahr eines Einbruchs in den Kernel des Betriebssystems selbst.

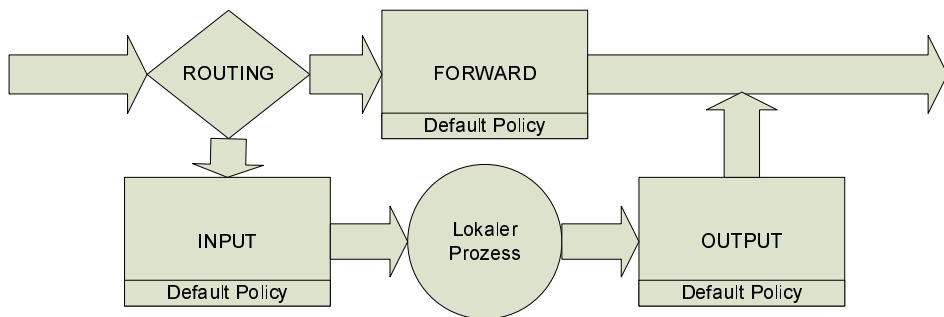


Abbildung 8.1: Für die Filterung der lokalen Pakete sind die INPUT- und OUTPUT-Ketten zuständig.

## 8.2 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 8.1).

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 8.1:** Diese lokale Firewall erlaubt alle ausgehenden Verbindungen.

```
# (1) Verwerfe 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, das nicht explizit durch eine Regel akzeptiert wird, verworfen.
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, `/proc/net/ip_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 weiteren 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 8.2: Diese lokale Firewall erlaubt nur den Zugriff auf DNS und HTTP.*

```
# (1) Verwerfe alle nicht explizit akzeptierten Pakete
$IPTABLES -F INPUT DROP
```

## 8.2 Die Ketten

```

$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 (Listing 8.1) 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 im Abschnitt 32.2. Der Punkt 3b erlaubt dann neue HTTP-Verbindungen. Da die Pakete mit dem Zustand neu in der OUTPUT-Kette akzeptiert werden, ist nicht der Aufbau neuer Verbindungen von außen möglich. Der Rest der Regeln gleicht wieder dem ersten Ansatz.



### Achtung

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<sup>2</sup> 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!

<sup>2</sup> 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.

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.

#### Hinweis



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 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 8.3: Die lokale Firewall schützt einen Root-Server.*

```
# (1) Verwerfe 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
```

## 8.2 Die Ketten

```

$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 nochmal 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!

## 8.3 Der Owner-Match

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 im Abschnitt [16.23](#) 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 Otto 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 Otto das nicht darf. Dabei können Sie dem Benutzer Ralf auch nur bestimmte Befehle erlauben. Wenn der Benutzer Ralf nur Verbindungen mit dem Kommando `ssh` aufbauen darf, können Sie folgende Zeile verwenden:

```
# Benutzer Ralf hat die UID 500 und darf ssh benutzen
$IPTABLES -A OUTPUT -m owner --uid-owner 500 --cmd-owner ssh -m state --state NEW -j ACCEPT
```

**Achtung**

Allerdings müssen Sie diese Funktion mit Vorsicht genießen. Der Anwender kann sehr einfach versuchen, diese Funktion zu unterlaufen, da er einen beliebigen Befehl in sein Heimatverzeichnis unter dem Namen `ssh` kopieren und dort aufrufen kann.

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.

**Listing 8.4:** *Nur der E-Mail-Server darf eine Verbindung nach außen aufbauen. Dem Webserver ist dies untersagt.*

```
# Der Webserver hat die UID 48 (apache)
# Der E-Mail-Server hat die UID 89 (postfix)

# Der E-Mail-Server 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 im Abschnitt [16.22](#).

Zum Abschluss möchte ich Ihnen noch ein komplettes Beispielskript für einen Root-Server vorschlagen. Sie können es entsprechend Ihren Wünschen natürlich anpassen:

**Listing 8.5:** *Dieses Firewall-Skript schützt einen typischen Root-Server.*

```
#
#
# Firewall-Skript für einen Root-Server
# (c) Ralf Spenneberg 15. Juni 2005
#
```

```
# Dieser Root-Server bietet die folgende 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)
```

## 8.4 Kombination mit Gateway-Regeln

```

$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"

```

### Tipp



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.

## 8.4 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 beim 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 5.14), 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 8.6: 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
```



### Exkurs: 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:

```
[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.

### Tipp



Anstelle einer For-Schleife können Sie auch IP-Sets 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 im Kapitel [25](#).

Möchten Sie entsprechend den SSH-Zugriff von außen erlauben, so ersetzen Sie in der Regel `-i $INTDEV` durch `-i $EXTDEV`.

Häufig wünschen 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, in dem 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.

