

KAPITEL 13

Eine DMZ – Demilitarized Zone

Mittlerweile haben Sie eine Firewall eingerichtet, die nicht nur Heinz, unseren Heimanwender¹, zufriedenstellen dürfte, sie kommt auch schon recht nah an die Vorstellung von Sören heran, der sein Studentenwohnheim mit dem Internet verbinden möchte². Zwar sind bis jetzt noch keine Proxies zum Zwischenspeichern von Seiten und zum Filtern von Werbung installiert, ein geschütztes Surfen ist mit den bereits vorgestellten Regeln aber möglich.

Lediglich Herr Friedrich, unser Firmenanwender³, hat noch nicht einmal die Hälfte seiner Ziele erreicht. Neben dem Zugang der Anwender zum Internet hat er auch die Aufgabe, einen Webserver an das Internet anzubinden. Dieser soll ebenfalls durch eine Firewall vor Angriffen aus dem Internet geschützt werden. Dies bedeutet aber nicht, daß der Server im lokalen Netz aufgestellt werden soll. Zu groß ist die Gefahr, daß der Server trotz aller Vorsichtsmaßnahmen kompromittiert wird und dem erfolgreichen Angreifer dann als Brückenkopf im lokalen Netz dient. Vielmehr soll ein eigener Netzwerkstrang eingerichtet werden, der ebenfalls durch eine Firewall geschützt ist, von dem aus aber keine Verbindungen in das lokale Netz aufgebaut werden dürfen: eine DMZ.

Das Netzwerk planen

Bevor wir mit dem Einrichten einer DMZ beginnen, müssen wir uns als erstes über die Struktur des neu einzurichtenden Netzes klarwerden. Da wir einen Webserver betreiben wollen, brauchen wir eine permanente Anbindung an das Internet. Auch brauchen wir feste IP-Adressen, die sich nicht ändern. Wir müssen daher bei unserem Provider einen Bereich von IP-Adressen einkaufen, den wir für diese Zwecke nutzen können. Dann müssen wir festlegen, welche dieser Adressen wir für den Webserver, die verschiedenen Interfaces der Firewall und bestimmte administrative Adressen (Netzwerk- und Broadcast-Adresse) benutzen wollen. Um die Sache anschaulicher zu machen, wollen wir im folgenden einmal sehen, wie Herr Friedrich diese Aufgabe angegangen ist.

1 Siehe Kapitel 2, Unterabschnitt *Der Privathaushalt*, ab Seite 9

2 Siehe Kapitel 2, Unterabschnitt *Das Studentenwohnheim*, ab Seite 11

3 Siehe Kapitel 2, Unterabschnitt *Die Firma*, ab Seite 12

Herr Friedrich mietet von seinem Provider den Adreßbereich 10.0.0.0 bis 10.0.0.7⁴. Allerdings können die Adressen 10.0.0.0 und 10.0.0.7 nicht genutzt werden, da es sich hierbei um die Netzwerk- bzw. die Broadcast-Adresse des Subnetzes handelt. Auch die Adresse 10.0.0.1 wird schon von einem Router benutzt, den der Provider Herrn Friedrich zur Verfügung gestellt hat und der die Verbindung zum Internet darstellt. Damit bleiben für Herrn Friedrich nur noch die Adressen 10.0.0.2 bis 10.0.0.6 übrig.

Nun möchte Herr Friedrich einen Teil dieser Adressen für seine DMZ benutzen. Dabei muß er einige Dinge beachten:

- Die Anzahl der Adressen in einem IP-Subnetz ist immer eine Potenz von 2.⁵
- Ein Subnetz kann nicht an einer beliebigen Adresse anfangen. Ein Subnetz mit a Adressen fängt entweder an der Adresse 0 an oder mit einer Adresse, die ein Vielfaches von a ist.⁶
- Da immer zwei Adressen von der Netzwerk- und der Broadcast-Adresse blockiert werden, enthält ein sinnvolles Subnetz mindestens 4 Adressen.

Hieraus folgt, daß für den Adreßbereich seiner DMZ nur eines der beiden 4er-Subnetze in Frage kommt, in die er sein 8er-Subnetz aufteilen kann. Entweder die DMZ benutzt den Bereich 10.0.0.0 bis 10.0.0.3 oder 10.0.0.4 bis 10.0.0.7. Er kann seine DMZ z.B. nicht in den Bereich 10.0.0.2 bis 10.0.0.5 legen.

Damit sind Kollisionen vorprogrammiert. Wählt er den unteren Bereich, so haben sowohl die DMZ als auch das 8er-Subnetz dieselbe Netzwerk-Adresse. Wählt er dagegen den oberen, so ist die Broadcast-Adresse der beiden identisch. Beides ist aber nicht wirklich kritisch. Weder die sogenannte Netzwerk-Adresse noch die Broadcast-Adresse werden in unserem Szenario wirklich benötigt.

Anders sieht es mit der Adresse des Routers aus. Damit die Firewall Pakete ins Internet leiten kann, muß sie wissen, über welchen Netzwerkstrang sie den Router erreichen kann. Würde die Adresse aber sowohl in der DMZ als auch im äußeren Netz verwendet, so wäre eine verlässliche Zustellung von Paketen nicht mehr möglich. Eine Verwendung des unteren Subnetzes für die DMZ scheidet damit aus. Damit ergibt sich die in Abbildung 13-1 dargestellte Aufteilung der Adressen.

4 Dieser Adreßbereich ist kein Beispiel aus dem wirklichen Leben, sondern frei erfunden. Jede Ähnlichkeit mit tatsächlichen im Internet verwendeten Adreßbereichen wäre nicht nur rein zufällig, sondern extrem unwahrscheinlich, da dieser Bereich für die Benutzung in lokalen Netzen reserviert ist.

5 Die ersten $(32 - n)$ Bits einer Adresse beschreiben das Netz, die letzten n Bits den Rechner. Somit sind prinzipiell 2^n Rechneradressen in einem Netz vorhanden.

6 Adressen in einem Subnetz werden immer aus $(32 - n)$ unveränderlichen (Netzwerk-Adresse) und n veränderlichen Bits (Adressen der einzelnen Rechner) gebildet. Dabei ist $a = 2^n$. Die veränderlichen Bits sind die niedrigstwertigen $(2^n - 1 \dots 1)$. Da jedes der $(32 - n)$ Bits der Rechneradresse ein Vielfaches von a darstellt ($2^{16} \dots 2^n$), ist auch die Summe der Bits immer ein Vielfaches von a . (Um es abzukürzen, definieren wir hier auch 0 als ein Vielfaches von a .)

Die Adresse eines Rechners ist wiederum die Summe der Netzwerkbits und der Bits seiner Rechneradresse. Die niedrigste Rechneradresse ist 0. 0 addiert zu einem Vielfachen von a ergibt ein Vielfaches von a .

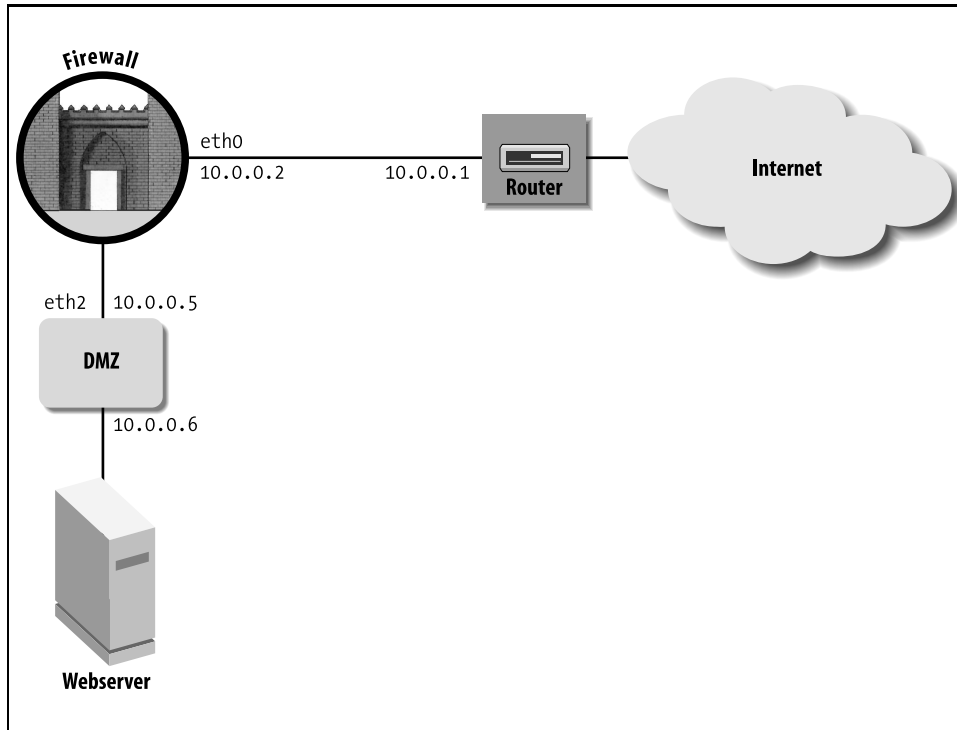


Abbildung 13-1: Die Planung einer DMZ

Die Adresse 10.0.0.3 wird nicht genutzt und könnte prinzipiell von einer weiteren Firewall verwendet werden. Damit wäre es z. B. möglich, eine eigene Firewall zum Schutz des lokalen Netzes aufzusetzen. Herr Friedrich verzichtet aber momentan darauf und benutzt statt dessen eine weitere Netzwerkkarte, um das lokale Netz an dieselbe Firewall anzuschließen, die auch für den Webserver zuständig ist.

Proxy-ARP

Nun, da wir wissen, welche Netze wir konfigurieren müssen, sollte das Einrichten einer zusätzlichen Netzwerkkarte für die DMZ kein Problem sein. Wie dies geht, haben wir ja schon in Kapitel 10, Abschnitt *Konfiguration der Netzwerkkarte*, ab Seite 219 gesehen.

Damit ist unser Webserver aber noch nicht aus dem Internet ansprechbar. Wir können dies testen, indem wir:

- statt des Routers einen Testrechner anschließen, dessen Netzwerkkarte für die Adresse unseres Routers (hier: 10.0.0.1) und die uns vom Provider zugewiesene Subnetzmaske (hier 255.255.255.248) konfigurieren,

- die Firewallregeln für den Test außer Kraft setzen⁷ und
- einen provisorischen Webserver in unserer zukünftigen DMZ aufsetzen, der unsere Firewall als Default-Router benutzt.

Versuchen wir nun, unseren Webserver vom Testrechner aus mit ping anzusprechen, so haben wir keinen Erfolg:

```
# ping -c 1 10.0.0.6
PING 10.0.0.6 (10.0.0.6): 56 data bytes

--- 10.0.0.6 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
```

Wenn wir einen Blick auf die Routing-Tabellen des Testrechners werfen, wird auch klar, woran dies liegt:

```
# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.0.0.0 * 255.255.255.248 U 0 0 0 eth0
localnet * 255.0.0.0 U 0 0 0 lo
default 10.0.0.1 0.0.0.0 UG 0 0 0 eth0
```

Wie später auch der Router des Providers, so will unser Testrechner alle Pakete für Adressen im Bereich 10.0.0.0 bis 10.0.0.7 direkt zustellen, ohne einen Router zu bemühen. Im Falle unseres Webserver ist dies aber nicht möglich, da er sich vom Testrechner aus gesehen hinter dem Firewallrechner befindet. Die Pakete dürfen also nicht direkt zugestellt werden, sondern müssen der Firewall übergeben werden, die sie dann weiterleitet.

Man könnte dieses Problem lösen, indem man in die Routing-Tabelle des Testrechners einen passenden Eintrag einfügt:

```
# route add 10.0.0.6 gw 10.0.0.2
```

Dies ist aber in unserem Fall keine Lösung, da der Testrechner hier den Router unseres Providers simuliert. Im Normalfall wird uns der Provider aber nicht erlauben, seinen Router umzukonfigurieren.

Uns bleibt nur, dem Router vorzugaukeln, die Firewall wäre der Webserver. Sind die Pakete erst einmal an der Firewall angekommen, so übernimmt deren Routing dann schon die eigentliche Zustellung der Pakete an den Webserver.

Hierzu können wir ARP, das Address Resolution Protocol, nutzen. Wann immer ein Rechner ein Paket an einen Router oder einen Zielrechner zustellen will, muß er erst herausfinden, unter welcher MAC-Adresse dieser zu erreichen ist. Dazu sendet er an alle Rechner im lokalen Netz eine Anfrage, die seine eigene MAC-Adresse, seine eigene IP-Adresse und die IP-Adresse des gesuchten Rechners enthält. Normalerweise wird daraufhin der gesuchte Rechner antworten, indem er ein Paket schickt,

⁷ Dies bedeutet, alle Regeln zu löschen und ACCEPT als Policy für alle Chains einzutragen (inklusive der für das Forwarding zuständigen).

in dem er die MAC-Adresse des Anfragenden durch seine eigene Adresse ersetzt hat.⁸

Grundsätzlich gibt es aber keinen Mechanismus, der verhindert, daß statt des gesuchten ein anderer Rechner antwortet. Dies können wir ausnutzen, indem wir die Firewall anweisen, nicht nur für sich selbst, sondern auch für den Webserver zu antworten:

```
# arp -i eth0 -Ds 10.0.0.6 eth0 pub
```

Die Optionen haben die folgende Bedeutung:

- i <Interface> gibt an, auf welchem Netzwerk-Interface Fragen beantwortet werden sollen.
- D legt fest, daß wir nicht direkt die zu sendende MAC-Adresse angeben wollen, sondern daß die MAC-Adresse des angegebenen Interfaces benutzt werden soll (das zweite eth0).
- s <IP-Adresse> <MAC-Adresse/Interface> ordnet einer IP-Adresse eine MAC-Adresse zu. pub bedeutet, daß wir nicht einfach nur intern eine Zuordnung von MAC-Adresse zu IP-Adresse speichern wollen, sondern daß diese auch dazu benutzt werden soll, Anfragen anderer Rechner zu beantworten.

Mit dem Befehl arp können wir auch überprüfen, ob die Zuordnung wie gewünscht in den ARP-Cache eingetragen wurde:

```
# arp -n
Adresse          HWTyp HWAdresse  Flags  Maske Iface
10.0.0.6         ether 00:10:A7:0E:A2:08 C      Maske eth2
10.0.0.6         *      *           MP      eth0
```

Hier enthält der erste Eintrag die tatsächliche MAC-Adresse unseres Webserver, die nach einer früheren Anfrage für eine gewisse Zeit zwischengespeichert wurde (Flag C). Der zweite Eintrag wurde von uns dagegen permanent eingetragen (Flag M) und wird vom System auch dazu benutzt, Anfragen anderer Rechner zu beantworten (Flag P).

Damit kann der Webserver nun von unserem Testrechner aus angesprochen werden:

```
ping -c 1 10.128.0.2
PING 10.128.0.2 (10.128.0.2): 56 data bytes
64 bytes from 10.128.0.2: icmp_seq=0 ttl=254 time=2.8 ms

--- 10.128.0.2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 2.8/2.8/2.8 ms
```

Die umgekehrte Richtung sollte ebenfalls problemlos funktionieren.

Wenn Sie den ARP-Befehl nicht jedesmal von Hand eingeben wollen, so können Sie auch das folgende Runlevel-Skript benutzen. Verlinken Sie es so, daß es nach ethdevs aufgerufen wird. Unter SuSE-Linux können Sie es einfach mit insserv installieren:

⁸ Siehe Kapitel 3, Abschnitt ARP, ab Seite 20

```
#!/bin/sh
#####
#
# arpit
#
#   Beantwortet ARP-Anfragen nach einem Rechner im DMZ-Netz auf dem
#   externen Interface mit der eigenen MAC-Adresse
#
# Usage: arpit {start|stop}
#
#
# Copyright (C) 2003 Andreas G. Lessig
#
# Lizenz: GPL v2 oder h"ohere Version
#
#####

### BEGIN INIT INFO
# Provides:          arpit
# Required-Start:    $network
# Should-Start:
# Required-Stop:     $network
# Should-Stop:
# Default-Start:     3 5
# Default-Stop:      0 1 2 6
# Short-Description: beantwortet ARP-Anfragen f"ur Server in der DMZ
# Description:       beantwortet ARP-Anfragen f"ur Server in der DMZ
### END INIT INFO

# -----
# Grundeinstellungen
# -----

# Kurzbezeichnungen

if test -x /sbin/arp
then
    ARP=/sbin/arp
elif test -x /usr/sbin/arp
then
    ARP=/usr/sbin/arp
else
    echo "ERROR: arp not found"
    exit 1
fi

# Ein paar grunds"atzliche Daten
# Diese m"ussen an die eigenen Bed"urfnisse angepa"st werden

# - Das externe Interface

EXTIF="eth0"

# - Unser Server

DMZSERVER="10.0.0.6"
```

```

# -----
# ARP
# -----

case $1 in
start)
  for s in $DMZSERVER
  do
    echo "$0: ARPe f"ur $s auf $EXTIF"
    $ARP -i $EXTIF -Ds $s $EXTIF pub
  done
  ;;
stop)
  for s in $DMZSERVER
  do
    echo "$0: entferne $s aus dem ARP-Cache"
    $ARP -i $EXTIF -d $s pub
  done
  ;;
*)
  echo "Usage: $0 {start|stop}"
  exit 1
  ;;
esac

```

Die Variable *EXTIF* gibt hierbei an, auf welchem Interface die ARP-Anfragen beantwortet werden sollen, während *DMZSERVER* die IP-Adresse des Servers enthält. Betreiben Sie mehrere Server in Ihrer DMZ, so können Sie auch mehr als eine IP-Adresse in die Variable eintragen.

Reverse Proxies

Man kann Proxies nicht nur dazu verwenden, die Internet-Zugriffe der Klienten im lokalen Netz zu filtern, es funktioniert auch umgekehrt. Man kann einen Proxy-Server einrichten, der Zugriffe nur an einen bestimmten eigenen Server weiterleitet. Wenn man nun noch den direkten Zugriff auf den eigentlichen Server unterbindet, dann muß jeder Zugriff auf den Server über den Proxy-Server erfolgen. Dies erlaubt es, mit Hilfe entsprechender Filterregeln unerwünschte Zugriffe auf den eigentlichen Server zu unterbinden.

Einen solchen Proxy, der nicht dazu dient, den Zugriff eines beschränkten Klientenkreises auf beliebige Server zu kontrollieren, sondern der umgekehrt den Zugang für beliebige Klienten auf einen einzelnen Server oder einige wenige Server vermittelt, nennt man *Reverse Proxy*.

Derzeit existieren noch nicht viele Reverse Proxy-Implementationen. Teilweise werden Reverse Proxies in Verbindung mit Webservern eingesetzt, um die Last auf dem eigentlichen Server zu verringern. Die Aufgabe des Reverse Proxy liegt dabei weniger darin, Zugriffe einzuschränken, sondern er soll häufiger geladene Seiten zwischenspeichern. Ein Beispiel für einen cachenden Proxy, der in dieser Weise eingesetzt werden kann, stellt z. B. der squid dar.

Im folgenden soll es aber um einen anderen Einsatzzweck gehen. Wir wollen einen Reverse Proxy konfigurieren, der unerwünschte Zugriffe unterbinden kann. Die einzige mir bekannte Software, die gezielt zu diesem Zweck entwickelt wurde, ist die SuSE Proxy-Suite (siehe Kapitel 14, Unterabschnitt *ftp-proxy aus der SuSE Proxy-Suite*, ab Seite 418). Ursprünglich war diese als Satz von Reverse Proxies für verschiedene Protokolle geplant. Derzeit existiert aber nur ein FTP-Proxy, dem man recht detailliert vorgeben kann, welche FTP-Kommandos erlaubt und welche verboten sind.

Damit bleibt noch die Frage, wo wir den Reverse Proxy implementieren.

Wir könnten den Proxy auf dem gleichen Rechner installieren, auf dem auch die Server für die direkt erreichbaren Dienste laufen. Der zu schützende Dienst hätte dann einen eigenen Rechner in der DMZ, der aus dem Internet nicht direkt erreichbar wäre.

Den Dienst auf demselben Rechner zu installieren, auf dem auch der Proxy läuft, ist in der Regel nicht möglich, da FTP-Proxies im Gegensatz zu HTTP-Proxies normalerweise den gleichen Port benutzen, den auch der eigentliche Server verwendet.

Aus Sicht der Firewall bedeutet dies, daß sich in der DMZ ein Server mehr befindet. Wir müssen daher noch gezielt eine Regel einfügen, die den direkten Zugriff aus dem Internet auf den zu schützenden Server unterbindet. Darüber hinaus müssen wir u. U. mehr IP-Adressen von unserem Provider einkaufen.

Wir könnten auch den Server in der DMZ mit zwei Netzwerkkarten ausstatten. Die zweite Netzwerkkarte verbindet ihn mit einem weiteren LAN, in dem sich der geschützte Server befindet. Dieses zweite LAN hat private Subnetz-Adressen (z. B. *192.168.3.x*), wodurch keine weiteren IP-Adressen vom Provider benötigt werden. Der Server in der DMZ kann so wie gehabt als Webserver fungieren, statt eines FTP-Servers leitet aber der *ftp-proxy* die Anfragen in das zweite LAN weiter.

Aus Sicht der Firewall ist dieses Szenario identisch mit einer Lösung ohne Reverse Proxy. Preislich entfallen im Gegensatz zur ersten Lösung die Kosten für zusätzliche IP-Adressen.

Schließlich können wir den Reverse Proxy auf unserer Firewall installieren. Dies erspart uns die Zusatzkosten eines weiteren Servers, hat aber den Nachteil, daß wir nun auf der Firewall einen Dienst anbieten, der aus dem Internet zugreifbar ist. Enthält dieser Dienst einen schweren Programmierfehler, so kann dies prinzipiell zur Kompromittierung der Firewall führen.

Der hier betrachtete *ftp-proxy* wurde speziell als Reverse Proxy geschrieben. Er läuft mit eingeschränkten Benutzerrechten, und man kann ihn in einer *chroot*-Umgebung betreiben, wodurch er auf keine Dateien außerhalb eines bestimmten Verzeichnisses zugreifen kann.

Diese Eigenschaften lassen das Risiko in diesem Fall beherrschbar erscheinen. Man muß aber klar sagen, daß jeder aus dem Internet zugreifbare Dienst auf einer Firewall wie der hier beschriebenen eine potentielle Angriffsmöglichkeit ist. Gelingt es, die Firewall zu kompromittieren, so steht dem Angreifer kein Hindernis mehr im Weg bei dem Versuch, in das lokale Netz zu gelangen.

Falls Sie noch einen 2.2er Kernel benutzen, muß ich Ihnen allerdings generell von dem Betrieb eines Reverse Proxys für FTP auf der Firewall abraten. ipchains kennt kein Stateful Packet Filtering. Beim passiven FTP können beliebige Verbindungen der Klienten zu hohen Ports der Firewall geöffnet werden. Wenn Sie Ihre Firewall nicht sehr sorgfältig konfigurieren und sicherstellen, daß jeder Dienst auf der Firewall, der aus dem Internet nicht erreichbar sein soll, durch eine eigene Filterregel geschützt ist, dann ist ein Einbruch in die Firewall vorprogrammiert.



Davon abgesehen würde ich im konkreten Fall wahrscheinlich erst einmal überlegen, ob das Geld für einen weiteren Rechner vorhanden ist. Ist dies der Fall, würde ich die zweite Variante vorziehen. Diese Entscheidung kann und will ich Ihnen aber nicht abnehmen. Im folgenden werden wir daher bei der Erstellung der Firewallskripte alle oben aufgeführten Fälle mit Ausnahme der Kombination von Reverse Proxy auf der Firewall mit ipchains berücksichtigen.

Paketfilter

Im folgenden werden wir die nötigen Filterregeln definieren, um die Anbindung unserer DMZ an das Internet sicher zu gestalten. Zuvor müssen wir uns aber noch darüber klarwerden, was wir eigentlich erreichen wollen. Wir können die Server in unserer DMZ mit einer Firewall nur bedingt gegen Angriffe schützen. Viele Angriffe gegen Server sehen für die Firewall wie ganz normale legale Anfragen aus. Angriffe durch Firewalling zu verhindern, würde daher bedeuten, den Dienst komplett einzustellen.

Zwar kann eine sicherheitsbewußte Konfiguration des jeweiligen Servers die Gefahr deutlich verringern, es besteht aber immer die Gefahr, daß ein Fehler in der Software all unsere Anstrengungen zunichte macht. Aus diesem Grund müssen wir immer damit rechnen, daß unsere Server kompromittiert werden. Ist dies erst geschehen, können sie vom Angreifer als Ausgangsbasis für weitere Angriffe genutzt werden.

Um dies zu verhindern, sollten wir uns ernsthaft überlegen, keine Klientenzugriffe vom Server aus in das Internet zuzulassen. Allerdings bedeutet dies auch, daß es uns nicht möglich ist, vom Webserver aus z. B. Software-Updates herunterzuladen. Wir müßten die Downloads von einem anderen Rechner aus durchführen und die Dateien dann offline z. B. mit Hilfe einer CD auf den Server bringen.

Wenn dies nicht durchsetzbar ist, weil wir den Server z. B. nicht selbst betreiben und der Zuständige uneinsichtig ist, dann sollten wir zumindest die Anzahl der Server im Internet einschränken, auf die aus der DMZ zugegriffen werden kann. Wir könnten z. B. nur Zugriffe auf bestimmte Domains erlauben, die den Herstellern der Software gehören, die auf den DMZ-Servern läuft.

Schließlich gilt es noch, die Regeln für den Datenverkehr zwischen DMZ und lokalem Netz festzulegen. Dabei sollten wir Server in der DMZ generell wie Server im Internet behandeln. Da wir grundsätzlich davon ausgehen müssen, daß sie bereits kompromittiert sein könnten, dürfen wir ihnen nicht gestatten, auf Rechner im lokalen Netz zuzugreifen.

Im folgenden werden wir am Beispiel eines Web- und FTP-Servers einmal sehen, wie Filterregeln für eine DMZ konkret aussehen können.

Paketfilterung mit ipchains

Wir gehen in unserem Szenario davon aus, daß die Firewall, welche die Benutzer beim Surfen im Internet benutzen, auch die ist, durch welche die DMZ geschützt wird. Daher liegt es nahe, das schon vorhandene Firewallskript so zu erweitern, daß es auch die Bedürfnisse der DMZ erfüllt. Wir werden also von dem in Kapitel 11 ab Seite 269 beschriebenen Skript `pfilter.ipchains` ausgehen und die Teile anfügen, die wir zusätzlich benötigen. Beginnen wir damit, daß wir uns die wesentlichen Neuerungen ansehen. Das komplette Skript finden Sie dann am Ende des Kapitels.

Beginnen wir mit den Variablen, die die verwendeten Netzwerkadressen definieren. Hier benötigen wir zusätzliche Angaben für das Netzwerk-Interface, über das die DMZ angebunden ist:

```
# - Das DMZ-Netz
DMZIP="10.0.0.5"
DMZMASK="255.255.255.252"
DMZIF="eth2"
DMZNET="$DMZIP"/"$DMZMASK"
```

Weiterhin sollten wir definieren, auf welchen Ports TCP-basierte Dienste auf eingehende Verbindungen warten. Wenn Sie einen FTP-Server betreiben, definieren Sie hier bitte keinen Port. Dieses Protokoll ist etwas komplizierter und wird gleich gesondert behandelt.

Im folgenden wollen wir einen HTTP-Server (Port 80) betreiben, der auch SSL-Zugriffe erlaubt (Port 443). Wollen Sie weitere Server für den Zugriff aus dem Internet bereitstellen, die ihren Datenverkehr über eine einfache TCP-Verbindung abwickeln, so tragen Sie mit Leerzeichen getrennt die entsprechenden Ports ein (z. B. 25 für SMTP):

```
# - TCP-Ports des Servers, die offen sein müssen
DMZPORTS="80 443"
```

Nun sollten wir noch definieren, ob und auf welche Weise wir FTP-Verbindungen in die DMZ zulassen wollen. Dazu definieren wir die Variable `DMZFTP`:

```
# - Art des FTP-Servers, mögliche Werte:
#
# NONE - keiner
# SRV - in der DMZ wird ein Server betrieben, keine Filterung
DMZFTP="SRV"
```

In einigen Fällen kann es vorkommen, daß unser FTP-Server einen ungewöhnlichen Port für seine Datenverbindungen benutzt. Dies ist insbesondere der Fall, wenn es sich nicht wirklich um den eigentlichen FTP-Server handelt, sondern noch ein Proxy vorgeschaltet ist. In diesem Fall müssen wir angeben, um welchen Port es sich handelt:

```
# - Datenport, den der FTP-Server in der DMZ benutzt. Falls Sie in der
# DMZ einen FTP-Proxy betreiben, kann es sein, daß Sie diesen Wert
# anpassen müssen. Normale Server benutzen Port 20.
```

```
DMZFTPDATAPOrt="20"
```

Falls wir in der DMZ einen FTP-Server betreiben wollen, der nicht direkt aus dem Internet zugreifbar sein soll, sondern der nur über einen Proxy auf einem anderen Rechner in der DMZ erreichbar sein soll, so müssen wir dem Skript seine Adresse mitteilen. Dies geschieht in der Variable *DMZIPPROT*:

```
# - Server in der DMZ, die nicht direkt zugreifbar sein sollten:
# (Mehrfachnennungen sind möglich, z.B. ä.b.c.d w.x.y.z")
```

```
DMZIPPROT=""
```

Es kann aber auch sein, daß auf unserem Server Dienste auf hohen Ports aktiv sind, auf die nicht aus dem Internet zugegriffen werden soll. In so einem Fall müssen wir den Zugriff explizit verbieten, wenn wir einen FTP-Server betreiben. Für diesen müssen wir nämlich den Zugriff von hohen Ports im Internet auf hohe Ports des DMZ-Servers erlauben.

Die folgenden beiden Variablen erlauben es, den Zugriff auf solche Ports explizit zu sperren:

```
# - Hohe Ports des Servers, die zusätzlich geschützt werden müssen
```

```
DMZUDPPROT=""
DMZTCPPROT="8080 2049"
```

DMZUDPPROT enthält die zu schützenden UDP-Ports, *DMZTCPPROT* ihre TCP-Gegenstücke.

Die Regeln zur Protokollierung gespoofter Pakete sind etwas komplizierter geworden. Schließlich existiert ein weiteres Teilnetz, das berücksichtigt werden muß. Darüber hinaus brauchen wir auch noch eine Regel, die verhindert, daß unser DMZ-Server gespoofte Pakete verschickt. Dies könnte z. B. dann geschehen, wenn er von einem Angreifer übernommen und als Ausgangsbasis für weitere Angriffe benutzt wird.

Der Symmetrie halber habe ich zusätzlich eine solche Anti-Spoofing-Regel für Rechner aus dem internen Netz eingefügt:

```
# Protokollierung gespoofter Pakete

$R -A input -i ! "$INTIF" -s "$INTNET" -j DENY -l
$R -A input -i "$INTIF" -s ! "$INTNET" -j DENY -l
$R -A input -i ! lo -s 127.0.0.1 -j DENY -l
$R -A input -i ! "$DMZIF" -s "$DMZNET" -j DENY -l
$R -A input -i "$DMZIF" -s ! "$DMZNET" -j DENY -l
```

Als nächstes müssen wir Chains für die Regeln definieren, die gelten sollen, wenn Pakete zwischen Internet und DMZ vermittelt werden:

```
# - DMZ
$R -N ext2dmz
$R -N dmz2ext
```

Nun gilt es, eingehende Pakete auf die Chains zu verteilen. Zusätzlich zu den bisher behandelten Fällen sind mehrere neue hinzugekommen.

Zuerst einmal kann es sein, daß eine Kommunikation zwischen der DMZ und Rechnern im lokalen Netz oder der Firewall stattfinden soll. In diesen Fällen wird die DMZ wie das Internet behandelt. D. h., dieselben Chains wie bei Internet-Zugriffen werden angewendet. Eine direkte Kontaktaufnahme des DMZ-Rechners mit Rechnern im lokalen Netz wird verhindert. Solche Verbindungen werden grundsätzlich maskiert:

```
$R -A input -i "$DMZIF" -d "$DMZIP" -j ext-in
$R -A input -i "$DMZIF" -d "$EXTIF" -j ext-in
$R -A input -i "$DMZIF" -d "$INTNET" -l -j DENY
$R -A forward -i "$DMZIF" -s "$INTNET" -j int-fw
$R -A output -i "$DMZIF" -s "$DMZIP" -j ext-out
```

Als nächstes folgen Regeln, die explizit Verbindungen zwischen DMZ und Internet behandeln. Solche Pakete werden in der input- und output-Chain angenommen und in der forward-Chain gefiltert.

```
$R -A input -i "$DMZIF" -d !"$DMZIP" -j ACCEPT
$R -A input -i "$EXTIF" -d "$DMZNET" -j ACCEPT
$R -A forward -i "$DMZIF" -d "$DMZNET" -j ext2dmz
$R -A forward -i "$EXTIF" -s "$DMZNET" -j dmz2ext
$R -A output -i "$DMZIF" -s !"$DMZIP" -j ACCEPT
$R -A output -i "$EXTIF" -s "$DMZNET" -j ACCEPT
```

Schließlich bleibt noch der bereits bekannte Fall, daß Rechner im lokalen Netz auf Rechner im Internet zugreifen wollen:

```
$R -A input -i "$INTIF" -j int-in
$R -A input -i "$EXTIF" -j ext-in
$R -A forward -i "$EXTIF" -s "$INTNET" -j int-fw
$R -A output -i "$INTIF" -j int-out
$R -A output -i "$EXTIF" -j ext-out
```

Wie unsere Firewall, so muß auch unser Server ICMP-Fehlermeldungen empfangen und senden können. Allerdings müssen wir auch hier davon ausgehen, daß ein Angreifer den Server kompromittiert haben könnte und ihn nun für Angriffe einsetzt. Wir müssen daher nicht nur den Empfang, sondern auch das Senden von ICMP-Paketen auf jene Pakete einschränken, die wir als unbedenklich oder notwendig eingestuft haben:

```
# ICMP
$R -A ext2dmz -p icmp --icmp-type 0 -j ACCEPT
$R -A ext2dmz -p icmp --icmp-type 3 -j ACCEPT
$R -A ext2dmz -p icmp --icmp-type 8 -j ACCEPT
$R -A ext2dmz -p icmp --icmp-type 11 -j ACCEPT
$R -A ext2dmz -p icmp --icmp-type 12 -j ACCEPT
```

```
$R -A dmz2ext -p icmp --icmp-type 0 -j ACCEPT
$R -A dmz2ext -p icmp --icmp-type 3 -j ACCEPT
$R -A dmz2ext -p icmp --icmp-type 8 -j ACCEPT
$R -A dmz2ext -p icmp --icmp-type 11 -j ACCEPT
$R -A dmz2ext -p icmp --icmp-type 12 -j ACCEPT
```

Unser Server muß auch in der Lage sein, DNS-Anfragen zu stellen:

```
# DNS-Anfragen der DMZ-Server

for DNS in $DNSSERVER
do
    $R -A dmz2ext -p udp -d "$DNS" 53 -j ACCEPT
    $R -A ext2dmz -p udp -s "$DNS" 53 -j ACCEPT

    $R -A dmz2ext -p tcp -d "$DNS" 53 -j ACCEPT
    $R -A ext2dmz -p tcp -s "$DNS" 53 ! -y -j ACCEPT
done
```

Wir haben ja bereits definiert, welche Server und Ports aus dem Internet nicht zugreifbar sein sollen. Nun kommt es noch darauf an, die entsprechenden Variablen auszuwerten und in Regeln umzusetzen:

```
# Schutz von Servern, die nicht direkt zugreifbar sein sollen

for i in $DMZIPPROT
do
    $R -A ext2dmz -p tcp -d "$i" -l -j DENY
done

# Schutz von Diensten, die nur intern zugreifbar sein sollen

for p in $DMZUDPPROT
do
    $R -A ext2dmz -p udp --dport "$p" -l -j DENY
done

for p in $DMZTCPPROT
do
    $R -A ext2dmz -p tcp --dport "$p" -l -j DENY
done
```

Genauso einfach können Zugriffe auf definierte TCP-Ports freigeschaltet werden:

```
# Einfache TCP-Server

for p in $DMZPORTS
do
    $R -A dmz2ext -p tcp --sport "$p" --dport 1024:65535 -j ACCEPT
    $R -A ext2dmz -p tcp --dport "$p" -j ACCEPT
done
```

Wenn wir einen FTP-Server betreiben wollen, so müssen wir zusätzliche Regeln einrichten.

```
# FTP

case "$DMZFTP" in
SRV)

# - Server in der DMZ ohne Filterung
# -- Kontrollverbindung

    $R -A dmz2ext -p tcp --sport 21 --dport 1024:65535 -j ACCEPT
    $R -A ext2dmz -p tcp --dport 21 -j ACCEPT

# -- Datenverbindungen
# --- aktiv

    $R -A dmz2ext -p tcp --sport "$DMZFTPDATAPOrt" \
    --dport 1024:65535 -j ACCEPT
    $R -A ext2dmz -p tcp --sport 1024:65535 \
    --dport 20 ! -y -j ACCEPT

# --- passiv

    $R -A ext2dmz -p tcp --sport 1024:65535 \
    --dport 1024:65535 -j ACCEPT
    $R -A dmz2ext -p tcp --sport 1024:65535 \
    --dport 1024:65535 ! -y -j ACCEPT
    ;;
esac
```

Wie in den anderen Chains, so müssen auch in den DMZ-Chains Regeln eingetragen werden, die greifen, wenn keine andere Regel der Chain zutrifft.

```
$R -A ext2dmz -s 0.0.0.0/0 -j DENY -1
$R -A dmz2ext -s 0.0.0.0/0 -j DENY -1
```

Fügen wir nun alles zusammen, so bekommen wir ein Skript, das wir anstelle des Skripts `pfilter` benutzen können. Denken Sie aber daran, die Variablen am Anfang des Skripts Ihren konkreten Netzwerkdaten anzupassen und gegebenenfalls die Kommentarzeichen vor den Regeln für den Zugriff auf die Proxies und den DNS-Server zu entfernen:

```
#!/bin/sh
#####
#
# dmz.ipchains
#
#   Filterregeln für eine DMZ und ein maskiertes lokales Netz mit
#   ipchains
#
# Usage: dmz {start|stop}
#
# Copyright (C) 2003 Andreas G. Lessig
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the license, or
# (at your option) any later version.
#
```

```
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
#####

### BEGIN INIT INFO
# Provides:          dmz-ipc
# Required-Start:    $local_fs
# Should-Start:
# Required-Stop:     proccnf
# Should-Stop:
# Default-Start:     2 3 5
# Default-Stop:      0 1 6
# Short-Description: Firewalling f"ur ein LAN und eine DMZ
# Description:       Firewalling f"ur ein LAN und eine DMZ
### END INIT INFO

# -----
# Grundeinstellungen
# -----

# Kurzbezeichnungen

R=/sbin/ipchains

# Ein paar grunds"atzliche Daten
# Diese m"ussen an die eigenen Bed"urfnisse angepa"st werden

# - Das externe Interface

EXTIP="10.0.0.2"
EXTIF="eth0"

# - Das interne Interface

INTIP="192.168.20.1"
INTMASK="255.255.255.0"
INTIF="eth1"
INTNET="$INTIP"/"$INTMASK"

# - Das DMZ-Netz

DMZIP="10.0.0.5"
DMZMASK="255.255.255.252"
DMZIF="eth2"
DMZNET="$DMZIP"/"$DMZMASK"

# - Der DNS-Server des Providers

DNSSERVER='cat /etc/resolv.conf | grep '^nameserver' | sed 's/nameserver/''
```

```
# - - TCP
TCPPROTECTED='3128 8000 8080 8118'

# - - UDP
UDPPROTECTED='515'

# - TCP-Ports des Servers, die offen sein m"ussen
DMZPORTS="80 443"

# - Art des FTP-Servers, m"ogliche Werte:
#
# NONE - keiner
# SRV - in der DMZ wird ein Server betrieben, keine Filterung
DMZFTP="SRV"

# - Datenport, den der FTP-Server in der DMZ benutzt. Falls Sie in der
# DMZ einen FTP-Proxy betreiben, kann es sein, da"s Sie diesen Wert
# anpassen m"ussen. Normale Server benutzen Port 20.
DMZFTPDATAPOINT="20"

# - Server in der DMZ, die nicht direkt zugreifbar sein sollten.
# Mehrfachnennungen sind m"oglich, z.B. "a.b.c.d w.x.y.z"
DMZIPPROT=""

# - Hohe Ports des Servers, die zus"atzlich gesch"utzt werden m"ussen
DMZUDPPROT=""
DMZTCPROT="8080 2049"

# -----
# Falls das Skript mit falschen Parametern aufgerufen wurde
# -----

case "$1" in
start)
    echo "Starte die Firewall ..."
    ;;
stop)
    echo "Beende die Vermittlung von Paketen ..."
    ;;
*)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac
```

```
# -----  
# Regeln, die immer gelten  
# -----  
  
# Alle Regeln und selbstdefinierten Chains l"oschen  
  
$R -F  
$R -X  
  
# Alle Pakete, die nicht explizit erlaubt sind, sind verboten  
  
$R -P input DENY  
$R -P forward DENY  
$R -P output DENY  
  
# Protokollierung gespoofter Pakete  
  
$R -A input -i ! "$INTIF" -s "$INTNET" -j DENY -l  
$R -A input -i "$INTIF" -s ! "$INTNET" -j DENY -l  
$R -A input -i ! lo -s 127.0.0.1 -j DENY -l  
$R -A input -i ! "$DMZIF" -s "$DMZNET" -j DENY -l  
$R -A input -i "$DMZIF" -s ! "$DMZNET" -j DENY -l  
  
# Lokale Pakete sind erlaubt  
  
$R -A input -i lo -j ACCEPT  
$R -A output -i lo -j ACCEPT  
  
# NetBIOS "uber TCP/IP  
  
$R -A input -p UDP -s "$INTNET" 137:139 -j DENY  
$R -A input -p UDP -s "$INTNET" --dport 137:139 -j DENY  
$R -A input -p TCP -s "$INTNET" 137:139 -j DENY  
$R -A input -p TCP -s "$INTNET" --dport 137:139 -j DENY  
  
case $1 in  
# -----  
# Die Firewall soll heruntergefahren werden  
# -----  
stop)  
  
# Protokollierung ungew"ohnlicher Pakete  
  
$R -A input -s 0.0.0.0/0 -j DENY -l  
$R -A output -s 0.0.0.0/0 -j DENY -l  
$R -A forward -s 0.0.0.0/0 -j DENY -l  
  
;;  
  
# -----  
# Die Firewall soll ihre Arbeit aufnehmen  
# -----  
start)
```

```
# ICMP

$I -A input -p icmp --sport 0 -j ACCEPT
$I -A input -p icmp --sport 3 -j ACCEPT
$I -A input -p icmp --sport 8 -j ACCEPT
$I -A input -p icmp --sport 11 -j ACCEPT
$I -A input -p icmp --sport 12 -j ACCEPT
$I -A output -p icmp -j ACCEPT

# Eigene Chains

# - Externes Interface

$I -N ext-in
$I -N ext-out

# - Internes Interface

$I -N int-in
$I -N int-fw
$I -N int-out

# - DMZ

$I -N ext2dmz
$I -N dmz2ext

# - Verteilung der Pakete auf die Chains

$I -A input -i "$DMZIF" -d "$DMZIP" -j ext-in
$I -A input -i "$DMZIF" -d "$EXTIF" -j ext-in
$I -A input -i "$DMZIF" -d "$INTNET" -l -j DENY
$I -A forward -i "$DMZIF" -s "$INTNET" -j int-fw
$I -A output -i "$DMZIF" -s "$DMZIP" -j ext-out

$I -A input -i "$DMZIF" -d !"$DMZIP" -j ACCEPT
$I -A input -i "$EXTIF" -d "$DMZNET" -j ACCEPT
$I -A forward -i "$DMZIF" -d "$DMZNET" -j ext2dmz
$I -A forward -i "$EXTIF" -s "$DMZNET" -j dmz2ext
$I -A output -i "$DMZIF" -s !"$DMZIP" -j ACCEPT
$I -A output -i "$EXTIF" -s "$DMZNET" -j ACCEPT

$I -A input -i "$INTIF" -j int-in
$I -A input -i "$EXTIF" -j ext-in
$I -A forward -i "$EXTIF" -s "$INTNET" -j int-fw
$I -A output -i "$INTIF" -j int-out
$I -A output -i "$EXTIF" -j ext-out

# Zugriffe auf Server der Firewall

# - TCP

for port in $TCPPROTECTED
do
    $I -A ext-in -p tcp --dport $port -j DENY -l
done
```

```
# - UDP

for port in $UDPPROTECTED
do
    $R -A ext-in -p udp --dport $port -j DENY -l
done

# DNS

# - Alle eingetragenen Server freischalten

for DNS in $DNSERVER
do

# - Zugriff auf den externen Server

    $R -A ext-in -p udp -s "$DNS" 53 --dport 1024:65535 -j ACCEPT
    $R -A ext-out -p udp -d "$DNS" 53 --sport 1024:65535 -j ACCEPT

    $R -A ext-in -p tcp -s "$DNS" 53 --dport 1024:65535 ! -y -j ACCEPT
    $R -A ext-out -p tcp -d "$DNS" 53 --sport 1024:65535 -j ACCEPT

# - Masquerade durch die Firewall

    $R -A int-in -p udp --sport 1024:65535 -d "$DNS" 53 -j ACCEPT
    $R -A int-fw -p udp -d "$DNS" 53 -j MASQ
    $R -A int-out -p udp -s "$DNS" 53 -j ACCEPT

    $R -A int-in -p tcp --sport 1024:65535 -d "$DNS" 53 -j ACCEPT
    $R -A int-fw -p tcp -d "$DNS" 53 -j MASQ
    $R -A int-out -p tcp -s "$DNS" 53 ! -y -j ACCEPT

done

# - Server auf der Firewall

# $R -A int-in -p udp -d "$INTIP" 53 -j ACCEPT
# $R -A int-out -p udp -s "$INTIP" 53 -j ACCEPT

# $R -A int-in -p tcp -d "$INTIP" 53 -j ACCEPT
# $R -A int-out -p tcp -s "$INTIP" 53 ! -y -j ACCEPT

# - Transparente Umleitung

# $R -A int-in -p UDP --sport 1024:65535 --dport 53 -j REDIRECT 53
# $R -A int-out -p UDP --sport 53 --dport 1024:65535 -j ACCEPT

# $R -A int-in -p TCP --sport 1024:65535 --dport 53 -j REDIRECT 53
# $R -A int-out -p TCP --sport 53 --dport 1024:65535 ! -y -j ACCEPT

# Ident

$R -A ext-in -p tcp --dport 113 -j REJECT -l
```

```
# HTTP

# - Zugriff auf den Server

$R -A ext-in -p tcp --dport 1024:65535 --sport 80 ! -y -j ACCEPT
$R -A ext-out -p tcp --sport 1024:65535 --dport 80 -j ACCEPT

# - Masquerade durch die Firewall

$R -A int-in -p tcp --sport 1024:65535 --dport 80 -j ACCEPT
$R -A int-fw -p tcp --sport 1024:65535 --dport 80 -j MASQ
$R -A int-out -p tcp --dport 1024:65535 --sport 80 ! -y -j ACCEPT

# - Proxies
# - - Proxy auf Port 3128 (squid)

# $R -A int-in -p TCP --sport 1024:65535 -d "$INTIP" 3128 -j ACCEPT
# $R -A int-out -p TCP --dport 1024:65535 -s "$INTIP" 3128 ! -y -j ACCEPT

# - - Proxy auf Port 8000 (junkbuster)

# $R -A int-in -p TCP --sport 1024:65535 -d "$INTIP" 8000 -j ACCEPT
# $R -A int-out -p TCP --dport 1024:65535 -s "$INTIP" 8000 ! -y -j ACCEPT

# - - Proxy auf Port 8080 (http-gw)

# $R -A int-in -p TCP --sport 1024:65535 -d "$INTIP" 8080 -j ACCEPT
# $R -A int-out -p TCP --dport 1024:65535 -s "$INTIP" 8080 ! -y -j ACCEPT

# - - Proxy auf Port 8118 (privoxy)

# $R -A int-in -p TCP --sport 1024:65535 -d "$INTIP" 8080 -j ACCEPT
# $R -A int-out -p TCP --dport 1024:65535 -s "$INTIP" 8080 ! -y -j ACCEPT

# - Transparente Umleitungen
# - - Transparente Umleitung auf Port 3128 (squid)

# $R -A int-in -p TCP --sport 1024:65535 --dport 80 -j REDIRECT 3128
# $R -A int-out -p TCP --sport 80 --dport 1024:65535 ! -y -j ACCEPT

# - - Transparente Umleitung auf Port 8118 (privoxy)

# $R -A int-in -p TCP --sport 1024:65535 --dport 80 -j REDIRECT 8118
# $R -A int-out -p TCP --sport 80 --dport 1024:65535 ! -y -j ACCEPT

# HTTPS

# - Zugriff auf den Server

$R -A ext-in -p tcp --dport 1024:65535 --sport 443 ! -y -j ACCEPT
$R -A ext-out -p tcp --sport 1024:65535 --dport 443 -j ACCEPT

# - Masquerade durch die Firewall

$R -A int-in -p tcp --sport 1024:65535 --dport 443 -j ACCEPT
$R -A int-fw -p tcp --sport 1024:65535 --dport 443 -j MASQ
$R -A int-out -p tcp --dport 1024:65535 --sport 443 ! -y -j ACCEPT
```

```
# SMTP
# - Zugriff auf den Server
$R -A ext-in -p tcp --dport 1024:65535 --sport 25 ! -y -j ACCEPT
$R -A ext-out -p tcp --sport 1024:65535 --dport 25 -j ACCEPT

# - Masquerade durch die Firewall
$R -A int-in -p tcp --sport 1024:65535 --dport 25 -j ACCEPT
$R -A int-fw -p tcp --sport 1024:65535 --dport 25 -j MASQ
$R -A int-out -p tcp --dport 1024:65535 --sport 25 ! -y -j ACCEPT

# POP3
# - Zugriff auf den Server
$R -A ext-in -p tcp --dport 1024:65535 --sport 110 ! -y -j ACCEPT
$R -A ext-out -p tcp --sport 1024:65535 --dport 110 -j ACCEPT

# - Masquerade durch die Firewall
$R -A int-in -p tcp --sport 1024:65535 --dport 110 -j ACCEPT
$R -A int-fw -p tcp --sport 1024:65535 --dport 110 -j MASQ
$R -A int-out -p tcp --dport 1024:65535 --sport 110 ! -y -j ACCEPT

# POP3S
# - Zugriff auf den Server
$R -A ext-in -p tcp --dport 1024:65535 --sport 995 ! -y -j ACCEPT
$R -A ext-out -p tcp --sport 1024:65535 --dport 995 -j ACCEPT

# - Masquerade durch die Firewall
$R -A int-in -p tcp --sport 1024:65535 --dport 995 -j ACCEPT
$R -A int-fw -p tcp --sport 1024:65535 --dport 995 -j MASQ
$R -A int-out -p tcp --dport 1024:65535 --sport 995 ! -y -j ACCEPT

# IMAP
# - Zugriff auf den Server
$R -A ext-in -p tcp --dport 1024:65535 --sport 143 ! -y -j ACCEPT
$R -A ext-out -p tcp --sport 1024:65535 --dport 143 -j ACCEPT

# - Masquerade durch die Firewall
$R -A int-in -p tcp --sport 1024:65535 --dport 143 -j ACCEPT
$R -A int-fw -p tcp --sport 1024:65535 --dport 143 -j MASQ
$R -A int-out -p tcp --dport 1024:65535 --sport 143 ! -y -j ACCEPT
```

```
# IMAPS
# - Zugriff auf den Server
$I -A ext-in -p tcp --dport 1024:65535 --sport 993 ! -y -j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 993 -j ACCEPT

# - Masquerade durch die Firewall
$I -A int-in -p tcp --sport 1024:65535 --dport 993 -j ACCEPT
$I -A int-fw -p tcp --sport 1024:65535 --dport 993 -j MASQ
$I -A int-out -p tcp --dport 1024:65535 --sport 993 ! -y -j ACCEPT

# NNTP
# - Zugriff auf den Server
$I -A ext-in -p tcp --dport 1024:65535 --sport 119 ! -y -j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 119 -j ACCEPT

# - Masquerade durch die Firewall
$I -A int-in -p tcp --sport 1024:65535 --dport 119 -j ACCEPT
$I -A int-fw -p tcp --sport 1024:65535 --dport 119 -j MASQ
$I -A int-out -p tcp --dport 1024:65535 --sport 119 ! -y -j ACCEPT

# FTP
# - Vorbereitung
/sbin/modprobe ip_masq_ftp

# - Zugriff auf den Server
# - - Kontrollverbindung
$I -A ext-in -p tcp --dport 1024:65535 --sport 21 ! -y -j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 21 -j ACCEPT

# - - Aktives FTP
$I -A ext-in -p tcp --dport 1024:65535 --sport 20 -j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 20 ! -y -j ACCEPT

# - - Passives FTP
$I -A ext-in -p tcp --dport 1024:65535 --sport 1024:65535 ! -y -j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 1024:65535 -j ACCEPT

# - Masquerade durch die Firewall
# - - Kontrollverbindung
$I -A int-in -p tcp --sport 1024:65535 --dport 21 -j ACCEPT
$I -A int-fw -p tcp --sport 1024:65535 --dport 21 -j MASQ
$I -A int-out -p tcp --dport 1024:65535 --sport 21 ! -y -j ACCEPT
```

```
# - - Aktives FTP
$R -A int-in -p tcp --sport 1024:65535 --dport 20 ! -y -j ACCEPT
$R -A int-fw -p tcp --sport 1024:65535 --dport 20 ! -y -j MASQ
$R -A int-out -p tcp --dport 1024:65535 --sport 20 -j ACCEPT

# - - Passives FTP
$R -A int-in -p tcp --sport 1024:65535 --dport 1024:65535 -j ACCEPT
$R -A int-fw -p tcp --sport 1024:65535 --dport 1024:65535 -j MASQ
$R -A int-out -p tcp --dport 1024:65535 --sport 1024:65535 ! -y -j ACCEPT

# - Proxy auf der Firewall
# - - Kontrollverbindung
# $R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" 21 -j ACCEPT
# $R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" 21 ! -y -j ACCEPT

# - - Aktives FTP
# - - - "Normaler Proxy"
# $R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" 20 ! -y -j ACCEPT
# $R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" 20 -j ACCEPT

# - - - Proxy im chroot-K"afig mit ungew"ohnlichem Datenport
# (SuSE ftp-proxy)
# $R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" 2020 ! -y -j ACCEPT
# $R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" 2020 -j ACCEPT

# - - Passives FTP
# $R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" 1024:65535 -j ACCEPT
# $R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" 1024:65535 ! -y -j ACCEPT

# - - Transparente Umleitung
# $R -A int-in -p TCP --sport 1024:65535 --dport 21 -j REDIRECT 21
# $R -A int-out -p TCP --sport 21 --dport 1024:65535 ! -y -j ACCEPT

# -----
# DMZ
# -----

# ICMP
$R -A ext2dmz -p icmp --icmp-type 0 -j ACCEPT
$R -A ext2dmz -p icmp --icmp-type 3 -j ACCEPT
$R -A ext2dmz -p icmp --icmp-type 8 -j ACCEPT
$R -A ext2dmz -p icmp --icmp-type 11 -j ACCEPT
$R -A ext2dmz -p icmp --icmp-type 12 -j ACCEPT
```

```
$R -A dmz2ext -p icmp --icmp-type 0 -j ACCEPT
$R -A dmz2ext -p icmp --icmp-type 3 -j ACCEPT
$R -A dmz2ext -p icmp --icmp-type 8 -j ACCEPT
$R -A dmz2ext -p icmp --icmp-type 11 -j ACCEPT
$R -A dmz2ext -p icmp --icmp-type 12 -j ACCEPT

# DNS-Anfragen der DMZ-Server

for DNS in $DNSSERVER
do
    $R -A dmz2ext -p udp -d "$DNS" 53 -j ACCEPT
    $R -A ext2dmz -p udp -s "$DNS" 53 -j ACCEPT

    $R -A dmz2ext -p tcp -d "$DNS" 53 -j ACCEPT
    $R -A ext2dmz -p tcp -s "$DNS" 53 ! -y -j ACCEPT
done

# Schutz von Servern, die nicht direkt zugreifbar sein sollen

for i in $DMZIPPROT
do
    $R -A ext2dmz -p tcp -d "$i" -l -j DENY
done

# Schutz von Diensten, die nur intern zugreifbar sein sollen

for p in $DMZUDPPROT
do
    $R -A ext2dmz -p udp --dport "$p" -l -j DENY
done

for p in $DMZTCPROT
do
    $R -A ext2dmz -p tcp --dport "$p" -l -j DENY
done

# Einfache TCP-Server

for p in $DMZPORTS
do
    $R -A dmz2ext -p tcp --sport "$p" --dport 1024:65535 -j ACCEPT
    $R -A ext2dmz -p tcp --dport "$p" -j ACCEPT
done

# FTP

case "$DMZFTP" in
SRV)

# - Server in der DMZ ohne Filterung
# -- Kontrollverbindung

    $R -A dmz2ext -p tcp --sport 21 --dport 1024:65535 -j ACCEPT
    $R -A ext2dmz -p tcp --dport 21 -j ACCEPT
```

```
# -- Datenverbindungen
# --- aktiv

    $R -A dmz2ext -p tcp --sport "$DMZFTPDATAPORT" --dport 1024:65535 \
    -j ACCEPT
    $R -A ext2dmz -p tcp --sport 1024:65535 \
    --dport 20 ! -y -j ACCEPT

# --- passiv

    $R -A ext2dmz -p tcp --sport 1024:65535 \
    --dport 1024:65535 -j ACCEPT
    $R -A dmz2ext -p tcp --sport 1024:65535 \
    --dport 1024:65535 ! -y -j ACCEPT
;;
esac

# -----
# Protokollierung ungew"ohnlicher Pakete
# -----

$R -A input -s 0.0.0.0/0 -j DENY -1
$R -A output -s 0.0.0.0/0 -j DENY -1
$R -A forward -s 0.0.0.0/0 -j DENY -1
$R -A int-in -s 0.0.0.0/0 -j DENY -1
$R -A int-out -s 0.0.0.0/0 -j DENY -1
$R -A int-fw -s 0.0.0.0/0 -j DENY -1
$R -A ext-in -s 0.0.0.0/0 -j DENY -1
$R -A ext-out -s 0.0.0.0/0 -j DENY -1
$R -A ext2dmz -s 0.0.0.0/0 -j DENY -1
$R -A dmz2ext -s 0.0.0.0/0 -j DENY -1

;;
esac
```

Paketfilterung mit iptables

Auch für die Paketfilterung mit iptables bietet es sich an, das bereits vorgestellte Skript `pfilter.iptables` (siehe Kapitel 12, Abschnitt *Regeln in Systemdateien eintragen*, ab Seite 336) zu nutzen und es um die nötigen Regeln zu erweitern, um auch die Vermittlung von Paketen zwischen DMZ und Internet bzw. lokalem Netz abzudecken.

Dazu müssen wir als erstes zusätzliche Variablen einfügen, die das Netzwerk-Interface beschreiben, an das die DMZ angeschlossen ist:

```
# - DMZ Interface

DMZIP="10.0.0.5"
DMZMASK="255.255.255.252"
DMZIF="eth2"
DMZNET="$DMZIP"/"$DMZMASK"
```

Nun definieren wir die Nummern der Ports, auf denen einfache TCP-basierte Server auf Anfragen warten. Für einen Webserver wäre das z. B. 80 (HTTP). Unterstützt Ihr Server auch HTTPS, so muß auch Port 443 freigeschaltet werden.

```
# - TCP-Ports des Servers, die offen sein müssen
```

```
DMZPORTS="80 443"
```

Für einen FTP-Server benötigen wir deutlich kompliziertere Regeln. Daher tragen wir für diesen Dienst nichts in *DMZPORTS* ein, sondern benutzen eine eigene Variable:

```
# - Art des FTP-Servers, mögliche Werte:
#
# NONE - keiner
# SRV - in der DMZ wird ein Server betrieben, keine Filterung
# PROXY - auf der Firewall läuft ein Proxy, ein direkter Zugriff in die
#         DMZ erfolgt nicht
```

```
DMZFTP="SRV"
```

Im Einzelfall kann es sein, daß der FTP-Server in der DMZ einen ungewöhnlichen Datenport benutzt. Dies ist insbesondere dann der Fall, wenn es sich nicht um den Server selbst, sondern um einen Proxy handelt, der in einer chroot-Umgebung läuft. In diesem Fall sollte man die folgende Variable anpassen:

```
# - Datenport, den der FTP-Server in der DMZ benutzt. Falls Sie in der
#   DMZ einen FTP-Proxy betreiben, kann es sein, daß Sie diesen Wert
#   anpassen müssen. Normale Server benutzen Port 20.
```

```
DMZFTPDATAPOINT="20"
```

Falls wir in der DMZ einen Proxy auf einem Rechner betreiben, den eigentlichen Server aber auf einem anderen, so wollen wir unter Umständen nicht, daß der eigentliche Server direkt aus dem Internet erreichbar ist. Mit der folgenden Variable können wir die Adressen solcher Rechner vorgeben, die dann explizit für die Weiterleitung gesperrt werden:

```
# - Server in der DMZ, die nicht zugreifbar sein sollen. Bitte geben Sie
#   mit Leerzeichen getrennte IP-Adressen an (z.B. ä.b.c.d w.x.y.z")
```

```
DMZIPPROT=""
```

Weiterhin kann es sein, daß aus irgendwelchen Gründen Dienste auf hohen Ports auf Anfragen warten, diese aber nicht aus dem Internet erreichbar sein sollen. Für diesen Fall führen wir zwei neue Variablen ein, in denen wir diese Ports eintragen können. Im folgenden Beispiel ist auf dem Server ein Dienst für das Network File System (NFS) installiert. Dies ist das Unix-Gegenstück zum Dateifreigabedienst unter Windows. Da er ähnlich unsicher ist, wie sein Windows-Pendant wollen wir auf keinen Fall, daß er aus dem Internet zugreifbar ist:

```
# - Hohe Ports des Servers, die zusätzlich geschützt werden müssen
```

```
DMZUDPPROT=""
DMZTCPPROT="2049"
```

Zu den Regeln, die ansprechen, wenn gefälschte Pakete empfangen werden, die als Absenderangabe eine Adresse der Firewall enthalten, müssen wir zusätzlich Regeln für das DMZ-Netzwerk-Interface eintragen:

```
# Alle Pakete, die nicht explizit erlaubt sind, sind verboten

[...]

$R -A INPUT -i ! "$DMZIF" -s "$DMZNET" -j LOG --log-level warning \
--log-prefix "$DMZIF gespooft: "
$R -A INPUT -i ! "$DMZIF" -s "$DMZNET" -j DROP

[...]

$R -A FORWARD -i ! "$DMZIF" -s "$DMZNET" -j LOG --log-level warning \
--log-prefix "$DMZIF gespooft: "
$R -A FORWARD -i ! "$DMZIF" -s "$DMZNET" -j DROP
```

Zusätzlich sollten wir aber auch verhindern, daß unsere eigenen Rechner gefälschte Pakete senden. War dies bisher kaum ein Problem, da Pakete aus dem lokalen Netz sowieso maskiert wurden, so hat es sich jetzt geändert, da wir einen unmaskierten Datenverkehr zwischen unserem DMZ-Server und dem Internet erlauben. Erschwerend kommt hinzu, daß das Risiko einer Kompromittierung unseres Servers deutlich höher ist als das eines erfolgreichen Angriffs auf unsere Klienten im lokalen Netz. Benutzt ihn ein Angreifer aber als Ausgangsbasis z. B. für Angriffe auf andere Rechner im Internet, so wollen wir ihm nicht auch noch helfen, seine Spuren zu verwischen:

```
$R -A INPUT -i "$INTIF" -s ! "$INTNET" -j LOG --log-level warning \
--log-prefix "Spoofer auf $INTIF: "
$R -A INPUT -i "$INTIF" -s ! "$INTNET" -j DROP

$R -A FORWARD -i "$INTIF" -s ! "$INTNET" -j LOG --log-level warning \
--log-prefix "Spoofer auf $INTIF: "
$R -A FORWARD -i "$INTIF" -s ! "$INTNET" -j DROP

$R -A INPUT -i "$DMZIF" -s ! "$DMZNET" -j LOG --log-level warning \
--log-prefix "Spoofer auf $DMZIF: "
$R -A INPUT -i "$DMZIF" -s ! "$DMZNET" -j DROP

$R -A FORWARD -i "$DMZIF" -s ! "$DMZNET" -j LOG --log-level warning \
--log-prefix "Spoofer auf $DMZIF: "
$R -A FORWARD -i "$DMZIF" -s ! "$DMZNET" -j DROP
```

Auch unsere zustandsbehafteten Filterregeln müssen wir überdenken. Bisher konnten wir alle Pakete entsorgen, die einen Verbindungsaufbau aus dem Internet bedeutet hätten. Nun müssen wir aber eine Regel einfügen, die für den Aufbau von Verbindungen in die DMZ eine Ausnahme macht. Genauso müssen wir Verbindungsaufbauten an Port 21 der Firewall erlauben, falls wir einen Reverse Proxy auf ihr betreiben.

```
# Unaufgeforderte Verbindungsaufbauten (NEW) und ungültige Pakete
# - Die Regeln
$I -A states -m state --state NEW -d "$DMZNET" -j RETURN

if [ "$DMZFTP"="PROXY" ]
then
    $I -A states -m state --state NEW -p tcp --dport 21 -j RETURN
fi

$I -A states -m state --state NEW,INVALID -j LOG \
--log-prefix Überwünschte Verbindung:"
```

Für die DMZ-spezifischen Regeln benötigen wir zwei eigene Chains:

```
# - DMZ
$I -N ext2dmz
$I -N dmz2ext
```

Das Verteilen der Pakete auf die einzelnen Chains ist relativ unkompliziert. Pakete zwischen DMZ und Internet werden in die neuen Chains weitergeleitet, während alle anderen Pakete aus der DMZ oder in die DMZ an die Chains weitergeleitet werden, die auch für den Datenverkehr zwischen lokalem Netz bzw. Prozessen der Firewall und dem Internet zuständig sind:

```
# - Verteilung der Pakete auf die Chains
$I -A INPUT -i "$INTIF" -s "$INTNET" -j int-in
$I -A INPUT -i "$EXTIF" -j ext-in
$I -A INPUT -i "$DMZIF" -j ext-in
$I -A FORWARD -i "$INTIF" -o "$EXTIF" -s "$INTNET" -j int-fw
$I -A FORWARD -i "$INTIF" -o "$DMZIF" -s "$INTNET" -j int-fw
$I -A FORWARD -i "$EXTIF" -o "$INTIF" -j ext-fw
$I -A FORWARD -i "$DMZIF" -o "$INTIF" -j ext-fw
$I -A FORWARD -i "$EXTIF" -o "$DMZIF" -j ext2dmz
$I -A FORWARD -i "$DMZIF" -o "$EXTIF" -j dmz2ext
$I -A OUTPUT -o "$INTIF" -j int-out
$I -A OUTPUT -o "$EXTIF" -j ext-out
$I -A OUTPUT -o "$DMZIF" -j ext-out
```

Die einzelnen Regeln für das lokale Netz ändern sich nicht. Wir brauchen aber zusätzliche Regeln für den Datenverkehr zwischen DMZ und Internet. Beginnen wir mit ICMP. Grundsätzlich ist es sinnvoll, auch dem DMZ-Server Fehlermeldungen zuzustellen. Außerdem sollte er mit ping angesprochen werden können. Eingangsseitig können wir daher die Regeln übernehmen, die auch für die Firewall selbst gelten.

Im Gegensatz zur Firewall sollten wir unserem DMZ-Server aber nicht erlauben, beliebige ICMP-Pakete zu senden. Einige dieser Pakete können für Angriffe auf Rechner im Internet verwendet werden. Um also im Falle einer Kompromittierung zu verhindern, daß einer unserer eigenen Rechner plötzlich zur Ausgangsbasis für Angriffe wird, sollten wir dem Server nur das Senden derjenigen Pakete erlauben, die er auch empfangen darf:

```
# ICMP

$I -A ext2dmz -p icmp --icmp-type 0 -j ACCEPT
$I -A ext2dmz -p icmp --icmp-type 3 -j ACCEPT
$I -A ext2dmz -p icmp --icmp-type 8 -j ACCEPT
$I -A ext2dmz -p icmp --icmp-type 11 -j ACCEPT
$I -A ext2dmz -p icmp --icmp-type 12 -j ACCEPT

$I -A dmz2ext -p icmp --icmp-type 0 -j ACCEPT
$I -A dmz2ext -p icmp --icmp-type 3 -j ACCEPT
$I -A dmz2ext -p icmp --icmp-type 8 -j ACCEPT
$I -A dmz2ext -p icmp --icmp-type 11 -j ACCEPT
$I -A dmz2ext -p icmp --icmp-type 12 -j ACCEPT
```

Der DMZ-Server sollte in der Lage sein, auch DNS-Anfragen zu stellen. Dies ist z. B. nötig, falls Zugriffe protokolliert werden. Grundsätzlich entsprechen die dafür nötigen Regeln denen, die auch benutzt werden, um der Firewall selbst das Stellen von DNS-Anfragen zu erlauben. Dabei müssen wir allerdings beachten, daß wir für die DMZ-Chains nicht grundsätzlich den Aufbau von Verbindungen ausfiltern. Wir sollten daher die Regeln etwas modifizieren, um zu verhindern, daß wir Port Scans hoher Ports von Port 53 aus erlauben:

```
# DNS-Anfragen der DMZ-Server

for d in $DNSSERVER
do
    $I -A dmz2ext -p udp -d "$d" --dport 53 -j ACCEPT
    $I -A ext2dmz -p udp -m state --state ESTABLISHED \
    -s "$d" --sport 53 -j ACCEPT

    $I -A dmz2ext -p tcp -d "$d" --dport 53 -j ACCEPT
    $I -A ext2dmz -p tcp -m state --state ESTABLISHED \
    -s "$d" --sport 53 ! --syn -j ACCEPT
done
```

Wir sollten auch Regeln definieren, um die als besonders schutzbedürftig eingestuften Server und Dienste für Zugriffe zu sperren:

```
# Schutz von Servern vor direkten Zugriffen aus dem Internet

for i in $DMZIPPROT
do
    $I -A ext2dmz -p tcp -d "$i" -j LOG --log-level notice \
    --log-prefix "ext2dmz (ip): "
    $I -A ext2dmz -p tcp -d "$i" -j DROP
done

# Schutz von Diensten, die nur intern zugreifbar sein sollen

for p in $DMZUDPPROT
do
    $I -A ext2dmz -p udp --dport "$p" -j LOG --log-level notice \
    --log-prefix "ext2dmz (udp): "
    $I -A ext2dmz -p udp --dport "$p" -j DROP
done
```

```

for p in $DMZTCPROT
do
  $R -A ext2dmz -p tcp --dport "$p" -j LOG --log-level notice \
  --log-prefix "ext2dmz (tcp): "
  $R -A ext2dmz -p tcp --dport "$p" -j DROP
done

```

Nachdem das geschehen ist, können wir Zugriffe auf die normalen Serverports erlauben:

```

# Einfache TCP-Server

for p in $DMZPORTS
do
  $R -A dmz2ext -p tcp --sport "$p" --dport 1024:65535 -j ACCEPT
  $R -A ext2dmz -p tcp --dport "$p" -j ACCEPT
done

```

Etwas komplizierter sind die FTP-Verbindungen. Insbesondere müssen wir hier sicherstellen, daß eingehende Verbindungen nur dann zugelassen werden, wenn es sich um Datenverbindungen handelt, die vorher über eine Kontrollverbindung explizit angefordert wurden. Unterlassen wir dies, so sind beliebige Verbindungen zwischen Rechnern im Internet und dem DMZ-Server möglich, vorausgesetzt, auf beiden Seiten wird eine Portnummer verwendet, die über 1023 liegt:

```

# FTP

case "$DMZFTP" in
SRV)
# - Server in der DMZ ohne Filterung
# -- Kontrollverbindung

  $R -A dmz2ext -p tcp --sport 21 --dport 1024:65535 -j ACCEPT
  $R -A ext2dmz -p tcp --dport 21 -j ACCEPT

# -- Datenverbindungen
# --- aktiv

  $R -A dmz2ext -m state --state RELATED -p tcp \
  --sport "$DMZFTPDATAPOINT" --dport 1024:65535 -j ACCEPT

  $R -A dmz2ext -m state --state ESTABLISHED -p tcp \
  --sport "$DMZFTPDATAPOINT" --dport 1024:65535 -j ACCEPT

  $R -A ext2dmz -m state --state ESTABLISHED -p tcp \
  --sport 1024:65535 --dport "$DMZFTPDATAPOINT" -j ACCEPT

# --- passiv

  $R -A ext2dmz -m state --state RELATED -p tcp \
  --sport 1024:65535 --dport 1024:65535 -j ACCEPT

  $R -A ext2dmz -m state --state ESTABLISHED -p tcp \
  --sport 1024:65535 --dport 1024:65535 -j ACCEPT

  $R -A dmz2ext -m state --state ESTABLISHED -p tcp \
  --sport 1024:65535 --dport 1024:65535 -j ACCEPT

;;

```

```
PROXY)

# - Reverse Proxy auf der Firewall
# -- Kontrollverbindung

$I_R -A ext-in -p tcp --sport 1024:65535 --dport 21 \
-j ACCEPT
$I_R -A ext-out -p tcp --dport 1024:65535 --sport 21 \
! --syn -j ACCEPT

$I_R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" \
--dport 21 -j ACCEPT
$I_R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" \
--sport 21 ! --syn -j ACCEPT

# -- Datenverbindung, aktiv

$I_R -A ext-in -p tcp --sport 1024:65535 --dport 2020 \
! --syn -j ACCEPT
$I_R -A ext-out -p tcp --dport 1024:65535 --sport 2020 \
-j ACCEPT

$I_R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" \
--dport 2020 ! --syn -j ACCEPT
$I_R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" \
--sport 2020 -j ACCEPT

# -- Datenverbindung, passiv

$I_R -A ext-in -p tcp --sport 1024:65535 --dport 1024:65535 \
-j ACCEPT
$I_R -A ext-out -p tcp --dport 1024:65535 --sport 1024:65535 \
! --syn -j ACCEPT

$I_R -A int-in -p tcp --sport 1024:65535 --dport 1024:65535 \
-d "$INTIP" -j ACCEPT
$I_R -A int-out -p tcp --dport 1024:65535 --sport 1024:65535 \
-s "$INTIP" ! --syn -j ACCEPT

# -- Transparente Umleitung auf die Firewall

$I_R -t nat -A PREROUTING -i "$EXTIF" -p tcp --dport 21 \
-j REDIRECT --to-port 21

$I_R -t nat -A PREROUTING -i "$INTIF" -p tcp -d "$DMZNET" \
--dport 21 -j REDIRECT --to-port 21

;;
esac
```

Daß hier im Fall des Proxys auf der Firewall kein Gebrauch von Stateful Packet Filtering gemacht wird, liegt daran, daß bereits zu Beginn des Skriptes entsprechende Regeln definiert wurden.

Wie in allen anderen Chains, so müssen wir auch in den DMZ-Chains Regeln einfügen, um all die Pakete zu protokollieren und zu entsorgen, die nicht explizit durch bereits definierte Regeln erlaubt wurden:

```
# -----
# Protokollierung ungewöhnlicher Pakete
# -----

$I -A ext2dmz -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "ext2dmz (default): "
$I -A dmz2ext -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "dmz2ext (default): "

$I -A dmz2ext -s 0.0.0.0/0 -j DROP
$I -A ext2dmz -s 0.0.0.0/0 -j DROP
```

Schließlich müssen wir noch das Masquerading geringfügig anpassen. Es ist besser, alle Pakete der Rechner im lokalen Netz zu maskieren, nicht nur solche, die für Rechner im Internet bestimmt sind. Auch der DMZ-Server ist nicht vertrauenswürdig genug, als daß wir ihm erlauben sollten, direkt mit den Rechnern im LAN zu kommunizieren:

```
# Network Address Translation
# - Masquerading

$I -t nat -A POSTROUTING -o "$EXTIF" -j MASQUERADE
$I -t nat -A POSTROUTING -s "$INTNET" -o "$EXTIF" -j MASQUERADE
$I -t nat -A POSTROUTING -s "$INTNET" -o "$DMZIF" -j MASQUERADE
```

Wenn wir dies alles zusammensetzen, so erhalten wir ein Skript, wie es im folgenden zu sehen ist. Ich habe an dieser Stelle noch Code eingefügt, um festzustellen, ob der Befehl iptables sich in */sbin/* oder */usr/sbin/* befindet, da dies je nach Distribution unterschiedlich ist. Wird der Befehl in beiden Verzeichnissen gefunden, so bricht das Skript mit einer Fehlermeldung ab.

Denken Sie bitte daran, die Netzwerkeinstellungen Ihrer konkreten Situation anzupassen und gegebenenfalls die Kommentarzeichen vor den Regeln für Proxies, einen eigenen DNS-Server oder die transparente Umleitung auf einen lokalen Proxy zu entfernen:

```
#!/bin/sh
#####
#
# dmz.iptables
#
#   Filterregeln f"ur ein lokales Netz und eine DMZ (iptables-Version)
#
# Usage: dmz {start|stop}
#
# Copyright (C) 2003 Andreas G. Lessig
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
```

```
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
#####

### BEGIN INIT INFO
# Provides:          dmz-ipt
# Required-Start:    $local_fs
# Should-Start:
# Required-Stop:     proccnf
# Should-Stop:
# Default-Start:     2 3 5
# Default-Stop:      0 1 6
# Short-Description: Firewalling f"ur ein LAN und eine DMZ
# Description:       Firewalling f"ur ein LAN und eine DMZ
### END INIT INFO

# -----
# Grundeinstellungen
# -----

# Warnfarbe

C_RED='\033[1m\033[31m'
C_RESET='\033[m'

# Eine Kurzbezeichnung f"ur iptables
# (iptables mag es, sich zu verstecken)

if test -x /sbin/iptables
then
    R=/sbin/iptables
    if test -x /usr/sbin/iptables
    then
        echo -en "${C_RED}${0}: ERROR: Es gibt 2 Programme iptables. "
        echo -e "Breche ab! ${C_RESET}"
        exit 1
    fi
else
    if test -x /usr/sbin/iptables
    then
        R=/usr/sbin/iptables
    else
        echo -en "${C_RED}${0}: ERROR: iptables nicht gefunden. "
        echo -e "Breche ab! ${C_RESET}"
        exit 1
    fi
fi

# Ein paar grunds"atzliche Daten

# - Externes Interface

EXTIF="eth0"
```

```
# - Internes Interface
INTIP="192.168.20.1"
INTMASK="255.255.255.0"
INTIF="eth1"
INTNET="$INTIP"/"$INTMASK"

# - DMZ-Interface
DMZIP="10.0.0.5"
DMZMASK="255.255.255.252"
DMZIF="eth2"
DMZNET="$DMZIP"/"$DMZMASK"

# - DNS-Server
DNSSERVER='cat /etc/resolv.conf | grep '^nameserver' | sed 's/nameserver//''

# - Hohe Ports, die aus dem Internet nicht zugreifbar sein sollen

# - - TCP
TCPPROTECTED='3128 8000 8080 8118'

# - - UDP
UDPPROTECTED='515'

# Einstellungen f"ur den Betrieb eines Servers in der DMZ

# - TCP-Ports des Servers, die offen sein m"ussen
DMZPORTS="80 443"

# - Art des FTP-Servers, m"ogliche Werte:
#
# NONE - keiner
# SRV - in der DMZ wird ein Server betrieben, keine Filterung
# PROXY - auf der Firewall l"auft ein Proxy, ein direkter Zugriff in die
#         DMZ erfolgt nicht
DMZFTP="SRV"

# - Datenport, den der FTP-Server in der DMZ benutzt. Falls Sie in der
#   DMZ einen FTP-Proxy betreiben, kann es sein, da"s Sie diesen Wert
#   anpassen m"ussen. Normale Server benutzen Port 20.
DMZFTPDATAPORT="20"

# - Server in der DMZ, die nicht zugreifbar sein sollen. Bitte geben Sie
#   mit Leerzeichen getrennte IP-Adressen an (z.B. "a.b.c.d w.x.y.z")
DMZIPPROT=""

# - Hohe Ports des Servers, die zus"atzlich gesch"utzt werden m"ussen
DMZUDPPROT=""
DMZTCPROT="2049"
```

```
# -----  
# Falls das Skript mit falschen Parametern aufgerufen wurde  
# -----  
  
case "$1" in  
start)  
    echo "Starte die Firewall ..."  
    ;;  
stop)  
    echo "Beende die Vermittlung von Paketen ..."  
    ;;  
*)  
    echo "Usage: $0 {start|stop}"  
    exit 1  
    ;;  
esac  
  
# -----  
# Regeln, die immer gelten  
# -----  
  
# Alle Regeln und selbstdefinierten Chains l"oschen  
  
$R -F  
$R -X  
$R -t nat -F  
$R -t nat -X  
  
# Alle Pakete, die nicht explizit erlaubt sind, sind verboten  
  
$R -P INPUT DROP  
$R -P FORWARD DROP  
$R -P OUTPUT DROP  
  
# Im Table nat werden keine Pakete verworfen, das geschieht im Table filter  
  
$R -t nat -P PREROUTING ACCEPT  
$R -t nat -P POSTROUTING ACCEPT  
$R -t nat -P OUTPUT ACCEPT  
  
# Protokollierung gespoofter Pakete  
  
$R -A INPUT -i ! "$INTIF" -s "$INTNET" -j LOG --log-level warning \  
--log-prefix "$INTIF gespooft: "  
$R -A INPUT -i ! "$INTIF" -s "$INTNET" -j DROP  
$R -A INPUT -i ! lo -s 127.0.0.1 -j LOG --log-level warning \  
--log-prefix "loopback gespooft: "  
$R -A INPUT -i ! lo -s 127.0.0.1 -j DROP  
$R -A INPUT -i ! "$DMZIF" -s "$DMZNET" -j LOG --log-level warning \  
--log-prefix "$DMZIF gespooft: "  
$R -A INPUT -i ! "$DMZIF" -s "$DMZNET" -j DROP
```

```
$R -A FORWARD -i ! "$INTIF" -s "$INTNET" -j LOG --log-level warning \  
--log-prefix "$INTIF gespooft: "  
$R -A FORWARD -i ! "$INTIF" -s "$INTNET" -j DROP  
$R -A FORWARD -i ! lo -s 127.0.0.1 -j LOG --log-level warning \  
--log-prefix "loopback gespooft: "  
$R -A FORWARD -i ! lo -s 127.0.0.1 -j DROP  
$R -A FORWARD -i ! "$DMZIF" -s "$DMZNET" -j LOG --log-level warning \  
--log-prefix "$DMZIF gespooft: "  
$R -A FORWARD -i ! "$DMZIF" -s "$DMZNET" -j DROP  
  
$R -A INPUT -i "$INTIF" -s ! "$INTNET" -j LOG --log-level warning \  
--log-prefix "Spoofer auf $INTIF: "  
$R -A INPUT -i "$INTIF" -s ! "$INTNET" -j DROP  
  
$R -A FORWARD -i "$INTIF" -s ! "$INTNET" -j LOG --log-level warning \  
--log-prefix "Spoofer auf $INTIF: "  
$R -A FORWARD -i "$INTIF" -s ! "$INTNET" -j DROP  
  
$R -A INPUT -i "$DMZIF" -s ! "$DMZNET" -j LOG --log-level warning \  
--log-prefix "Spoofer auf $DMZIF: "  
$R -A INPUT -i "$DMZIF" -s ! "$DMZNET" -j DROP  
  
$R -A FORWARD -i "$DMZIF" -s ! "$DMZNET" -j LOG --log-level warning \  
--log-prefix "Spoofer auf $DMZIF: "  
$R -A FORWARD -i "$DMZIF" -s ! "$DMZNET" -j DROP  
  
# Ident  
  
#$R -A INPUT -i "$EXTIF" -p tcp --dport 113 -j LOG --log-level info \  
#--log-prefix "ident probe: "  
$R -A INPUT -i "$EXTIF" -p tcp --dport 113 -j REJECT  
  
#$R -A FORWARD -i "$EXTIF" -p tcp --dport 113 -j LOG --log-level info \  
#--log-prefix "ident probe: "  
$R -A FORWARD -i "$EXTIF" -p tcp --dport 113 -j REJECT  
  
# Unaufgeforderte Verbindungsaufbauten (NEW) und ung"ultige Pakete  
# (INVALID) des externen Interfaces.  
  
# - Eine eigene Chain f"ur diese Tests  
  
$R -N states  
$R -F states  
  
# - Dorthin verzweigen  
  
$R -A INPUT -i "$EXTIF" -j states  
$R -A FORWARD -i "$EXTIF" -j states  
  
# - Die Regeln  
  
# -- Zugriffe in die DMZ  
  
$R -A states -m state --state NEW -d "$DMZNET" -j RETURN
```

```
# -- FTP-Proxy auf der Firewall
if [ "$DMZFTP"="PROXY" ]
then
    $R -A states -m state --state NEW -p tcp --dport 21 -j RETURN
fi

# -- Sonstige Verbindungsaufbauten

$R -A states -m state --state NEW,INVALID -j LOG \
--log-prefix "Unerwunschte Verbindung:"

$R -A states -m state --state NEW,INVALID -j DROP

# - Ruecksprung, falls noch nicht verworfen

# $R -A states -j RETURN

# Lokale Pakete sind erlaubt

$R -A INPUT -i lo -j ACCEPT
$R -A OUTPUT -o lo -j ACCEPT

# NetBIOS "uber TCP/IP

$R -A INPUT -p UDP -s "$INTNET" --sport 137:139 -j DROP
$R -A INPUT -p UDP -s "$INTNET" --dport 137:139 -j DROP
$R -A INPUT -p TCP -s "$INTNET" --sport 137:139 -j DROP
$R -A INPUT -p TCP -s "$INTNET" --dport 137:139 -j DROP

$R -A FORWARD -p UDP -s "$INTNET" --sport 137:139 -j DROP
$R -A FORWARD -p UDP -s "$INTNET" --dport 137:139 -j DROP
$R -A FORWARD -p TCP -s "$INTNET" --sport 137:139 -j DROP
$R -A FORWARD -p TCP -s "$INTNET" --dport 137:139 -j DROP

case $1 in
# -----
# Die Firewall soll heruntergefahren werden
# -----
stop)

# Protokollierung ungew"ohnlicher Pakete

$R -A INPUT -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "INPUT (default): "
$R -A INPUT -s 0.0.0.0/0 -j DROP
$R -A OUTPUT -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "OUTPUT (default): "
$R -A OUTPUT -s 0.0.0.0/0 -j DROP
$R -A FORWARD -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "FORWARD (default): "
$R -A FORWARD -s 0.0.0.0/0 -j DROP

;;
```

```
# -----  
# Die Firewall soll ihre Arbeit aufnehmen  
# -----  
start)  
  
# -----  
# Allgemeine Regeln  
# -----  
  
# ICMP  
  
$R -A INPUT -p icmp --icmp-type 0 -j ACCEPT  
$R -A INPUT -p icmp --icmp-type 3 -j ACCEPT  
$R -A INPUT -p icmp --icmp-type 8 -j ACCEPT  
$R -A INPUT -p icmp --icmp-type 11 -j ACCEPT  
$R -A INPUT -p icmp --icmp-type 12 -j ACCEPT  
$R -A OUTPUT -p icmp -j ACCEPT  
  
$R -I states -p icmp --icmp-type 8 -j RETURN  
  
# Eigene Chains  
  
# - Externes Interface  
  
$R -N ext-in  
$R -N ext-fw  
$R -N ext-out  
  
# - Internes Interface  
  
$R -N int-in  
$R -N int-fw  
$R -N int-out  
  
# - DMZ  
  
$R -N ext2dmz  
$R -N dmz2ext  
  
# - Verteilung der Pakete auf die Chains  
  
$R -A INPUT -i "$INTIF" -s "$INTNET" -j int-in  
$R -A INPUT -i "$EXTIF" -j ext-in  
$R -A INPUT -i "$DMZIF" -j ext-in  
$R -A FORWARD -i "$INTIF" -o "$EXTIF" -s "$INTNET" -j int-fw  
$R -A FORWARD -i "$INTIF" -o "$DMZIF" -s "$INTNET" -j int-fw  
$R -A FORWARD -i "$EXTIF" -o "$INTIF" -j ext-fw  
$R -A FORWARD -i "$DMZIF" -o "$INTIF" -j ext-fw  
$R -A FORWARD -i "$EXTIF" -o "$DMZIF" -j ext2dmz  
$R -A FORWARD -i "$DMZIF" -o "$EXTIF" -j dmz2ext  
$R -A OUTPUT -o "$INTIF" -j int-out  
$R -A OUTPUT -o "$EXTIF" -j ext-out  
$R -A OUTPUT -o "$DMZIF" -j ext-out  
  
# -----  
# Lokales Netz  
# -----
```

```
# Zugriffe auf Server der Firewall

# - TCP

for port in $TCPPROTECTED
do
  $R -A ext-in -p tcp --dport $port -j LOG \
  --log-prefix "Zugriff auf Port $port TCP"
  $R -A ext-in -p tcp --dport $port -j DROP
done

# - UDP

for port in $UDPPROTECTED
do
  $R -A ext-in -p udp --dport $port -j LOG \
  --log-prefix "Zugriff auf Port $port UDP"
  $R -A ext-in -p udp --dport $port -j DROP
done

# DNS

# - Alle eingetragenen Server freischalten

for DNS in $DNSSERVER
do

# - - Zugriff auf den externen Server

  $R -A ext-in -p udp -s "$DNS" --sport 53 --dport 1024:65535 \
  -j ACCEPT
  $R -A ext-out -p udp -d "$DNS" --dport 53 --sport 1024:65535 \
  -j ACCEPT
  $R -A ext-in -p tcp -s "$DNS" --sport 53 --dport 1024:65535 \
  ! --syn -j ACCEPT
  $R -A ext-out -p tcp -d "$DNS" --dport 53 --sport 1024:65535 \
  -j ACCEPT

# - - Forwarding durch die Firewall

  $R -A int-fw -p udp -d "$DNS" --dport 53 -j ACCEPT
  $R -A ext-fw -p udp -s "$DNS" --sport 53 -j ACCEPT

  $R -A int-fw -p tcp -d "$DNS" --dport 53 -j ACCEPT
  $R -A ext-fw -p tcp -s "$DNS" --sport 53 -j ACCEPT

done

# - Server auf der Firewall

# $R -A int-in -p udp -d "$INTIP" --dport 53 -j ACCEPT
# $R -A int-out -p udp -s "$INTIP" --sport 53 -j ACCEPT
# $R -A int-in -p tcp -d "$INTIP" --dport 53 -j ACCEPT
# $R -A int-out -p tcp -s "$INTIP" --sport 53 ! --syn -j ACCEPT
```

```
# HTTP
# - Zugriff auf den Server

$I_R -A ext-in -p tcp --dport 1024:65535 --sport 80 ! --syn \
-j ACCEPT
$I_R -A ext-out -p tcp --sport 1024:65535 --dport 80 -j ACCEPT

# - Forwarding durch die Firewall

$I_R -A int-fw -p tcp --sport 1024:65535 --dport 80 -j ACCEPT
$I_R -A ext-fw -p tcp --dport 1024:65535 --sport 80 -j ACCEPT

# HTTP-Proxies

# - Zugriff auf den Squid
$I_R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" --dport 3128 \
-j ACCEPT
$I_R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" --sport 3128 \
#! --syn -j ACCEPT

# - Zugriff auf den Junkbuster
$I_R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" --dport 8000 \
-j ACCEPT
$I_R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" --sport 8000 \
#! --syn -j ACCEPT

# - Zugriff auf den Privoxy
$I_R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" --dport 8118 \
-j ACCEPT
$I_R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" --sport 8118 \
#! --syn -j ACCEPT

# - Zugriff auf den httpgw
$I_R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" --dport 8080 \
-j ACCEPT
$I_R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" --sport 8080 \
#! --syn -j ACCEPT

# HTTPS
# - Zugriff auf den Server

$I_R -A ext-in -p tcp --dport 1024:65535 --sport 443 ! --syn \
-j ACCEPT
$I_R -A ext-out -p tcp --sport 1024:65535 --dport 443 -j ACCEPT

# - Forwarding durch die Firewall

$I_R -A int-fw -p tcp --sport 1024:65535 --dport 443 -j ACCEPT
$I_R -A ext-fw -p tcp --dport 1024:65535 --sport 443 -j ACCEPT

# SMTP
# - Zugriff auf den Server

$I_R -A ext-in -p tcp --dport 1024:65535 --sport 25 ! --syn \
-j ACCEPT
$I_R -A ext-out -p tcp --sport 1024:65535 --dport 25 -j ACCEPT
```

```
# - Forwarding durch die Firewall
$R -A int-fw -p tcp --sport 1024:65535 --dport 25 -j ACCEPT
$R -A ext-fw -p tcp --dport 1024:65535 --sport 25 -j ACCEPT

# POP3
# - Zugriff auf den Server
$R -A ext-in -p tcp --dport 1024:65535 --sport 110 ! --syn \
-j ACCEPT
$R -A ext-out -p tcp --sport 1024:65535 --dport 110 -j ACCEPT

# - Forwarding durch die Firewall
$R -A int-fw -p tcp --sport 1024:65535 --dport 110 -j ACCEPT
$R -A ext-fw -p tcp --dport 1024:65535 --sport 110 -j ACCEPT

# POP3S
# - Zugriff auf den Server
$R -A ext-in -p tcp --dport 1024:65535 --sport 995 ! --syn \
-j ACCEPT
$R -A ext-out -p tcp --sport 1024:65535 --dport 995 -j ACCEPT

# - Forwarding durch die Firewall
$R -A int-fw -p tcp --sport 1024:65535 --dport 995 -j ACCEPT
$R -A ext-fw -p tcp --dport 1024:65535 --sport 995 -j ACCEPT

# IMAP
# - Zugriff auf den Server
$R -A ext-in -p tcp --dport 1024:65535 --sport 143 ! --syn \
-j ACCEPT
$R -A ext-out -p tcp --sport 1024:65535 --dport 143 -j ACCEPT

# - Forwarding durch die Firewall
$R -A int-fw -p tcp --sport 1024:65535 --dport 143 -j ACCEPT
$R -A ext-fw -p tcp --dport 1024:65535 --sport 143 -j ACCEPT

# IMAPS
# - Zugriff auf den Server
$R -A ext-in -p tcp --dport 1024:65535 --sport 993 ! --syn \
-j ACCEPT
$R -A ext-out -p tcp --sport 1024:65535 --dport 993 -j ACCEPT

# - Forwarding durch die Firewall
$R -A int-fw -p tcp --sport 1024:65535 --dport 993 -j ACCEPT
$R -A ext-fw -p tcp --dport 1024:65535 --sport 993 -j ACCEPT
```

```
# NNTP
# - Zugriff auf den Server

$I -A ext-in -p tcp --dport 1024:65535 --sport 119 ! --syn \
-j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 119 -j ACCEPT

# - Forwarding durch die Firewall

$I -A int-fw -p tcp --sport 1024:65535 --dport 119 -j ACCEPT
$I -A ext-fw -p tcp --dport 1024:65535 --sport 119 -j ACCEPT

# Gopher
# - Zugriff auf den Server

$I -A ext-in -p tcp --dport 1024:65535 --sport 70 ! --syn \
-j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 70 -j ACCEPT

# - Forwarding durch die Firewall

$I -A int-fw -p tcp --sport 1024:65535 --dport 70 -j ACCEPT
$I -A ext-fw -p tcp --dport 1024:65535 --sport 70 -j ACCEPT

# WAIS
# - Zugriff auf den Server

$I -A ext-in -p tcp --dport 1024:65535 --sport 210 ! --syn \
-j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 210 -j ACCEPT

# - Forwarding durch die Firewall

$I -A int-fw -p tcp --sport 1024:65535 --dport 210 -j ACCEPT
$I -A ext-fw -p tcp --dport 1024:65535 --sport 210 -j ACCEPT

# FTP
# - Zugriff auf den Server -----
# - - Kontrollverbindung

$I -A ext-in -p tcp --dport 1024:65535 --sport 21 ! --syn \
-j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 21 -j ACCEPT

# - - Aktives FTP

$I -A ext-in -p tcp --dport 1024:65535 --sport 20 -j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 20 ! --syn \
-j ACCEPT

# - - Passives FTP

$I -A ext-in -p tcp --dport 1024:65535 --sport 1024:65535 \
! --syn -j ACCEPT
$I -A ext-out -p tcp --sport 1024:65535 --dport 1024:65535 \
-j ACCEPT
```

```
# - Forwarding durch die Firewall -----
# - - Kontrollverbindung

$R -A int-fw -p tcp --sport 1024:65535 --dport 21 -j ACCEPT
$R -A ext-fw -p tcp --dport 1024:65535 --sport 21 ! --syn \
-j ACCEPT

# - - Aktives FTP

$R -A int-fw -p tcp --sport 1024:65535 --dport 20 ! --syn \
-j ACCEPT
$R -A ext-fw -p tcp --dport 1024:65535 --sport 20 -j ACCEPT

# - - Passives FTP

$R -A int-fw -p tcp --sport 1024:65535 --dport 1024:65535 \
-j ACCEPT
$R -A ext-fw -p tcp --dport 1024:65535 --sport 1024:65535 \
! --syn -j ACCEPT

# - Proxy auf der Firewall -----
# - - Kontrollverbindung

# $R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" --dport 21 \
#-j ACCEPT
# $R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" --sport 21 \
#! --syn -j ACCEPT

# - - Aktives FTP

# - - - Klassischer Proxy z.B. ftp-gw

# $R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" --dport 20 \
#! --syn -j ACCEPT
# $R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" --sport 20 \
#-j ACCEPT

# - - - SuSE ftp-proxy im chroot
# - - - (benutzt hohen Port f"ur die Datenverbindung)

# $R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" --dport 2020 \
#! --syn -j ACCEPT
# $R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" --sport 2020 \
#-j ACCEPT

# - - Passives FTP

# $R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" --dport 1024:65535 \
#-j ACCEPT
# $R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" --sport 1024:65535 \
#! --syn -j ACCEPT
```

```
# -----
# DMZ
# -----

# ICMP

$I -A ext2dmz -p icmp --icmp-type 0 -j ACCEPT
$I -A ext2dmz -p icmp --icmp-type 3 -j ACCEPT
$I -A ext2dmz -p icmp --icmp-type 8 -j ACCEPT
$I -A ext2dmz -p icmp --icmp-type 11 -j ACCEPT
$I -A ext2dmz -p icmp --icmp-type 12 -j ACCEPT

$I -A dmz2ext -p icmp --icmp-type 0 -j ACCEPT
$I -A dmz2ext -p icmp --icmp-type 3 -j ACCEPT
$I -A dmz2ext -p icmp --icmp-type 8 -j ACCEPT
$I -A dmz2ext -p icmp --icmp-type 11 -j ACCEPT
$I -A dmz2ext -p icmp --icmp-type 12 -j ACCEPT

# DNS-Anfragen der DMZ-Server

for d in $DNSSERVER
do
    $I -A dmz2ext -p udp -d "$d" --dport 53 -j ACCEPT
    $I -A ext2dmz -p udp -m state --state ESTABLISHED \
    -s "$d" --sport 53 -j ACCEPT

    $I -A dmz2ext -p tcp -d "$d" --dport 53 -j ACCEPT
    $I -A ext2dmz -p tcp -m state --state ESTABLISHED \
    -s "$d" --sport 53 ! --syn -j ACCEPT
done

# Schutz von Servern vor direkten Zugriffen aus dem Internet

for i in $DMZIPPROT
do
    $I -A ext2dmz -p tcp -d "$i" -j LOG --log-level notice \
    --log-prefix "ext2dmz (ip): "
    $I -A ext2dmz -p tcp -d "$i" -j DROP
done

# Schutz von Diensten, die nur intern zugreifbar sein sollen

for p in $DMZUDPPROT
do
    $I -A ext2dmz -p udp --dport "$p" -j DROP
done

for p in $DMZTCPPROT
do
    $I -A ext2dmz -p tcp --dport "$p" -j DROP
done
```

```
# Einfache TCP-Server

for p in $DMZPORTS
do
    $R -A dmz2ext -p tcp --sport "$p" --dport 1024:65535 -j ACCEPT
    $R -A ext2dmz -p tcp --dport "$p" -j ACCEPT
done

# FTP

case "$DMZFTP" in
SRV)
# - Server in der DMZ ohne Filterung
# -- Kontrollverbindung

    $R -A dmz2ext -p tcp --sport 21 --dport 1024:65535 -j ACCEPT
    $R -A ext2dmz -p tcp --dport 21 -j ACCEPT

# -- Datenverbindungen
# --- aktiv

    $R -A dmz2ext -m state --state RELATED -p tcp \
    --sport "$DMZFTPDATAPOINT" --dport 1024:65535 -j ACCEPT

    $R -A dmz2ext -m state --state ESTABLISHED -p tcp \
    --sport "$DMZFTPDATAPOINT" --dport 1024:65535 -j ACCEPT

    $R -A ext2dmz -m state --state ESTABLISHED -p tcp \
    --sport 1024:65535 --dport "$DMZFTPDATAPOINT" -j ACCEPT

# --- passiv

    $R -A ext2dmz -m state --state RELATED -p tcp \
    --sport 1024:65535 --dport 1024:65535 -j ACCEPT

    $R -A ext2dmz -m state --state ESTABLISHED -p tcp \
    --sport 1024:65535 --dport 1024:65535 -j ACCEPT

    $R -A dmz2ext -m state --state ESTABLISHED -p tcp \
    --sport 1024:65535 --dport 1024:65535 -j ACCEPT

    ;;
PROXY)

# - Reverse Proxy auf der Firewall
# -- Kontrollverbindung

    $R -A ext-in -p tcp --sport 1024:65535 --dport 21 \
    -j ACCEPT
    $R -A ext-out -p tcp --dport 1024:65535 --sport 21 \
    ! --syn -j ACCEPT

    $R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" \
    --dport 21 -j ACCEPT
    $R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" \
    --sport 21 ! --syn -j ACCEPT
```

```
# -- Datenverbindung, aktiv

$I_R -A ext-in -p tcp --sport 1024:65535 --dport 2020 \
! --syn -j ACCEPT
$I_R -A ext-out -p tcp --dport 1024:65535 --sport 2020 \
-j ACCEPT

$I_R -A int-in -p tcp --sport 1024:65535 -d "$INTIP" \
--dport 2020 ! --syn -j ACCEPT
$I_R -A int-out -p tcp --dport 1024:65535 -s "$INTIP" \
--sport 2020 -j ACCEPT

# -- Datenverbindung, passiv

$I_R -A ext-in -p tcp --sport 1024:65535 --dport 1024:65535 \
-j ACCEPT
$I_R -A ext-out -p tcp --dport 1024:65535 --sport 1024:65535 \
! --syn -j ACCEPT

$I_R -A int-in -p tcp --sport 1024:65535 --dport 1024:65535 \
-d "$INTIP" -j ACCEPT
$I_R -A int-out -p tcp --dport 1024:65535 --sport 1024:65535 \
-s "$INTIP" ! --syn -j ACCEPT

# -- Transparente Umleitung auf die Firewall

$I_R -t nat -A PREROUTING -i "$EXTIF" -p tcp --dport 21 \
-j REDIRECT --to-port 21

$I_R -t nat -A PREROUTING -i "$INTIF" -p tcp -d "$DMZNET" \
--dport 21 -j REDIRECT --to-port 21

;;
esac

# -----
# Protokollierung ungew"ohnlicher Pakete
# -----

$I_R -A INPUT -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "INPUT (default): "
$I_R -A OUTPUT -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "OUTPUT (default): "
$I_R -A FORWARD -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "FORWARD (default): "
$I_R -A int-in -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "int-in (default): "
$I_R -A int-out -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "int-out (default): "
$I_R -A int-fw -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "int-fw (default): "
$I_R -A ext-in -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "ext-in (default): "
$I_R -A ext-fw -s 0.0.0.0/0 -j LOG --log-level notice \
--log-prefix "ext-fw (default): "
```

```
$R -A ext-out -s 0.0.0.0/0 -j LOG --log-level notice \  
--log-prefix "ext-out (default): "  
$R -A ext2dmz -s 0.0.0.0/0 -j LOG --log-level notice \  
--log-prefix "ext2dmz (default): "  
$R -A dmz2ext -s 0.0.0.0/0 -j LOG --log-level notice \  
--log-prefix "dmz2ext (default): "  
  
$R -A INPUT -s 0.0.0.0/0 -j DROP  
$R -A OUTPUT -s 0.0.0.0/0 -j DROP  
$R -A FORWARD -s 0.0.0.0/0 -j DROP  
$R -A int-in -s 0.0.0.0/0 -j DROP  
$R -A int-out -s 0.0.0.0/0 -j DROP  
$R -A int-fw -s 0.0.0.0/0 -j DROP  
$R -A ext-in -s 0.0.0.0/0 -j DROP  
$R -A ext-fw -s 0.0.0.0/0 -j DROP  
$R -A ext-out -s 0.0.0.0/0 -j DROP  
$R -A dmz2ext -s 0.0.0.0/0 -j DROP  
$R -A ext2dmz -s 0.0.0.0/0 -j DROP  
  
# Network Address Translation  
# - Masquerading  
  
$R -t nat -A POSTROUTING -s "$INTNET" -o "$EXTIF" -j MASQUERADE  
$R -t nat -A POSTROUTING -s "$INTNET" -o "$DMZIF" -j MASQUERADE  
  
# - Umleitung von Webserverzugriffen (TCP 80) auf den lokalen squid  
  
# $R -t nat -A PREROUTING -i "$INTIF" -p tcp --dport 80 -j REDIRECT \  
# --to-port 3128  
  
# - Umleitung von Webserverzugriffen (TCP 80) auf den lokalen privoxy  
  
# $R -t nat -A PREROUTING -i "$INTIF" -p tcp --dport 80 -j REDIRECT \  
# --to-port 8118  
  
# - Umleitung von DNS-Serverzugriffen auf den lokalen Server  
  
# $R -t nat -A PREROUTING -i "$INTIF" -p udp --dport 53 \  
# -j REDIRECT --to-port 53  
# $R -t nat -A PREROUTING -i "$INTIF" -p tcp --dport 53 \  
# -j REDIRECT --to-port 53  
  
# - Umleitung von FTP-Serverzugriffen (TCP 21) auf den lokalen ftp-proxy  
  
# $R -t nat -A PREROUTING -i "$INTIF" -p tcp --dport 21 -j REDIRECT \  
# --to-port 21  
  
;;  
esac
```

