

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



9 Aufbau einer DMZ

Eine demilitarisierte Zone (DMZ) ist ein eigenes Netzwerk, in das bestimmte Rechner ausgelagert werden, die einen unbeschränkteren Zugriff auf das Internet benötigen oder vom Internet erreicht werden sollen (z.B. Webserver). Dieser Aufbau einer demilitarisierten Zone bietet viele Vorteile. Wenn ein Angreifer erfolgreich in einen dieser Rechner einbrechen sollte, so hat er noch keinen Zugriff auf das immer noch geschützte interne LAN. Befände sich der Webserver nicht in der DMZ, sondern direkt im internen LAN, hätte der Angreifer Zugriff auf alle Systeme!

Dieses Kapitel beschreibt den Aufbau einer DMZ, zeigt unterschiedliche Architekturen und beispielhaft den Aufbau der Firewall-Skripten, um diese zu implementieren.

9.1 DMZ-Architekturen

Bereits im Kapitel 3, »Firewall-Architekturen« haben wir verschiedene Möglichkeiten angesprochen, eine DMZ zu implementieren. Dabei wurden auch Vor- und Nachteile der verschiedenen Varianten erörtert. Im Folgenden soll die Implementierung einer DMZ als drittes Bein eines Paketfilters (Abbildung 3.2) und als Netz zwischen zwei Paketfiltern (Abbildung 3.3) exemplarisch durchgespielt werden. Während die Implementierung als drittes Bein einen relativ geringen Hardwareaufwand bedeutet, benötigt die Variante mit zwei Paketfiltern mindestens ein System als Paketfilter mehr. Dies bedeutet Geld-, Platz- und Kühlaufwand.

In der DMZ der Beispielfirma *Nohup.info* soll jeweils ein Web-, ein E-Mail, ein cachender DNS- und ein Proxy-Server betrieben werden. Diese Systeme erhalten private IP-Adressen aus dem Bereich 192.168.255.0/24. Dabei soll aus dem Internet ein Zugriff auf den Web- und den E-Mail-Server möglich sein. Aus dem internen Netz ist ein Zugriff auf den Proxy- und den E-Mail-Server notwendig. Der Zugriff auf den Webserver wird über den Proxy-Server realisiert. Ein direkter Zugriff auf das Internet ist nicht vorgesehen. Der Paketfilter erhält als externe IP-Adresse eine feste statische IP-Adresse 3.0.0.1. Alle Dienste werden im Internet unter dieser IP-Adresse angebunden. Das bedeutet, dass ein DNS-Server bei der Frage nach der Adresse `www.nohup.info` die IP-Adresse 3.0.0.1 zurückliefert.

9.2 Dreibeiniger Paketfilter mit DMZ

Die Variante des dreibeinigen Paketfilters zur Implementierung einer demilitarisierten Zone ist die einfachste Variante einer DMZ. Der Hardwareaufwand ist gering, denn Sie benötigen hierfür nur eine zusätzliche Netzwerkkarte in Ihrem Paketfilter. Auch der weitere technische Aufwand in Bezug auf Platz und Kühlung ist eher zu vernachlässigen, da Sie ja sowieso bereits ein System als Paketfilter einsetzen.

Falls Sie bereits ein Firewall-Skript für einen Paketfilter besitzen und dieses um die Funktionen für eine DMZ erweitern wollen, ist auch dies recht einfach machbar. In diesem Kapitel werden wir jedoch ein neues Skript entwickeln. Sie können die hier vorgestellten Vorschläge aber auch in jedes andere Skript übernehmen.

Zunächst sollten Sie die Netzwerkkarten zuordnen und in Ihrem Skript Variablen definieren, denen Sie die entsprechenden Werte zuweisen. Zusätzlich zu der Variable `INTDEV` und `EXTDEV` werden wir noch eine Variable `DMZDEV` verwenden. Dieser Variablen weisen Sie zunächst den Namen der Netzwerkkarte zu, über die die DMZ angebunden wird.

Listing 9.1: Eine dreibeinige Firewall mit DMZ-Anschluss

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

INTDEV="eth1"
# Achtung: Bei Einsatz von User-Mode-Linux lautet diese Variable
# INTDEV="tap0"

EXTDEV="eth0"

DMZDEV="eth2"

IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
```

Nun sollten Sie für die verschiedenen Dienste und Systeme Variablen definieren. Das erleichtert später die Wartung. Vielleicht installieren Sie zunächst nur ein System in der DMZ, das sowohl einen Web-, einen E-Mail- als auch einen Proxy-Server beherbergt. Später möchten Sie vielleicht den E-Mail-Server auf einem eigenen System betreiben. Wenn Sie zu Beginn darauf geachtet haben, für alle Systeme Variablen zu verwenden, müssen Sie diese anschließend nur entsprechend anpassen und Ihr Skript neu starten.

```
MAIL=192.168.255.2
WEB=192.168.255.3
DNS=192.168.255.4
PROXY=192.168.255.5
EXTERN=3.0.0.1
```

Tipp



Auch wenn in unserem Skript für den DNS-Server eine eigene IP-Adresse vorgesehen wurde, ist es natürlich nicht erforderlich, diesen auf eigener dedizierter Hardware zu installieren. Wenn Sie den DNS-Server auf derselben Hardware wie den Proxy-Server installieren wollen, verwenden Sie einfach für beide Variablen dieselbe IP-Adresse!

Nun sollten Sie, bevor Sie irgendwelche Regeln hinzufügen, einen sauberen Grundzustand erzeugen.

```
# Setze Default-Regeln
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Lösche die Filter-Tabelle
$IPTABLES -F

# Lösche die NAT-Tabelle
$IPTABLES -t nat -F
```

Beginnen wir nun mit den NAT-Regeln. Insgesamt müssen wir die folgenden NAT-Regeln konfigurieren:

1. Der Proxy muss auf das Internet zugreifen können.
2. Der cachende DNS-Server muss auf das Internet zugreifen können.
3. Der -Server muss auf das Internet zugreifen können.
4. Das Internet muss auf den Webserver zugreifen können.
5. Das Internet muss auf den E-Mail-Server zugreifen können.

Diese Anforderungen können mit den folgenden vier Regeln umgesetzt werden:

```
# (1) Proxy-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $PROXY -j SNAT --to-source $EXTERN
```

```
# (2) DNS-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $DNS -j SNAT --to-source $EXTERN

# (3) Mailserver-Zugriff nach außen
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $MAIL -j SNAT --to-source $EXTERN

# (4) Webserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN -m multiport --dport 80,443 \
-j DNAT --to-destination $WEB

# (5) Mailserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN --dport 25 -j DNAT \
--to-destination $MAIL
```

Die Regeln Eins bis Drei führen eine Network Address Translation (NAT) der Absenderadresse durch. Dabei wird die originale Absenderadresse des Proxy oder des Mailservers bei einem Verbindungsaufbau nach außen durch die Adresse der Firewall ersetzt. Die Regeln vier und fünf sorgen dafür, dass bei Verbindungen von außen, die auf den Ports 25, 80 und 443 aufgebaut werden, Pakete entsprechend auf den E-Mail- und Webserver in die DMZ weitergeleitet werden. Dies erfolgt durch einen Austausch der Ziel-IP-Adresse in der NAT-PREROUTING-Kette. Die Portnummer wird dabei nicht modifiziert. Da aber die Weiterleitung (Forwarding) in Abhängigkeit der Portnummer erfolgen soll, muss zusätzlich auch das Protokoll TCP angegeben werden.

Tipp



Wenn Sie aus bestimmten Gründen den Webserver in der DMZ auf einem anderen Port betreiben möchten, so wäre es auch möglich, die Portnummer zu modifizieren:

```
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN \
--dport 80 -j DNAT --to-destination $WEB:8080
```

Die NAT-Regeln sind jedoch lediglich für die Adressumsetzung beim Zugriff aus dem Internet und in das Internet verantwortlich. Weitere Firewall-Regeln sind für die Funktion der Firewall erforderlich. Diese Regeln müssen nun auch den gewünschten Verkehr erlauben.

Wir benötigen zunächst eine Regel, die sämtliche aufgebauten Verbindungen und deren Fehlermeldungen akzeptiert:

```
# Akzeptiere alle aufgebauten Verbindungen
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Tipp



Denken Sie daran: Die Zustandsüberwachung der Iptables-Firewall macht es möglich, dass Sie immer nur die Verbindungsaufbauten filtern und anschließend alle ermaßen aufgebauten Verbindungen über eine einzige Regel akzeptieren. Nur Verbindungen, die Sie vorher explizit erlaubt haben (im Status NEW), werden von dieser Regel zugelassen. Der Aufbau der Regeln wird so wesentlich einfacher und übersichtlicher.

Nun benötigen wir eine Regel, um von innen auf den Proxy und den Mailserver zuzugreifen. Der Port eines Proxys ist nicht so festgelegt wie zum Beispiel der Port beim Zugriff auf einen Web- oder E-Mail-Server. Hier werden vollkommen unterschiedliche Portnummern verwendet. Der Squid-Proxy-Server verwendet zum Beispiel die Portnummer 3128/tcp als Standardport. Es ist sinnvoll, auch hierfür eine Variable zu Beginn des Skripts zu definieren: `PROXYPORT=3128`. Dann können Sie diese Variable in Ihren Regeln nutzen und müssen bei einer Änderung nur zu Beginn des Skripts Änderungen vornehmen.

```
# Erlaube den Zugriff auf den Proxy
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $PROXY -p tcp --dport $PROXYPORT -m state \
--state NEW -j ACCEPT
```

```
# Erlaube den Zugriff auf den E-Mail-Server
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m state \
--state NEW -j ACCEPT
```

Tipp



Wenn der E-Mail-Server in der DMZ auch Dienste zum Abholen der E-Mail bereitstellt, müssen Sie die letzte Regel so erweitern, dass auch diese Funktion unterstützt wird. Das *Post Office Protocol* (POP-3) verwendet den Port 110, und das *Internet Message Access Protocol* (IMAP) verwendet den Port 143. Sinnvoller ist es aber, diese Dienste auf einem eigenen E-Mail-Server in dem internen Netz anzubieten und den E-Mail-Server in der DMZ nur als Relay zu nutzen. Dann können Sie natürlich auch den Zugriff auf den E-Mail-Server in der DMZ auf den internen E-Mail-Server beschränken, denn alle internen Anwender benutzen für ihren E-Mail-Zugriff nur den internen E-Mail-Server.

```
MAILINTERN=192.168.0.100
# Erlaube den Zugriff auf den E-Mail-Server
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -s $MAILINTERN -d $MAIL -p tcp \
--dport 25 -m state --state NEW -j ACCEPT
```

Sobald eine E-Mail in das Internet gesendet werden muss, sendet der interne E-Mail-Server diese E-Mail an den E-Mail-Server in der DMZ, der sie in das Internet weiterleitet. Eingehende E-Mails nehmen den entgegengesetzten Weg. Der Aufbau eines derartigen E-Mail-Relays mit Postfix ist in dem Anhang Postfix-Mail-Relay (siehe Anhang A) erläutert.

Für diesen Rückweg ist dann natürlich auch noch eine weitere Regel erforderlich, die den Transport aus der DMZ in das interne Netz erlaubt:

```
# Erlaube den Transport von E-Mail aus der DMZ in das interne Netz
$IPTABLES -A FORWARD -i $DMZDEV -o $INTDEV -d $MAILINTERN -s $MAIL -j ACCEPT
-p tcp --dport 25 -m state --state NEW -j ACCEPT
```

Des Weiteren benötigen natürlich der E-Mail-Server, der DNS-Server und der Proxy einen Zugriff auf das Internet. Wenn kein eigener cachender DNS-Server in der DMZ betrieben werden soll, benötigen diese Systeme natürlich auch einen Zugriff auf einen DNS-Server in dem Internet.

```
# Erlaube dem DNS-Server einen Zugriff auf Nameserver im Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p tcp --dport 53 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p udp --dport 53 -m state --state NEW -j ACCEPT
```

```
# Erlaube dem Proxy Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $PROXY -p tcp -m multiport --dport 21,80,443 -m state --state NEW -j ACCEPT
```

```
# Erlaube dem E-Mail-Server Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT
```

Wenn die Benutzer über den Proxy sowohl auf FTP-, HTTP- und HTTPS-Server zugreifen sollen, muss der Proxy auch auf entsprechende Ports zugreifen dürfen.



Achtung

Das FTP-Protokoll ist ein sehr kompliziertes Protokoll. Daher wird es in einem eigenen Abschnitt genauer betrachtet (siehe Abschnitt 32.10). Dieses Protokoll verwendet eine Steuerungsverbindung, die üblicherweise auf dem TCP-Port 21 terminiert wird, und zusätzlich für jede zu übertragende Datei eine eigene Datenverbindung, deren Ports dynamisch ausgehandelt werden. Da es sehr schwer ist, statische Regeln für derartig dynamisch ausgehandelte und damit

unbekannte Ports zu definieren, gibt es die Stateful-Inspection. Diese kann den Inhalt der Steuerungsverbindung mitlesen und die dynamisch ausgehandelten Ports erkennen. So können dann automatisch die ausgehandelten Datenverbindungen erlaubt werden. Sie müssen in Ihrem Skript dann nur noch die Steuerungsverbindung erlauben.

Damit das funktioniert, müssen Sie noch das Kernelmodul `ip_conntrack_ftp` laden:

```
MODPROBE=/sbin/modprobe
$MODPROBE ip_conntrack_ftp
```

Wenn Sie auf Ihrer Firewall auch NAT verwenden, ist zusätzlich auch noch das weitere Kernel-Modul `ip_nat_ftp` erforderlich:

```
$MODPROBE ip_nat_ftp
```

Diese Zeilen fügen Sie am besten zu Beginn Ihres Skripts ein. Alle beobachteten Datenverbindungen werden dann auch mit dem Zustand `RELATED` versehen. Da Sie derartige Pakete in der `FORWARD`-Kette bereits akzeptieren, werden diese Verbindungen erlaubt. Weitere Informationen über das FTP-Protokoll und die Kernelmodule erhalten Sie in Abschnitt [32.10](#).

Nun fehlt noch der Zugriff aus dem Internet auf den Webserver und den E-Mail-Server. Diese Zugriffe sind jetzt sehr einfach zu konfigurieren und erfolgen analog den bereits definierten Regeln. Der Zugriff erfolgt von außen auf die Systeme in der DMZ. Ein NAT wurde bereits konfiguriert. Es bleiben die folgenden Firewall-Regeln:

```
# Erlaube dem Internet den Zugriff auf den E-Mail-Server
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m state \
--state NEW -j ACCEPT
```

```
# Erlaube dem Internet den Zugriff auf den Webserver
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $WEB -p tcp -m multiport --dport 80,443 \
-m state --state NEW -j ACCEPT
```

Grundsätzlich sollte nun die komplette Funktionalität des Skripts bereits gewährleistet sein. Sinnvollerweise werden jedoch noch ein paar Protokollregeln und Ausnahmen hinzugefügt.

Als Erstes sollten Sie sicherstellen, dass Anfragen auf dem Port 113/`tcdd` (`identd`) abgelehnt und nicht verworfen werden. Ansonsten kann es zu Verzögerungen beim Aufbau von FTP- und SMTP-Verbindungen kommen.

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

Damit Ihre internen Benutzer auch nicht auf einen Timeout warten müssen, wenn die Benutzer ein nicht erlaubtes Protokoll verwenden möchten, macht es Sinn, Anfragen von innen, die abgelehnt werden sollen, nicht zu verwerfen, sondern zu protokollieren und abzulehnen. So behalten Sie immer den Überblick darüber, was Ihre Benutzer gerade treiben. Achten Sie jedoch auf die Datenschutzrichtlinien in Ihrem Unternehmen und die gesetzlichen Rahmenbedingungen.

```
$IPTABLES -A FORWARD -i $INTDEV -j LOG --log-prefix "Unerlaubt von innen: "  
$IPTABLES -A FORWARD -i $INTDEV -j REJECT  
$IPTABLES -A INPUT -i $INTDEV -j LOG --log-prefix "Unerlaubt von innen: "  
$IPTABLES -A INPUT -i $INTDEV -j REJECT
```

Da Sie diese Regeln an das Ende der Kette hängen, werden sie nur die Pakete betreffen, die nicht im Vorfeld der Kette von Ihren Regeln akzeptiert wurden.

Möglicherweise möchten Sie eine entsprechende Regel auch für Verbindungen von außen aufsetzen. Je nach Ihrer Internetanbindung möchte ich jedoch von einer derartigen grundsätzlichen Regel abraten, da Sie wahrscheinlich nicht die Zeit haben werden, Protokolle mit mehreren zehntausend Einträgen pro Tag zu analysieren. Sehr aufschlussreich ist aber noch eine derartige Regel für Verbindungen aus der DMZ, die Sie nicht explizit in Ihren Regeln erlaubt haben. Damit sind Sie in der Lage, sehr früh Probleme in Ihrem Regelwerk oder auf Ihren Systemen in der DMZ zu erkennen.

```
$IPTABLES -A FORWARD -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der DMZ: "  
$IPTABLES -A FORWARD -i $DMZDEV -j REJECT  
$IPTABLES -A INPUT -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der DMZ: "  
$IPTABLES -A INPUT -i $DMZDEV -j REJECT
```

Diese Regeln können ein Intrusion-Detection-System sehr gut komplettieren und bei der Analyse eines Angriffs oder Einbruchs sinnvolle zusätzliche Informationen bieten.

Damit sollte das Skript schließlich fertig sein. Sicherlich haben Sie noch Ideen und Wünsche, wie Sie das Skript erweitern können. Vielleicht haben Sie auch eine vierbeinige Firewall mit zwei DMZ. Dieses Skript sollten Ihnen dann aber eine gute Ausgangsposition für eigene Anpassungen geben. Im Folgenden ist das komplette Skript noch einmal zur Referenz abgedruckt.

Listing 9.2: Eine dreibeinige Firewall mit DMZ

```
#!/bin/sh  
#  
# Autor : Ralf Spenneberg  
# Version: 0.1  
# Datum : 17. Dez 2004  
#  
# Dieses Skript lädt die Regeln für die Firewall  
INTDEV="eth1"
```

9.2 Dreibeiniger Paketfilter mit DMZ

```
# Achtung: Bei Einsatz von User-Mode-Linux lautet diese Variable
# INTDEV="tap0"

EXTDEV="eth0"

DMZDEV="eth2"

IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
MODPROBE=/sbin/modprobe

MAIL=192.168.255.2
WEB=192.168.255.3
DNS=192.168.255.4
PROXY=192.168.255.5
PROXYPORT=3128
EXTERN=3.0.0.1

# Lade die Kernelmodule für die Stateful-Inspection des FTP-Protokolls
$MODPROBE ip_conntrack_ftp
$MODPROBE ip_nat_ftp

# Setze Default-Regeln
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Lösche die Filter-Tabelle
$IPTABLES -F

# Lösche die NAT-Tabelle
$IPTABLES -t nat -F

### NAT-Regeln ###
# (1) Proxy-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $PROXY -j SNAT --to-source $EXTERN
# (2) DNS-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $DNS -j SNAT --to-source $EXTERN

# (3) Mailserver-Zugriff nach außen
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $MAIL -j SNAT --to-source $EXTERN

# (4) Webserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN -m multiport --dport 80,443
-j DNAT --to-destination $WEB
```

```
# (5) Mailserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN --dport 25 -j DNAT
--to-destination $MAIL

## Firewall-Regeln
# Akzeptiere alle aufgebauten Verbindungen
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# Erlaube den Zugriff auf den Proxy
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $PROXY -p tcp --dport $PROXYPORT
-m state --state NEW -j ACCEPT

# Erlaube den Zugriff auf den E-Mail-Server
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m state
--state NEW -j ACCEPT

### Bei Verwendung eines internen E-Mail-Servers:
# MAILINTERN=192.168.0.100
# Erlaube den Zugriff auf den E-Mail-Server
# $IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -s $MAILINTERN -d $MAIL -p tcp --dport 25
-m state --state NEW -j ACCEPT
# Erlaube den Transport von E-Mail aus der DMZ in das interne Netz
# $IPTABLES -A FORWARD -i $DMZDEV -o $INTDEV -d $MAILINTERN -s $MAIL -p tcp --dport 25
-m state --state NEW -j ACCEPT
###

# Erlaube dem DNS-Server einen Zugriff auf Nameserver im Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p tcp --dport 53 -m state --state NEW
-j ACCEPT
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p udp --dport 53 -m state --state NEW
-j ACCEPT

# Erlaube dem Proxy den Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $PROXY -p tcp -m multiport --dport 21,80,443
-m state --state NEW -j ACCEPT

# Erlaube dem E-Mail-Server den Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $MAIL -p tcp --dport 25 -m state --state NEW
-j ACCEPT

# Erlaube dem Internet den Zugriff auf den E-Mail-Server
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m state --state NEW
-j ACCEPT
```

9.3 Optimierung mit benutzerdefinierten Ketten

```
# Erlaube dem Internet den Zugriff auf den Webserver
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $WEB -p tcp -m multiport --dport 80,443
-m state --state NEW -j ACCEPT

# Lehne Identd-Anfragen ab
$IPTABLES -A INPUT -p tcp --dport 113 -j REJECT

# Protokolliere alle anderen Anfragen von innen
$IPTABLES -A FORWARD -i $INTDEV -j LOG --log-prefix "Unerlaubt von innen: "
$IPTABLES -A FORWARD -i $INTDEV -j REJECT
$IPTABLES -A INPUT -i $INTDEV -j LOG --log-prefix "Unerlaubt von innen: "
$IPTABLES -A INPUT -i $INTDEV -j REJECT

$IPTABLES -A FORWARD -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der DMZ: "
$IPTABLES -A FORWARD -i $DMZDEV -j REJECT
$IPTABLES -A INPUT -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der DMZ: "
$IPTABLES -A INPUT -i $DMZDEV -j REJECT
```

In seiner jetzigen Form ist das Skript bereits durchaus lang und wahrscheinlich im ersten Moment auch für den einen oder anderen Administrator durchaus unübersichtlich. Im nächsten Abschnitt werden Sie sehen, wie diese Skripten lesbarer und wartbarer gestaltet werden können. Wir werden das Skript in Bezug auf seine Lesbarkeit und die Geschwindigkeit der Verarbeitung optimieren.

9.3 Optimierung mit benutzerdefinierten Ketten

Im letzten Abschnitt haben wir bereits ein recht umfangreiches Skript für eine dreibeinige Firewall mit DMZ entwickelt. Dieses Skript war bereits fast 100 Zeilen lang. Da das Skript für fast jede veränderliche Information (wie IP-Adressen) Variablen nutzt, ist eine weitere Wartung recht einfach. Dennoch ist es durchaus schwierig, einen Überblick zu behalten und den genauen Paketfluss nachzuvollziehen. Einfacher kann das geschehen, wenn Sie sich Ihre eigenen Ketten für jede Funktion erzeugen. Iptables bietet, wie früher schon Ipchains, die Möglichkeit, neue Ketten mit einem eigenen Namen zu definieren und diese dann mit Regeln zu füllen.

Diese benutzerdefinierte Kette weist mehrere Vorteile auf:

- Sie können durch eine sinnvolle Strukturierung der Regeln und anschließend durch eine entsprechende Auswahl der Ketten die Gesamtzahl der auszuwertenden Regeln pro Paket senken und damit die Abarbeitung der Regeln pro Paket stark beschleunigen. Stellen Sie sich vor, Ihre Firewall verfügte über 100 Regeln. Davon beträfen jeweils 50 Regeln TCP- und UDP-Pakete. Im besten Fall trifft die erste Regel bereits auf das aktuell betrachtete Paket zu, im schlechtesten Fall aber erst die Regel 100. Dabei müssten Sie bei einem TCP-Paket die UDP-Regeln gar nicht betrachten und umgekehrt. Also erzeugen Sie sich zwei neue

Ketten: `mein_udp` und `mein_tcp`. Anschließend verschieben Sie alle TCP-Regeln in die Kette `mein_tcp` und alle UDP-Regeln in die Kette `mein_udp`. Nun müssen Sie nur noch in der originalen Kette prüfen, ob es sich um ein TCP-Paket handelt. Dann springen Sie in die Kette `mein_tcp`. In dem anderen Fall springen Sie in die Kette `mein_udp`. Nun wird Ihr Paket im besten Fall von der Regel Zwei bereits akzeptiert (die erste Regel prüft ja, ob es ein TCP- oder UDP-Paket ist). Im schlechtesten Fall müssen aber nur 51 Regeln abgearbeitet werden. Die Gesamtzahl der pro Paket auszuwertenden Regeln sinkt also drastisch. Dies können Sie möglicherweise durch eine weitere Aufteilung weiter senken.

- Die benutzerdefinierten Ketten erlauben Ihnen auch eine Optimierung der Lesbarkeit. Sie können zum Beispiel drei Ketten erzeugen: `mein_extern`, `mein_dmz` und `mein_intern`. Sobald ein Paket aus dem entsprechenden Netz kommt, springen Sie in die jeweilige Kette. Dies erlaubt Ihnen eine logische Gruppierung der Regeln. Wenn Sie wissen möchten, was die DMZ verlassen darf, so müssen Sie nur noch in eine Kette schauen.
- Schließlich ist es auch in bestimmten Umgebungen mit benutzerdefinierten Ketten leichter möglich, Veränderungen an den Firewall-Regeln vorzunehmen. Vor einigen Jahren wurde ich gebeten, eine Firewall für ein größeres IT-Schulungsunternehmen aufzusetzen. Diese Firewall sollte neben dem Netz für die Personalverwaltung, den Vertrieb und dem Labornetz auch die Netze der einzelnen Schulungsräume untereinander und im Zugriff auf das Internet kontrollieren und regeln. Aufgrund der verschiedenen Themen, die in dem Unternehmen geschult wurden, waren häufig Änderungen an den Firewall-Regeln erforderlich. Die Administration sollte jedoch von Personen durchgeführt werden, die über keinerlei oder nur geringe Kenntnisse verfügten. Daher implementierte ich für jeden Schulungsraum eine eigene benutzerdefinierte Kette, in der die Regeln für den Zugriff der Rechner dieses Raumes auf das Internet definiert wurden. Zusätzlich schrieb ich ein einfaches Skript, das den Benutzer nach der Raumnummer fragte und anschließend fünf vordefinierte Firewall-Regelsätze anbot. Nach der Auswahl löschte das Skript nur die Regeln in der benutzerdefinierten Kette dieses Raumes und fügte die neuen Regeln ein. Dadurch war sichergestellt, dass alle weiteren Räume und auch die weiteren Netze zur Verwaltung der Firma ungestört weiterarbeiten konnten. Dieser Vorgang wäre ohne benutzerdefinierte Ketten ungleich komplizierter gewesen, denn ein kompletter Neustart des Firewall-Skripts kam nicht in Frage.

Bevor wir uns aber mit einer praktischen Implementierung beschäftigen, wollen wir zunächst schauen, wie die benutzerdefinierten Ketten arbeiten. Wenn Sie die Beispiele nachstellen wollen, sollten Sie zunächst sämtliche auf Ihrem System geladenen Firewall-Regeln löschen und die Default-Policies der Ketten auf `ACCEPT` setzen. Auf einem Fedora/RedHat-System kann das zum Beispiel recht komfortabel mit dem Befehl `service iptables stop` erfolgen. Ansonsten können Sie ein Skript wie in Listing 5.2 auf Seite 98 verwenden. Im Anhang finden Sie noch ein ausführliches Skript, das tatsächlich alle Ketten löscht und den Urzustand wiederherstellt (siehe Abschnitt B.1).

Hinweis



Dieses Skript zum Stopp der Firewall ist auch auf der CD in dem Ordner *Beispielskripte* enthalten.

Sie können eine neue benutzerdefinierte Kette mit dem Befehl `iptables` erzeugen. Hierzu verwenden Sie einfach:

```
[root@bibo ~]# iptables -N meinekette
[root@bibo ~]# iptables -vnL
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination

Chain INPUT (policy ACCEPT 93 packets, 58921 bytes)
 pkts bytes target    prot opt in     out     source                   destination

Chain OUTPUT (policy ACCEPT 92 packets, 7147 bytes)
 pkts bytes target    prot opt in     out     source                   destination

Chain meinekette (0 references)
 pkts bytes target    prot opt in     out     source                   destination
```

Sie sehen am unteren Ende der Ausgabe des Befehls `iptables -vnL` Ihre neue Kette. Sofort fällt auf, dass diese Kette scheinbar keine Policy hat. Anstelle der Policy wird hier die Anzahl der Referenzen angegeben. Damit Sie erkennen, was eine Referenz ist, erzeugen wir nun eine Regel, die die Kette nutzt:

```
[root@bibo ~]# iptables -A INPUT -j meinekette
[root@bibo ~]# iptables -vnL
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination

Chain INPUT (policy ACCEPT 654 packets, 243K bytes)
 pkts bytes target    prot opt in     out     source                   destination
 299 38464 meinekette all  --  *      *      0.0.0.0/0              0.0.0.0/0

Chain OUTPUT (policy ACCEPT 649 packets, 60506 bytes)
 pkts bytes target    prot opt in     out     source                   destination

Chain meinekette (1 references)
 pkts bytes target    prot opt in     out     source                   destination
```

In der Kette INPUT ist nun eine Regel vorhanden, die auf die Kette `meinekette` verweist. Dadurch wird die Kette `meinekette` in einer weiteren Kette referenziert. Dies wird in dem Referenzzähler in der Kette `meinekette` angezeigt.

Solange die Kette `meinekette` noch irgendwo referenziert wird, können Sie diese nicht entfernen. Die benutzerdefinierte Kette können Sie genauso entfernen, wie Sie diese auch erzeugen. Hierfür verwenden Sie lediglich die Option `-X`.

```
[root@bibo ~]# iptables -X meinekette
iptables: Too many links
```

Hier werden Sie darauf hingewiesen, dass noch eine Verknüpfung mit dieser Kette existiert. Diese müssen Sie zuerst löschen. Es gibt keine Suchfunktion für diese Referenz. Sie können sich nur durch den `grep`-Befehl unterstützen lassen.

Nun fügen wir der Kette `meinekette` zunächst eine Regel hinzu:

```
[root@bibo ~]# iptables -A meinekette -j DROP
```

Dann löschen wir die Referenz:

```
[root@bibo ~]# iptables -F INPUT
[root@bibo ~]# iptables -vnL
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
Chain INPUT (policy ACCEPT 1427 packets, 612K bytes)
  pkts bytes target     prot opt in     out     source            destination
Chain OUTPUT (policy ACCEPT 1455 packets, 135K bytes)
  pkts bytes target     prot opt in     out     source            destination
Chain meinekette (0 references)
  pkts bytes target     prot opt in     out     source            destination
    0    0 DROP      all  --  *      *       0.0.0.0/0        0.0.0.0/0
```

Wenn Sie nun versuchen, die Kette zu löschen, wird der Versuch wieder fehlschlagen, da die Kette noch nicht leer ist. Erst nach dem Flush der Kette kann die benutzerdefinierte Kette gelöscht werden.

```
[root@bibo ~]# iptables -X meinekette
iptables: Directory not empty
[root@bibo ~]# iptables -F meinekette
[root@bibo ~]# iptables -X meinekette
```

Damit ist schon fast alles zur Administration der benutzerdefinierten Ketten gesagt. Es bleibt der Verweis auf die Möglichkeit, dass Sie eine benutzerdefinierte Kette umbenennen können. Hierzu gibt es den folgenden Befehl:

```
[root@bibo ~]# iptables -E meinekette neuername
```

Eine Frage ist aber dennoch offen: Was passiert am Ende der benutzerdefinierten Kette, wenn es keine Standardrichtlinien (Policy) gibt? Antwort: Die benutzerdefinierte Kette arbeitet ähnlich einem Unterprogramm. Nach ihrer Abarbeitung springt sie in die aufrufende Kette zurück, die weiter abgearbeitet wird. Wenn Sie dieses Verhalten verhindern möchten, müssen Sie am Ende der benutzerdefinierten Kette eine eigene Regel vorsehen, die alle nicht von expliziten Regeln betrachteten Pakete betrifft. Das könnte zum Beispiel der folgende Iptables-Befehl erreichen:

```
[root@bibo ~]# iptables -A meinekette -j DROP
```

Diese Regel würde alle weiteren Pakete in dieser Kette verwerfen. Ein Rücksprung in die aufrufende Kette würde nicht mehr erfolgen, da für alle Pakete in dieser Kette eine Entscheidung getroffen wurde. Ob Sie die restlichen Pakete verwerfen (DROP), ablehnen (REJECT) oder akzeptieren (ACCEPT) hängt sicherlich von der Logik Ihrer Firewall ab.

Genauso kann es jedoch sein, dass Sie bei bestimmten Paketen einen sofortigen Rücksprung in die aufrufende Kette veranlassen möchten. Auch dies ist möglich. Hierfür gibt es das Ziel RETURN. Damit können Sie die benutzerdefinierte Kette sofort verlassen und in der aufrufenden Kette die weiteren Regeln betrachten.

```
[root@bibo ~]# iptables -A meinekette -j RETURN
```

9.3.1 Anwendung der benutzerdefinierten Ketten

Nun wollen wir die benutzerdefinierten Ketten nutzen, um das Skript der dreibeinigen Firewall übersichtlicher und wartbarer zu gestalten. Wenn Sie das entsprechende Kapitel nicht gelesen haben, möchte ich Ihnen nun ans Herz legen, es kurz querzulesen, so dass Sie den Sinn des folgenden Skripts und des Umbaus kennen.

Zunächst sollten wir uns überlegen, welche Ketten wir benutzerdefiniert anlegen möchten. Es gibt grundsätzlich mehrere konzeptionelle Möglichkeiten:

1. Wir erzeugen für jedes Netz eine Kette. In dieser Kette wird definiert, welche Verbindungen das Netz aufbauen darf. Dies trennt die Regeln also entsprechend der Herkunft der Verbindungen auf. Die drei Ketten heißen dann: DMZ, LAN und INTERNET.
2. Wir erzeugen für jedes IP-Protokoll eine Kette. So gibt es drei Ketten: TCP, UDP und ICMP. Dies kann in bestimmten Umgebungen von Vorteil sein. Hier ist es das sicher nicht.
3. Wir erzeugen für jeden Informationsfluss eine Kette. Das bedeutet, dass wir eine Kette für den Zugriff auf unseren Webserver erzeugen, eine Kette für den E-Mail-Verkehr und eine Kette für den Proxy. Sobald wir Änderungen an der Art und Weise der E-Mail-Zustellung vornehmen wollen, müssen wir nur in der entsprechenden Kette die Änderungen vornehmen.



Achtung

Es ist nicht möglich, eine Kette `icmp` zu erzeugen und zu benutzen, da dieses Schlüsselwort bereits belegt ist. Sie können diese Namenskonflikte sehr einfach vermeiden, indem Sie Großbuchstaben für Ihre Ketten verwenden oder jeder Kette einen Buchstaben voranstellen.

Welche der drei Konzepte Sie für Ihre zukünftigen Firewalls verwenden, möchte ich Ihnen überlassen. Ich wähle für die weitere Erläuterung den Fall Eins, bei dem drei Ketten (DMZ, LAN und INTERNET) erzeugt werden.

Das Skript 9.2 kann nun umgeschrieben werden. Dabei werden die ersten Zeilen inklusive der NAT-Regeln unverändert übernommen und anschließend die benutzerdefinierten Ketten erzeugt.

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

INTDEV="eth1"
# Achtung: Bei Einsatz von User-Mode-Linux lautet diese Variable
# INTDEV="tap0"

EXTDEV="eth0"

DMZDEV="eth2"

IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
MODPROBE=/sbin/modprobe

MAIL=192.168.255.2
WEB=192.168.255.3
DNS=192.168.255.4
PROXY=192.168.255.5
PROXYPORT=3128
EXTERN=3.0.0.1

# Lade die Kernelmodule für die Stateful-Inspection des FTP-Protokolls
$MODPROBE ip_conntrack_ftp
```

9.3 Optimierung mit benutzerdefinierten Ketten

```

$MODPROBE ip_nat_ftp

# Setze Default-Regeln
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Lösche die Filter-Tabelle
$IPTABLES -F

# Lösche die NAT-Tabelle
$IPTABLES -t nat -F

## NAT-Regeln ##
# (1) Proxy-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $PROXY -j SNAT --to-source $EXTERN
# (2) DNS-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $DNS -j SNAT --to-source $EXTERN

# (3) Mailserver-Zugriff nach außen
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $MAIL -j SNAT --to-source $EXTERN

# (4) Webserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN -m multiport --dport 80,443 \
-j DNAT --to-destination $WEB

# (5) Mailserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN --dport 25 -j DNAT \
--to-destination $MAIL

# Erzeuge benutzerdefinierte Ketten
$IPTABLES -N DMZ
$IPTABLES -N LAN
$IPTABLES -N INTERNET

## Firewall-Regeln
# Akzeptiere alle aufgebauten Verbindungen
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

```

Nun können die Regeln auf die Ketten verteilt werden und die Ketten angesprungen werden.

```

#####
#
# Zugriff aus dem LAN
#

```

```

# Erlaube den Zugriff auf den Proxy
$IPTABLES -A LAN -o $DMZDEV -d $PROXY -p tcp --dport $PROXYPORT -m state --state NEW -j ACCEPT

# Erlaube den Zugriff auf den E-Mail-Server
$IPTABLES -A LAN -o $DMZDEV -d $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT

### Bei Verwendung eines internen E-Mail-Servers:
# MAILINTERN=192.168.0.100
# Erlaube den Zugriff auf den E-Mail-Server
# $IPTABLES -A LAN -o $DMZDEV -s $MAILINTERN -d $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT

# Protokolliere alle anderen Anfragen von innen
$IPTABLES -A LAN -j LOG --log-prefix "Unerlaubt von innen: "
$IPTABLES -A LAN -j REJECT

#####
#
# Zugriff aus der DMZ
#
# Erlaube den Transport von E-Mail aus der DMZ in das interne Netz
# $IPTABLES -A DMZ -o $INTDEV -d $MAILINTERN -s $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT

###

# Erlaube dem DNS-Server einen Zugriff auf Nameserver im Internet
$IPTABLES -A DMZ -o $EXTDEV -s $DNS -p tcp --dport 53 -m state --state NEW -j ACCEPT
$IPTABLES -A DMZ -o $EXTDEV -s $DNS -p udp --dport 53 -m state --state NEW -j ACCEPT

# Erlaube dem Proxy den Zugriff auf das Internet
$IPTABLES -A DMZ -o $EXTDEV -s $PROXY -p tcp -m multiport --dport 21,80,443 -m state --state NEW -j ACCEPT

# Erlaube dem E-Mail-Server den Zugriff auf das Internet
$IPTABLES -A DMZ -o $EXTDEV -s $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT

# Protokolliere alle anderen Zugriffe aus der DMZ
$IPTABLES -A DMZ -j LOG --log-prefix "Unerlaubt aus der DMZ: "
$IPTABLES -A DMZ -j REJECT

#####
#
# Zugriff aus dem Internet
#

```

9.3 Optimierung mit benutzerdefinierten Ketten

```
# Erlaube dem Internet den Zugriff auf den E-Mail-Server
$IPTABLES -A INTERNET -o $DMZDEV -d $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT

# Erlaube dem Internet den Zugriff auf den Webserver
$IPTABLES -A INTERNET -o $DMZDEV -d $WEB -p tcp -m multiport --dport 80,443 -m state \
  --state NEW -j ACCEPT

# Rufe die Ketten auf

# FORWARD
$IPTABLES -A FORWARD -i $INTDEV -j LAN
$IPTABLES -A FORWARD -i $DMZDEV -j DMZ
$IPTABLES -A FORWARD -i $EXTDEV -j INTERNET

# INPUT
$IPTABLES -A INPUT -p tcp --dport 113 -j REJECT
$IPTABLES -A INPUT -i $INTDEV -j LAN
$IPTABLES -A INPUT -i $DMZDEV -j DMZ
```

Wenn Sie sich nun Ihre FORWARD-Kette ansehen, werden Sie feststellen, dass diese sehr übersichtlich geworden ist. Es sind nur noch 4 Regeln vorhanden:

```
[root@bibo ~]# iptables -vnL FORWARD
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source          destination
  0    0 ACCEPT      all  --  *      *       0.0.0.0/0      0.0.0.0/0
                                                    state RELATED,ESTABLISHED
  0    0 LAN        all  --  eth1   *       0.0.0.0/0      0.0.0.0/0
  0    0 DMZ        all  --  eth2   *       0.0.0.0/0      0.0.0.0/0
  0    0 INTERNET  all  --  eth0   *       0.0.0.0/0      0.0.0.0/0
```

Sie können sehr leicht feststellen, welchen Weg ein neues Verbindungsaufbaupaket durch Ihr Regelwerk nimmt. Denken Sie daran, dass alle Pakete, die zu aufgebauten Verbindungen gehören, bereits von der ersten Regel in dieser Kette akzeptiert werden. Alle Verbindungsaufbauten werden entsprechend der Netzwerkkarte, über die diese Pakete eingeht, auf die verschiedenen Ketten verteilt. Erreicht das Paket über die Netzwerkkarte `eth2` das System, so wird es an die Kette `DMZ` übergeben, die das Paket prüft, gegebenenfalls akzeptiert oder protokolliert und verwirft.

Falls Sie nun die genauen Regeln betrachten möchten, die das Paket dann prüfen, so genügt es, die Kette `DMZ` anzuzeigen.

```
[root@bibo ~]# iptables -vnL DMZ
Chain DMZ (2 references)
pkts bytes target      prot opt in      out     source          destination
  0    0 ACCEPT      tcp  --  *      eth0    192.168.255.4   0.0.0.0/0
                                                    tcp dpt:53 state NEW
```

```

0 0 ACCEPT  udp  --  *   eth0  192.168.255.4      0.0.0.0/0 ↓
                                     udp dpt:53 state NEW
0 0 ACCEPT  tcp  --  *   eth0  192.168.255.5      0.0.0.0/0 ↓
                                     multiport dports 21,80,443 state NEW
0 0 ACCEPT  tcp  --  *   eth0  192.168.255.2      0.0.0.0/0 ↓
                                     tcp dpt:25 state NEW
0 0 LOG     all  --  *   *      0.0.0.0/0          0.0.0.0/0 ↓
                                     LOG flags 0 level 4 prefix 'Unerlaubt aus der DMZ: '
0 0 REJECT  all  --  *   *      0.0.0.0/0          0.0.0.0/0 ↓
                                     reject-with icmp-port-unreachable

```

Der Vorteil der benutzerdefinierten Ketten in puncto Lesbarkeit sollte Ihnen nun klar geworden sein. Gleichzeitig haben wir aber auch etwas für die Optimierung der Geschwindigkeit getan. Ein Paket, das eine neue Verbindung aus dem Internet zu dem Webserver starten möchte, hätte beim alten Firewall-Skript von 10 Regeln betrachtet werden müssen, bevor es akzeptiert worden wäre. Dies kostet Zeit. Nun handelt es sich lediglich um vier Regeln in der `FORWARD`-Kette und dann um zwei Regeln in der Kette `INTERNET`. Insgesamt sind es also nur sechs Regeln. Bei einer derart geringen Regelzahl ist der Vorteil nur minimal, und häufig ist nicht das Firewall-System das Nadelöhr beim Pakettransport, sondern die Bandbreiten der beteiligten Netzwerkverbindungen. Stellen Sie sich aber eine Firewall mit mehreren hundert Regeln vor, die mit Gigabit-Geschwindigkeit die Pakete verarbeiten soll. Hier kann die Firewall sehr wohl ein Nadelöhr darstellen.

Hinweis



Natürlich ist streng genommen nicht die Anzahl der Regeln, sondern die Anzahl der durchzuführenden Prüfungen interessant. Die meisten Regeln prüfen nicht nur eine Eigenschaft des Pakets, sondern meist gleich mehrere. Diese Prüfungen dauern leider nicht alle gleich lang, so dass objektive Berechnungen nicht möglich sind. Die Anzahl der Prüfungen erlaubt jedoch eine grobe Abschätzung des Aufwands.

9.4 DMZ mit zwei Paketfiltern

Eine dreibeinige Firewall mit DMZ bietet bereits einen recht ordentlichen Schutz. Jedoch besteht immer die Gefahr, dass ein Angreifer auf dem System einen Konfigurationsfehler oder einen Bug in der verwendeten Software (zum Beispiel in dem Linux-Kernel) findet. Erlangt der Angreifer so die Kontrolle über das System, kann er von einer dreibeinigen Firewall aus jedes System in der DMZ, aber auch jedes System in dem internen LAN erreichen. Bei den Systemen in der DMZ muss ein möglicher Angriff mit anschließendem Einbruch immer als Risiko akzeptiert werden muss, da die Systeme im (eingeschränkten) Zugriff des Internets stehen. Dies soll die Firewall für die internen Systeme im LAN gerade verhindern. Sie können

dazu eine zweistufige Firewall mit zwei Paketfiltern einsetzen. Fällt der erste Paketfilter in die Hände des Angreifers, so schützt der zweite Paketfilter noch das interne LAN.

Natürlich besteht die Gefahr, dass der Angreifer auch in den zweiten Paketfilter eindringt. Als Firewall-Administrator hofft man jedoch, dass dies so viel Zeit kostet, dass man Gegenmaßnahmen einleiten kann. Das kann zum Beispiel eine Unterbrechung der Netzwerkverbindung sein. Häufig wird aus diesem Grund auch empfohlen, zwei unterschiedliche Produkte als Paketfilter einzusetzen¹. So kann ein Angreifer nicht denselben Softwarefehler für den Einbruch in beiden Paketfiltern nutzen, sondern muss nach neuen Lücken suchen. Auch Konfigurationsfehler treten selten auf beiden Systemen gleich auf. Dies erhöht also die Schwierigkeit für den Angreifer. Da dies jedoch ein Iptables-Buch ist, möchte ich mich hier darauf beschränken, die Empfehlung zu erwähnen und es Ihnen zu überlassen, ob Sie dieser Empfehlung folgen und welches zusätzliche Produkt Sie auswählen.

Die Implementierung der zweistufigen Firewall erfolgt wie in Abbildung 9.1.

Wie bereits zu Beginn des Kapitels erwähnt wurde (siehe Abschnitt 9.1), befinden sich in der DMZ der Beispielfirma *Nohup.info* jeweils ein Web-, ein E-Mail-, ein cachender DNS- und ein Proxy-Server. Diese Systeme erhalten private IP-Adressen aus dem Bereich 192.168.255.0/24. Dabei soll aus dem Internet ein Zugriff auf den Web- und den E-Mail-Server möglich sein. Aus dem internen Netz ist ein Zugriff auf den Proxy- und den E-Mail-Server notwendig. Der Zugriff auf den Webserver wird über den Proxy-Server realisiert. Ein direkter Zugriff auf das Internet ist nicht vorgesehen. Der äußere Paketfilter erhält als externe IP-Adresse eine feste statische IP-Adresse 3.0.0.1. Alle Dienste werden im Internet unter dieser IP-Adresse angebunden. Das bedeutet, dass ein DNS-Server bei der Frage nach der Adresse *www.nohup.info* die IP-Adresse 3.0.0.1 zurückliefert.

Teilen wir nun die Implementierung auf. Zunächst besprechen wir die Regeln für den äußeren Paketfilter, anschließend für den inneren Paketfilter.

Tipp



Wenn Sie den Hardwareaufwand für zwei Paketfilter scheuen und Ihr System über mindestens drei Netzwerkkarten verfügt, können Sie die beiden Paketfilter auch als virtuelle Systeme auf der Hardware laufen lassen. Hierfür können Sie VMware², User-Mode-Linux³ oder Xen⁴ verwenden. Alle diese Systeme sind in der Lage, die Hardware zu virtualisieren und voneinander getrennte Linux-Systeme auf derselben Hardware zu ermöglichen. Die Sicherheit ist sicherlich nicht so hoch wie bei zwei physikalisch getrennten Systemen, aber besser als bei einer dreibeinigen Firewall.

¹ Das BSI (Bundesamt für Sicherheit in der Informationstechnik) spricht zum Beispiel diese Empfehlung aus.

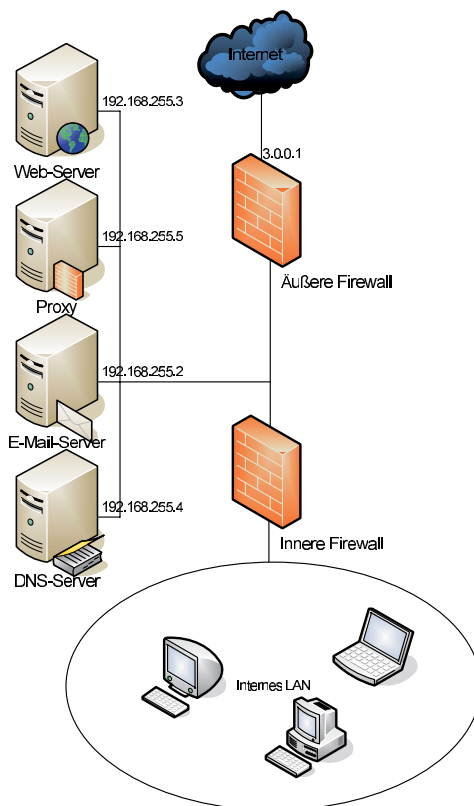


Abbildung 9.1: Eine zweistufige Firewall schützt die DMZ.

9.4.1 Der äußere Paketfilter

Der äußere Paketfilter muss den Zugriff des Internets auf den Webserver und den E-Mail-Server in der DMZ erlauben. Außerdem muss er dem Proxy, dem DNS-Server und dem E-Mail-Server den Zugriff auf die Ressourcen des Internets ermöglichen.

Das Skript beginnt wieder mit der üblichen Definition der Variablen. Die zwei vorhandenen Netzwerkkarten erhalten die Variablennamen `$EXTDEV` und `$DMZDEV`. Ansonsten kann der Skriptkopf des Skripts für die dreibeinige Firewall unverändert übernommen werden.

² Ein kommerzielles System zur Virtualisierung: <http://www.vmware.com>.

³ Ein Linux-Kernel, der auf einem bestehenden Linux-System ein weiteres System booten kann: <http://user-mode-linux.sf.net>.

⁴ Xen erlaubt eine komplette Virtualisierung eines Systems. Es ist nicht mehr möglich, zwischen einem Host und einem Gastsystem zu unterscheiden. Es ist mit der S/390-Großrechner-Technik und dem VMware-Produkt ESX-Server vergleichbar: <http://www.cl.cam.ac.uk/Research/SRG/netos/xen/>.

9.4 DMZ mit zwei Paketfiltern

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

EXTDEV="eth0"

DMZDEV="eth1"

IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
MODPROBE=/sbin/modprobe

MAIL=192.168.255.2
WEB=192.168.255.3
DNS=192.168.255.4
PROXY=192.168.255.5
EXTERN=3.0.0.1

# Lade die Kernelmodule für die Stateful-Inspection des FTP-Protokolls
$MODPROBE ip_conntrack_ftp
$MODPROBE ip_nat_ftp

# Setze Default-Regeln
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Lösche die Filter-Tabelle
$IPTABLES -F

# Lösche die NAT-Tabelle
$IPTABLES -t nat -F
```

Die äußere Firewall ist auch für ein NAT der Adressen zum Internet hin zuständig. Hier können die NAT-Regeln aus dem Skript für die dreibeinige Firewall unverändert übernommen werden:

```
## NAT-Regeln ##
# (1) Proxy-Zugriff
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $PROXY -j SNAT --to-source $EXTERN

# (2) DNS-Zugriff
```

```

$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $DNS -j SNAT --to-source $EXTERN

# (3) Mailserver-Zugriff nach außen
$IPTABLES -t nat -A POSTROUTING -o $EXTDEV -s $MAIL -j SNAT --to-source $EXTERN

# (4) Webserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN -m multiport --dport 80,443 -j DNAT --to-destination $WEB

# (5) Mailserver-Zugriff von außen
$IPTABLES -t nat -A PREROUTING -i $EXTDEV -p tcp -d $EXTERN --dport 25 -j DNAT --to-destination $MAIL

```

Diese Regeln sorgen dafür, dass bei einem Zugriff auf das Internet die Quell-IP-Adressen des DNS-, E-Mail- und Proxy-Servers genattet werden. Auch bei einem Zugriff aus dem Internet auf den E-Mail- und den Webserver wird der Zugriff genattet.

Es fehlen nun noch die Filterregeln. Auch diese können fast unverändert übernommen werden.

```

# Erlaube dem DNS-Server einen Zugriff auf Nameserver im Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p tcp --dport 53 -m state --state NEW -j ACCEPT
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $DNS -p udp --dport 53 -m state --state NEW -j ACCEPT

# Erlaube dem Proxy den Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $PROXY -p tcp -m multiport --dport 21,80,443 -m state --state NEW -j ACCEPT

# Erlaube dem E-Mail-Server den Zugriff auf das Internet
$IPTABLES -A FORWARD -i $DMZDEV -o $EXTDEV -s $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT

# Erlaube dem Internet den Zugriff auf den E-Mail-Server
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT

# Erlaube dem Internet den Zugriff auf den Webserver
$IPTABLES -A FORWARD -i $EXTDEV -o $DMZDEV -d $WEB -p tcp -m multiport --dport 80,443 -m state --state NEW -j ACCEPT

# Lehne Identd-Anfragen ab
$IPTABLES -A INPUT -p tcp --dport 113 -j REJECT

$IPTABLES -A FORWARD -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der DMZ: "

```

```
$IPTABLES -A FORWARD -i $DMZDEV -j REJECT
$IPTABLES -A INPUT -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der DMZ: "
$IPTABLES -A INPUT -i $DMZDEV -j REJECT
```

Dieses Skript genügt für die Konfiguration des äußeren Paketfilters. Ein direkter Zugriff von dem inneren Paketfilter wird nicht erlaubt. So ist der äußere Paketfilter auch vor Angriffen von innen geschützt. Wenn Sie einen zusätzlichen Administrationszugang wünschen, so können Sie zum Beispiel einen Zugang per SSH erlauben:

```
$IPTABLES -A INPUT -i $DMZDEV -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $DMZDEV -m state --state ESTABLISHED -j ACCEPT
```

Natürlich könnten Sie auch diesen Zugang auf eine bestimmte IP-Adresse begrenzen. Dann ist es aber sinnvoll, für diese Adresse auch eine Variable, z.B. ADMIN, zu verwenden.

9.4.2 Der innere Paketfilter

Die Aufgabe des inneren Paketfilters ist es, den Zugriff des internen LANs auf die Systeme in der DMZ zu regeln. Je nach der gewählten E-Mail-Struktur muss er auch eine Verbindung des DMZ-E-Mail-Servers in das LAN ermöglichen.

Ich beschränke mich nun auf die Darstellung des kompletten Skripts und einige wenige Erklärungen. Die einzelnen Regeln wurden in den vorangegangenen Kapiteln bereits ausreichend erläutert. Die Netzwerkkarten in diesem Skript heißen DMZDEV und INTDEV. Das folgende Skript implementiert den inneren Paketfilter:

```
#!/bin/sh
#
# Autor : Ralf Spenneberg
# Version: 0.1
# Datum : 17. Dez 2004
#
# Dieses Skript lädt die Regeln für den inneren Paketfilter

INTDEV="eth1"

DMZDEV="eth2"

IPTABLES=/sbin/iptables
SYSCTL=/sbin/sysctl
MODPROBE=/sbin/modprobe

MAIL=192.168.255.2
PROXY=192.168.255.5
PROXYPORT=3128
```

```
# Setze Default-Regeln
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Lösche die Filter-Tabelle
$IPTABLES -F

## Firewall-Regeln
# Akzeptiere alle aufgebauten Verbindungen
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#####
#
# Zugriff aus dem LAN
#
# Erlaube den Zugriff auf den Proxy
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $PROXY -p tcp --dport $PROXYPORT -m state --state NEW -j ACCEPT

# Erlaube den Zugriff auf den E-Mail-Server
$IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -d $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT

### Bei Verwendung eines internen E-Mail-Servers:
# MAILINTERN=192.168.0.100
# Erlaube den Zugriff auf den E-Mail-Server
# $IPTABLES -A FORWARD -i $INTDEV -o $DMZDEV -s $MAILINTERN -d $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT

# Protokolliere alle anderen Anfragen von innen
$IPTABLES -A FORWARD -i $INTDEV -j LOG --log-prefix "Unerlaubt von innen: "
$IPTABLES -A FORWARD -i $INTDEV -j REJECT

#####
#
# Zugriff aus der DMZ
#
# Erlaube den Transport von E-Mail aus der DMZ in das interne Netz
# $IPTABLES -A FORWARD -i $DMZDEV -o $INTDEV -d $MAILINTERN -s $MAIL -p tcp --dport 25 -m state --state NEW -j ACCEPT
###

# Protokolliere alle anderen Zugriffe aus der DMZ
```

```
$IPTABLES -A FORWARD -i $DMZDEV -j LOG --log-prefix "Unerlaubt aus der DMZ: "  
$IPTABLES -A FORWARD -i $DMZDEV -j REJECT
```

Dieser Paketfilter führt kein NAT durch, da kein Zugriff auf das Internet erfolgt. Dafür muss dann natürlich sichergestellt sein, dass die beteiligten Systeme über entsprechende Einträge in ihren Routingtabellen verfügen. Bei den Systemen im LAN ist dies recht einfach. Bei ihnen wird als Default-Gateway der Paketfilter eingetragen. Bei den Systemen in der DMZ ist es erforderlich, als Default-Gateway den äußeren Paketfilter einzutragen. Lediglich für die Verbindung zum internen LAN ist eine zusätzliche Route auf diesen Systemen erforderlich. Hier muss als Gateway der interne Paketfilter eingetragen werden.

Ich hoffe, dass ich Ihnen mit diesen Skripten ein paar Beispiele geliefert habe, die Sie nun bei der Erstellung Ihrer eigenen Skripten unterstützen. Natürlich handelt es sich hier nur um grobe Gerüste, die häufig um spezielle Anpassungen erweitert werden müssen. Diese hängen aber immer sehr von der Umgebung ab und werden daher hier nicht näher besprochen. Wenn Sie hierzu Hilfe benötigen, so schauen Sie in den Kapiteln zur fortgeschrittenen Konfiguration der Firewall nach. Dort werden dann die einzelnen Protokolle betrachtet und Beispielregeln für deren Filterung gezeigt. Auch fortgeschrittene Fähigkeiten des Iptables-Befehls werden dort besprochen. Ich bin mir sicher, dass Sie dort auch Anhaltspunkte für Ihr Problem finden werden.

