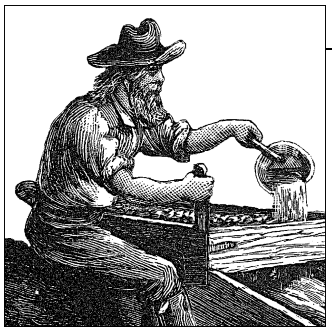

KAPITEL 12

Konfiguration der Paketfilter mit iptables

Nachdem wir im vorigen Kapitel die Filterung mit `ipchains` kennengelernt haben, wollen wir uns in diesem Kapitel mit `iptables` beschäftigen, dem Konfigurationswerkzeug für die Paketfilterung im Kernel der Serie 2.4. Man könnte zwar auch diesen Kernel mit Unterstützung für `ipchains` kompilieren, aber damit würden wir uns die neuen Möglichkeiten verbauen, die in ihm stecken.

Die Idee

Mit dem Kernel 2.4.0 wurden Paketfilterung und Network Address Translation zu einem Konzept zusammengefaßt. Beide Mechanismen werden dabei mit dem Befehl `iptables` konfiguriert. Mit ihm werden Regeln formuliert, welche anhand der Header-Information eines Paketes entscheiden, was mit ihm geschehen soll.

Diese Regeln werden in Gruppen, sogenannten »Chains«, zusammengefaßt. Einige dieser Chains sind vordefiniert und werden nur für bestimmte Pakete zu Rate gezogen. So existiert z. B. eine Chain, die nur Filterregeln enthält, die auf Pakete angewendet werden, die von dem Rechner weitergeleitet werden und nicht für ihn selbst bestimmt sind. Andere Chains können vom Benutzer selbst definiert und aus Regeln in den Standard-Chains angesprungen werden.

Je nach Aufgabe werden die Chains in mehrere »Tables« zusammengefaßt. Der Table `nat` enthält Chains für die Network Address Translation, während der Table `filter` Chains zur Paketfilterung enthält. Die für uns interessanten Chains dieser beiden Tables sind in Abbildung 12-1 dargestellt. Die mit »D-NAT« und »S-NAT« markierten Chains stehen dabei im Table `nat`, die mit »Filter« bezeichneten im Table `filter`.

Trifft nun ein Paket über ein Netzwerk-Interface ein, so werden als erstes Regeln der Chain `PREROUTING` im Table `nat` ausgeführt. Hier können gegebenenfalls Zieladresse und -port geändert werden. Dies ist insbesondere praktisch, wenn man einen transparenten Proxy aufsetzen, d. h. alle Anfragen, die direkt an Server im Internet gestellt werden, auf einen Proxy auf der Firewall umleiten will.

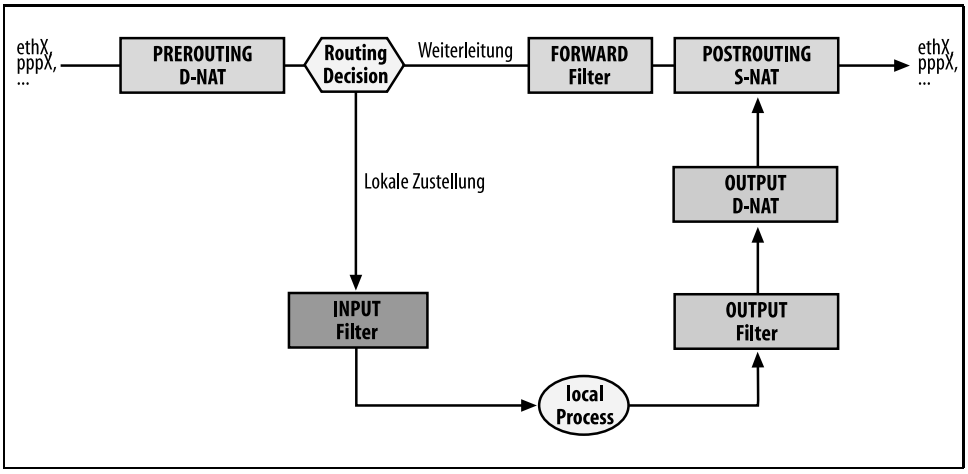

Abbildung 12-1: Network Address Translation und Paketfilterung im Kernel 2.4 nach [4] und [3]

Als nächstes wird eine Routing-Entscheidung getroffen. Falls das Paket für den Rechner selbst bestimmt ist, werden die Regeln in der Chain `INPUT` im Table `filter` angewendet. Hier wird entschieden, ob das Paket angenommen oder verworfen werden soll. Wird das Paket angenommen, so wird es nun an den Zielprozeß auf dem Rechner weitergeleitet.

Pakete, die nicht für den Rechner selbst bestimmt sind, sondern nur von ihm an ein anderes Netz weitergeleitet werden, werden statt dessen nach den Regeln in der Chain `FORWARD` im Table `filter` erlaubt oder verworfen.

Pakete, die von einem lokalen Prozeß ausgehen, werden erst einmal nach den Regeln der Chain `OUTPUT` im Table `filter` erlaubt oder verworfen. Hat ein Paket diese Prüfung bestanden, so besteht als nächstes die Möglichkeit, in der Chain `OUTPUT` im Table `nat` Zieladresse und -port zu ändern. Dies ist aber eine eher ungewöhnliche Maßnahme.

Für alle ausgehenden Pakete können schließlich mit den Regeln in der Chain `POSTROUTING` im Table `nat` Quelladresse und Port geändert werden. Damit kann z. B. Masquerading realisiert werden. Dabei werden alle Pakete so manipuliert, daß sie von der Firewall zu stammen scheinen. Dies ist insbesondere dann sinnvoll, wenn nur die Firewall selbst eine im Internet gültige Adresse besitzt. Darüber hinaus sorgt Masquerading dafür, daß die Strukur des internen Netzes aus dem Internet nicht sichtbar ist.

Policies

Jeder Standard-Chain im Table `filter` kann eine Grundregel mitgegeben werden, die Anwendung findet, wenn keine andere Regel greift. Dies nennt man eine *Policy*. Für `iptables` existieren die Policies:

ACCEPT Das Paket darf passieren.

DROP Das Paket wird verworfen.

Selbstdefinierte Chains besitzen keine Policies. Ist keine der Regeln der Chain anwendbar, so wird die nächste Regel derjenigen Chain untersucht, aus der in die selbstdefinierte Chain verzweigt wurde. Für das Einrichten einer Firewall erscheint die Policy ACCEPT nur in Sonderfällen sinnvoll. Üblicherweise wählt man beim Einrichten von Firewalls DROP nach dem Grundsatz: »Was nicht erlaubt ist, ist verboten.«

Die Syntax für das Erstellen einer Policy lautet:

```
iptables -P <Chain> <Policy>
```

Regeln

Eine Regel besteht aus Mustern und Aktionen. Die Muster legen fest, auf welche Pakete die Regel anzuwenden ist, und die Aktion definiert, was mit Paketen geschieht, wenn das Muster paßt.

Verwaltet werden die Regeln mit den folgenden Befehlen:

```
iptables [-t Table] -A chain Muster Aktion
iptables [-t Table] -I chain pos Muster Aktion
iptables [-t Table] -R chain pos Muster Aktion
iptables [-t Table] -D chain Muster Aktion
iptables [-t Table] -D chain pos
```

Dabei gilt:

-A hängt eine Regel hinten an eine Chain an (append).

-I fügt eine Regel an einer Position *pos* ein (insert). Ist *pos* ausgelassen, so wird die Regel als erste eingefügt. Die Numerierung der Regeln beginnt mit 1.

-R ersetzt die Regel an Position *pos* (replace).

-D löscht eine Regel, die entweder über ihre Position *pos* oder ihre genaue Definition spezifiziert wurde (delete).

Tables

Bisher sind drei Tables definiert worden. Das Konzept ist aber auf Erweiterbarkeit ausgelegt, so daß weitere hinzukommen können. Auch hängt das Vorhandensein der Tables davon ab, wie der Kernel kompiliert wurde und welche Module geladen sind.

filter Dieser Table ist auch die Standardvorgabe, wenn kein Table angegeben wurde. Hier werden in den Chains `INPUT`, `FORWARD` und `OUTPUT` Regeln vorgegeben, die die Filterung von Paketen betreffen.

nat Dieser Table enthält in den Chains `PREROUTING`, `OUTPUT` und `POSTROUTING` die Regeln zur Manipulation von Quell- und Zieladressen sowie -ports.

mangle Spezielle Manipulationen an Paketen, die den Rahmen dieser Darstellung sprengen. Nähere Details finden Sie in der Manpage zu `iptables`.

Muster

`iptables` ist auf Erweiterbarkeit ausgelegt. Nur wenige seiner Muster sind immer verfügbar. Weitere Muster können bei Bedarf hinzugeladen werden. Dies ist allerdings nur möglich, wenn Unterstützung für diese Erweiterungen als Modul oder Bestandteil des Kernels kompiliert wurde.

Wir wollen hier nur einen Blick auf die wichtigsten Muster werfen. Daneben existieren noch weitere, die Sie in der Manpage zu `iptables` nachschlagen können.

Standardmuster

Die folgenden Muster gehören zum Standardumfang. Jedes Muster kennt die Möglichkeit, mit »!« die Bedeutung des Musters in sein Gegenteil zu verkehren:

-p [!] Protokoll bezeichnet das verwendete Protokoll (tcp, udp, icmp oder eine numerische Angabe).

-s [!] Adresse[/Maske] bezeichnet die Quelladresse (source). Als Maske kann entweder die Anzahl der zu betrachtenden Bits oder eine Maske angegeben werden. *w.x.y.z/24* entspricht damit *w.x.y.z/255.255.255.0*.

-d [!] Adresse[/Maske] bezeichnet die Zieladresse (destination).

-i [!] interface[+] bezeichnet ein Interface, über das ein Paket empfangen wurde. Dabei ist es möglich, mit »+« alle Interfaces zu adressieren, die mit dem richtigen Namen anfangen. So würde `-i eth+` sowohl auf `eth0` als auch `eth1` passen. Dieses Muster kann nur in `INPUT`-, `FORWARD`- und `PREROUTING`-Chains verwendet werden.

-o [!] interface[+] bezeichnet das Interface, über das das Paket gesendet werden wird. Dieses Muster kann nur in `OUTPUT`-, `FORWARD`- und `POSTROUTING`-Chains verwendet werden.

[!] -f spezifiziert Folgefragmente. Diese enthalten weder einen Quell- oder Zielport noch einen ICMP-Typ. Die normalen Regeln werden daher normalerweise nicht auf Folgefragmente passen. Wenn Sie allerdings Network Address Translation oder Connection Tracking (z. B. für Stateful Packet Filtering oder Masquerading von FTP) benutzen, brauchen Sie sich keine Gedanken um Fragmente zu machen. In diesen Fällen werden Pakete sowieso zusammengefügt, bevor die Paketfilterregeln angewendet werden.

Protokollspezifische Erweiterungen

Einige Muster stehen nur zur Verfügung, wenn ein bestimmtes Protokoll spezifiziert wurde:

--sport Port[:Port] bezeichnet den Quellport (source port). Hierbei kann auch ein Bereich von Ports angegeben werden. Die Angabe »1:1023« würde z. B. den Bereich der privilegierten Ports abdecken.

Dieses Muster kann nur nach `-p tcp` oder `-p udp` verwendet werden.

--dport Port[:Port] bezeichnet den Zielport (destination port).

Kann nur nach `-p tcp` oder `-p udp` verwendet werden.

[!] --syn trifft auf Pakete zu, die einen Verbindungsaufbau bedeuten (SYN gesetzt, ACK und RST aber nicht).

Kann nur nach `-p tcp` verwendet werden.

[!] --tcp-flags Maske Aktiv *Maske* ist eine mit Kommata getrennte Liste von Flags, die angibt, welche der Flags SYN, ACK, FIN, RST, URG, PSH, ALL[1], NONE[2] uns interessieren. *Aktiv* gibt an, welche davon gesetzt sein sollen. So kann man `--syn` auch als `--tcp-flags SYN,RST,ACK SYN` schreiben.

Dieses Muster kann nur nach `-p tcp` verwendet werden.

--icmp-type [!] Typbezeichnung bezeichnet den Typ eines ICMP-Pakets mit einem logischen Namen oder einer Nummer. Eine Liste von Typbezeichnungen, die iptables kennt, kann mit `iptables -p icmp -h` erfragt werden.

Dieses Muster kann nur nach `-p icmp` verwendet werden.

Stateful Packet Filtering

Es existieren weitere Erweiterungen, die explizit mit `-m` *`Erweiterung`* im jeweiligen Aufruf von `iptables` geladen werden. Eine solche Erweiterung ist `state`. Wenn sie geladen wurde, kann das Muster `--state` *`Zustandsliste`* benutzt werden. Vier Zustände können benutzt werden. Sollen mehrere angegeben werden, so werden sie mit Kommata getrennt:

NEW Dieses Paket beginnt eine neue Verbindung.

ESTABLISHED Es handelt sich um ein Folgepaket einer bestehenden Verbindung.

RELATED Dieses Paket beginnt zwar eine neue Verbindung, diese steht aber in Zusammenhang mit einer bestehenden Verbindung (z. B. die Datenverbindung bei FTP, ICMP-Fehlermeldungen).

INVALID Pakete, die nicht eingeordnet werden können. Es schadet normalerweise nicht, diese Pakete generell zu verwerfen.

Aktionen

Ist ein Muster spezifiziert, gilt es, eine Aktion festzulegen, die von ihm ausgelöst werden soll. Dies geschieht mit

```
-j Target [Optionen]
```

Target kann hierbei u. a. eine der oben benannten Policies (ACCEPT, DROP) oder eine benutzerdefinierte Chain sein. Darüber hinaus existieren noch spezielle Targets, von de-

1 Dies ist kein Flag, sondern steht für »alle Flags«.

2 Dies ist kein Flag, sondern steht für »kein Flag«.

nen hier nur die wichtigsten beschrieben werden sollen. Eine ausführlichere Übersicht finden Sie in der Manpage zu `iptables`.

RETURN Das Paket »fällt aus der Chain«. D. h., es wird an die vorhergehende Chain zurückgereicht. Ist dies nicht möglich, weil es sich um die Regel einer Standard-Chain handelt, so tritt die Policy der Chain in Kraft.

LOG Das Paket wird im Systemprotokoll vermerkt. Dieses Target kennt mehrere Optionen, die die Protokollierung beeinflussen:

- ***--log-level Level*** gibt die Priorität der Meldung an. Die Angabe kann numerisch oder mit den in Kapitel 9, Abschnitt *Das Systemprotokoll*, ab Seite 205 besprochenen Bezeichnungen erfolgen.
- ***--log-prefix Prefix*** bezeichnet ein Präfix von bis zu 14 Zeichen, das der Protokollmeldung vorangestellt wird.
- ***--log-tcp-sequence*** bewirkt die Protokollierung der TCP-Folgenummer. Dies kann ein Sicherheitsrisiko sein, wenn normale Benutzer das Systemprotokoll lesen können.
- ***--log-tcp-options*** protokolliert die Optionen im TCP-Header.
- ***--log-ip-options*** protokolliert die Optionen im TCP-Header.

REJECT Das Paket wird verworfen, und es wird eine ICMP-Fehlermeldung an den Sender geschickt. Dieses Target kann nur im Table `filter` verwendet werden.

- ***--reject-with Typ*** gibt die Fehlermeldung an, die gesendet wird. Möglich sind `icmp-net-unreachable`, `icmp-host-unreachable`, `icmp-port-unreachable`, `icmp-proto-unreachable`, `icmp-net-prohibited` oder `icmp-host-prohibited`. Ohne diese Option wird `icmp-port-unreachable` verwendet. Als Reaktion auf ein Ping-Paket kann auch `echo-reply` verwendet werden. Für TCP-Pakete kann in der Chain `INPUT` oder einer eigenen Chain, in die aus `INPUT` verzweigt wurde, auch `tcp-reset` verwendet werden, wodurch ein TCP-Paket mit gesetztem RST-Flag gesendet wird.

SNAT Dieses Target legt fest, daß das Paket weitergeleitet, die Quelladresse aber verändert wird. Dies geschieht nicht nur für das Paket selbst, sondern auch für alle nachfolgenden Pakete derselben Verbindung.

Das Target kann nur in der `POSTROUTING`-Chain des Tables `nat` und in Chains, in die aus dieser verzweigt wurde, verwendet werden. Es kennt eine Option:

- ***--to-source Addr[-Addr][:Port-Port]*** legt fest, welche Quelladresse im Paket eingetragen werden soll. Hier kann auch ein Adreßbereich angegeben werden.

  Unter Umständen ist es auch nötig, den Quellport des Pakets zu ändern. Wenn die Regel die Protokollangabe `-p tcp` oder `-p udp` enthält, kann hierzu ein Portbereich angegeben werden, andernfalls wird darauf geachtet, daß für Pakete, deren Quellport in einem der Bereiche 1 – 511, 512 – 1023, 1023 – 65535 liegt, dieser jeweils durch einen Quellport aus demselben Bereich ersetzt wird. Wo immer möglich, wird die Portangabe nicht geändert.

DNAT ist das Gegenstück zu SNAT, hier wird die Zieladresse verändert. Gültig ist das Target nur in den `PREROUTING`- und `OUTPUT`-Chains des Tables `nat` sowie in daraus angesprungenen Chains.

Die nötigen Angaben erfolgen mit der folgenden Option:

--to-destination Addr[-Addr][:Port-Port] Hier kann sowohl ein Bereich von Adressen wie auch Ports angegeben werden, aus denen eine neue Zielangabe generiert wird. Eine Portangabe ist dabei allerdings nur zulässig, wenn die Regel `-p tcp` oder `-p udp` enthält. Fehlt die Portangabe, so wird der Zielport nicht geändert.

MASQUERADE Dieses Target legt fest, daß die Quelladresse des Pakets sowie nachfolgender Pakete derselben Verbindung in die Adresse des Interfaces geändert wird, über das es den Rechner verläßt. Wird der Zustand des Interfaces auf »down« geändert, so werden alle bestehenden Verbindungen »vergessen«. Dies ist insbesondere bei der Verwendung von dynamischen IP-Adressen sinnvoll. Beim nächsten Aktivieren des Interfaces wird ihm hier vielleicht schon eine ganz andere Adresse zugewiesen.

Das Target kann nur in Chains im Table `nat` verwendet werden. Hier auch nur in `POSTROUTING` sowie in Chains, in die aus `POSTROUTING` verzweigt wurde.

Wenn in der Regel `-p tcp` oder `-p udp` spezifiziert wurde, kann die folgende Option verwendet werden:

--to-ports Port[-Port] legt einen Portbereich fest, aus dem nötigenfalls ein Quellport gewählt wird. Es gelten dieselben Regeln wie bei SNAT.

REDIRECT Dieses Target entspricht DNAT. Allerdings wird die Zieladresse auf den Rechner selbst gesetzt. Auf diese Weise können z. B. Anfragen an Server im Internet auf einen Proxy auf der Firewall umgeleitet werden.

Wenn in der Regel `-p tcp` oder `-p udp` spezifiziert wurde, kann die folgende Option verwendet werden:

--to-ports Port[-Port] legt einen Bereich von Ports fest, aus denen ein Zielport für das Paket gewählt wird.

Verwalten von Chains

Eine Reihe von Befehlen dient dazu, Chains zu administrieren:

iptables [-t Table] -N Chain Eine neue Chain anlegen (new). Der Bezeichner *Chain* darf dabei allerdings nicht länger als acht Zeichen sein.

iptables [-t Table] -F [Chain] Alle Regeln (der Chain) löschen (flush).

iptables [-t Table] -X Chain Eine Chain löschen. Dies gelingt nur, wenn sie keine Regeln enthält. (Der Buchstabe wurde laut [3] gewählt, weil alle sprechenden schon vergeben waren.)

iptables [-t Table] -L [Chain] [-v] [-n] [--linenumbers] Die Regeln (der Chain) anzeigen (-n: numerisch, -v: ausführlich, –linenumbers: Regeln numerieren) (list).

iptables [-t Table] -Z Die Paketzähler aller Chains zurücksetzen (zero).

Einige Beispiele

Im folgenden wollen wir einige konkrete Beispiele für Paketfilterregeln betrachten. Zusammen bilden sie die Basis für einen maskierenden Paketfilter, der leicht an die eigenen Bedürfnisse angepaßt werden kann.

Dabei ist allerdings zu beachten, daß die Reihenfolge der Filterregeln nicht willkürlich gewählt wurde. So sollten zuerst die bisher geltenden Regeln gelöscht, neue Policies definiert und Anti-Spoofing-Regeln eingerichtet werden, bevor man die eigentlichen Regeln für die zu filternden Protokolle aufstellt. Protokollregeln, die Einfluß auf andere Regeln für andere Protokolle haben (insbesondere FTP), sollten dabei als letztes definiert werden. Den Abschluß bilden schließlich Regeln für all jene Pakete, die von den bisherigen Regeln nicht erfaßt wurden.

Wir gehen davon aus, daß wir uns in einem lokalen Netz mit zwei Rechnern befinden, welche die Adressen *192.168.20.100* (Klient) und *192.168.20.15* (Firewall) haben. Unser DNS-Server ist *10.0.0.77*. Die Adresse, unter der die Firewall im Internet bekannt ist, sei in der Variablen *EXTIP* gespeichert. Das Netzwerk-Interface sei `eth0` für das innere Netz, während der Name des externen Interfaces in *EXTIF* gespeichert sei. Verwenden wir z. B. ein Modem und haben den `pppd` wie in Kapitel 10, Abschnitt *Einrichten von Modem oder DSL*, ab Seite 227 beschrieben konfiguriert, müssen wir die folgenden Zeilen in unser Skript eintragen:

```
EXTIP=10.1.0.1
EXTIF=ppp0
```

Vergibt unser Provider IP-Adressen dynamisch, so kann es zu Problemen kommen, wenn wir in unseren Regeln die Adresse des externen Interfaces explizit benutzen[3]. Regeln, die davon betroffen sind, sind hier deswegen mit einem Achtungszeichen markiert. Wir werden später sehen, wie man die daraus resultierenden Probleme umgehen kann.

Schließlich müssen wir noch entscheiden, was wir mit Paketen machen, die wir nicht annehmen wollen. Wir haben zwei Möglichkeiten. `REJECT` sendet eine Fehlermeldung, die dem Absender mitteilt, daß auf dem gewünschten Port kein Dienst verfügbar ist. `DROP` verwirft das Paket dagegen, ohne den Absender zu informieren. Das ist prinzipiell eher unhöflich, da der Absender nicht weiß, was passiert, und es daher nach einer gewissen Wartezeit erneut probieren wird. Erst nach mehreren Versuchen ohne Antwort gibt er den Verbindungsaufbau auf.

3 Prinzipiell ist es sinnvoll, das Firewalling einzurichten, bevor die Netzwerk-Interfaces aktiviert werden. Allerdings kennen wir bei dynamischer Adreßvergabe die IP-Adresse des externen Interfaces erst, wenn die Verbindung zum Internet schon aufgebaut wurde.

Im Normalfall ist das aber nicht schlimm, da Rechner im Internet keinen Grund haben, eine Verbindung zu uns aufzumachen. Eine Ausnahme bildet hier nur das Ident-Protokoll, auf das ich aber in einem eigenen Abschnitt eingehen werde. Alle anderen Verbindungsaufnahmen sind aus unserer Sicht ungerechtfertigt und stellen in der Regel einen Port Scan dar. Wir haben also keinen Grund, besonders höflich zu sein.

Würden wir `REJECT` verwenden, so teilen wir dem Angreifer umgehend mit, daß der besagte Port nicht zur Verfügung steht und er daher mit dem Scannen des nächsten Ports fortfahren kann. Das beschleunigt seine Arbeit ungemein.

Verwenden wir dagegen `DROP`, so erhält er keine Rückmeldung. Er wird daher erst einmal eine Weile warten, um auszuschließen, daß die Übermittlung einfach nur langsam ist und die Antwort durch einen Stau auf der Datenautobahn aufgehalten wurde. Erst dann wird er beschließen, daß der Port nicht verfügbar ist.

Dies bedeutet, daß er deutlich länger zur Untersuchung eines Ports braucht. Will er den Rechner trotzdem in der gleichen Zeit scannen, so muß er zusätzliche Ressourcen einsetzen und mehrere Ports parallel untersuchen. Das heißt aber, daß diese Ressourcen nicht für die Untersuchung anderer Rechner zur Verfügung stehen. Da Angreifer oft nicht nur einzelne Rechner, sondern ganze Subnetze untersuchen, bedeutet der konsequente Einsatz von Firewalls, die Pakete ohne Rückmeldung verwerfen, daß Port Scans bedeutend länger dauern.

Aus diesem Grund werde ich im folgenden in der Regel `DROP` verwenden, wenn ein Paket verworfen werden soll. Ich sollte aber noch darauf hinweisen, daß dies auch dazu führen kann, daß versuchte Anfragen mehrfach im Systemprotokoll auftauchen, da der Anfragende Pakete mehrfach sendet, um auszuschließen, daß ein Paket im Netz einfach verlorengegangen ist.

Im Normalfall ist dies kein Problem. Im Falle einer Epidemie eines Internet-Wurms erhalten wir aber in der Regel nicht nur eine Anfrage eines einzelnen Rechners, sondern von einer ganzen Flut bereits infizierter Rechner. In so einem Fall wird unser Systemprotokoll sowieso schon überquellen. Wenn der Wurm aber nun auch noch jedes Anfragepaket mehrfach sendet, vervielfacht sich das Problem.

In so einem Fall läge es wirklich nahe, für die Zeit der Epidemie den vom Wurm verwendeten Port von einer `DROP`- auf eine `REJECT`-Regel umzustellen. Allerdings kann ich dafür keine Regel angeben, da ich dazu den Port kennen müßte, den der Wurm verwendet. Sie würde aber bis auf den Zielport der Regel für das Ident-Protokoll entsprechen, die ich in einem der folgenden Abschnitte beschreiben werde.

Die absolut sichere Firewall

Diese Regeln verbieten jeglichen Netzwerkzugriff:

```
# Alle Regeln und selbstdefinierten Chains löschen
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X

# Policies, die alle Pakete abweisen
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
```

Da wir später auch den Table `nat` verwenden werden, sollten wir auch für ihn Policies festlegen. Die folgende Festlegung dient aber nur der Vollständigkeit und stellt den Grundzustand her, in dem generell alle Pakete erlaubt sind.

Daß wir hier nicht `DROP` verwenden, liegt daran, daß wir den Table `nat` nicht zur Filterung benutzen, sondern in ihm nur Regeln definieren, die der Manipulation von Adreßangaben im Paketheader dienen. Die Filterung findet im Table `filter` statt:

```
# 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
```

Schutz vor Spoofing

Wir gehen davon aus, daß Spoofing-Angriffe nur von außerhalb unseres lokalen Netzes erfolgen. Hier nun Regeln, um die Angriffe zu verhindern und zu protokollieren:

```
# gespoofte Pakete des lokalen Netzes
iptables -A INPUT -i ! eth0 -s 192.168.20.0/24 -j LOG \
--log-prefix "Internes Netz gespooft: "
iptables -A INPUT -i ! eth0 -s 192.168.20.0/24 -j DROP
iptables -A FORWARD -i ! eth0 -s 192.168.20.0/24 -j LOG \
--log-prefix "Internes Netz gespooft: "
iptables -A FORWARD -i ! eth0 -s 192.168.20.0/24 -j DROP

# gespoofte Pakete des lokalen Interfaces
iptables -A INPUT -i ! lo -s 127.0.0.1 -j LOG \
--log-prefix "Loopback gespooft: "
iptables -A INPUT -i ! lo -s 127.0.0.1 -j DROP
iptables -A FORWARD -i ! lo -s 127.0.0.1 -j LOG \
--log-prefix "Loopback gespooft: "
iptables -A FORWARD -i ! lo -s 127.0.0.1 -j DROP

# gespoofte Pakete des externen Interfaces der Firewall
iptables -A INPUT -i ! lo -s "$EXTIP" -j LOG\
--log-prefix "$EXTIP gespooft: "
iptables -A INPUT -i ! lo -s "$EXTIP" -j DROP
```

```
iptables -A FORWARD -i ! lo -s "$EXTIP" -j LOG \
--log-prefix "$EXTIP gespooft: "
iptables -A FORWARD -i ! lo -s "$EXTIP" -j DROP
```

Hierbei tritt nun zum ersten Mal ein Problem bei der dynamischen IP-Adreßvergabe auf. Es wird die externe IP-Adresse benutzt, obwohl sie uns erst zugewiesen wird, wenn wir eine Verbindung zu unserem Provider aufgebaut haben.

Wir haben nun drei Möglichkeiten:

1. Wir können sämtliche Filterregeln in `/etc/ppp/ip-up` eintragen, wo sie erst dann angewendet werden, wenn die Adresse des externen Interfaces schon bekannt ist.
2. Wir können auch nur die betroffenen Regeln dort eintragen und sie dann in `/etc/ppp/ip-down` selektiv löschen.
3. Wir können uns auf den Schutz durch die in Kapitel 8, Unterabschnitt *Konfiguration des /proc-Dateisystems*, ab Seite 166 besprochenen Kerneleinstellungen verlassen.

Die erste Möglichkeit ist nicht sehr attraktiv. Sie ist äußerst aufwendig, da bei jedem Start erst einmal alle Regeln neu aufgesetzt werden müßten. Auch wartet der `pppd` nicht, bis das Skript `/etc/ppp/ip-up` ausgeführt wurde, bevor er damit beginnt, Pakete weiterzuleiten. Damit wäre die Firewall zumindest theoretisch eine Weile mit dem Internet verbunden, bevor er damit beginnt, die eingehenden Pakete auch zu filtern.

Die zweite Möglichkeit ist durchaus praktikabel. `iptables` erlaubt es, Regeln gezielt an eine bestimmte Stelle in einer Chain einzufügen. Kapitel 11, Abschnitt *Eintragen der Regeln in die Systemdateien*, ab Seite 296 beschreibt, wie dies aussehen kann. Dabei müssen wir allerdings angeben, an welcher Stelle die Regel eingefügt werden soll. Aus diesem Grund sollten Sie sich gründlich überlegen, ob Sie auf diese Möglichkeit zurückgreifen. Andernfalls könnte es geschehen, daß die Regeln in eine falsche Reihenfolge geraten, wodurch nicht abzusehende Nebeneffekte resultieren könnten.

Die dritte Möglichkeit schützt uns genauso effektiv wie die zweite. Sie hat allerdings den Nachteil, daß wir auf diese Weise keine Logeinträge erhalten, wenn doch einmal Pakete mit gefälschter Quelladresse auftreten.

Ident (Auth)

Bei diesem Dienst handelt es sich um eine Methode, mit deren Hilfe ein Rechner bei einem anderen nachfragen kann, welcher Benutzer eine bestimmte Verbindung zu ihm aufgemacht hat. In unserem Szenario wäre es extrem aufwendig und wenig zweckmäßig, dies zuzulassen. Allerdings sind derartige Anfragen kein Anzeichen eines Angriffes, sondern entspringen in der Regel der legitimen Neugier von frei zugänglichen Servern. Es ist daher sinnvoll, für diese Anfragen eine REJECT-Regel zu definieren, um der Gegenstelle mitzuteilen, daß dieser Dienst nicht zugreifbar ist. Im folgenden Beispiel wird der Zugriff trotzdem im Systemlog vermerkt:

```
# Ident

iptables -A INPUT -i "$EXTIF" -p tcp --dport 113 -j LOG \
--log-prefix "ident probe: "
iptables -A INPUT -i "$EXTIF" -p tcp --dport 113 -j REJECT
```

Unerwünschter Aufbau von Verbindungen

Da der Kernel 2.4 auch Stateful Packet Filtering beherrscht, bietet es sich an, Regeln zu definieren, die sämtliche Pakete verwerfen, die weder zu einer bestehenden Verbindung gehören (ESTABLISHED) noch zu einem Verbindungsaufbau, der in Zusammenhang mit einer bestehenden Verbindung steht (RELATED).[4]

Kurzum: Wir verbieten ungültige Pakete (INVALID) und solche, die einen neuen Verbindungsaufbau (NEW) einleiten:

```
# Unaufgeforderte Verbindungsaufbauten (NEW) und ungültige Pakete
# (INVALID) des externen Interfaces.

# - Eine eigene Chain für diese Tests

iptables -N states
iptables -F states

# - Dorthin verzweigen

iptables -A INPUT -i "$EXTIF" -j states
iptables -A FORWARD -i "$EXTIF" -j states

# - Die Regeln

iptables -A states -m state --state NEW,INVALID -j LOG \
--log-prefix Ünerwuenschte Verbindung:"

iptables -A states -m state --state NEW,INVALID -j DROP
```

Hier bauen wir eine eigene Chain für die Prüfung auf, in die sowohl aus der `INPUT`- als auch aus der `FORWARD`-Chain verzweigt wird. Dies hat den Vorteil, daß wir die eigentlichen Regeln nicht doppelt pflegen müssen. Damit wird vermieden, daß wir versehentlich verschiedene Regeln für weiterzuleitende und für die Firewall selbst bestimmte Pakete definieren.

Es kann allerdings vorkommen, daß wir in Einzelfällen doch einen Verbindungsaufbau von außen erlauben wollen. In diesem Fall müssen wir eine zusätzliche Regel **vor** den bereits definierten Regeln einfügen. Dies geschieht, indem wir als Aktion `-I` anstelle von `-A` benutzen. Als Ziel verwenden wir `RETURN`, womit die Chain `states` verlassen wird und die Regeln zur Protokollierung und zum Verwerfen des Pakets nicht zum Tragen kommen.

Ein Beispiel dafür finden Sie in Kapitel 12, Unterabschnitt *ICMP*, ab Seite 320.

4 Der letzte Typ bezieht sich z. B. auf eingehende FTP-Datenverbindungen.

Leser der ersten Auflage fragten mich, warum ich hier Stateful Packet Filtering nur dazu benutze, verdächtige Pakete zu verwerfen. Man könne doch mit Stateful Packet Filtering mit wenigen Regeln eine komplette Firewall aufsetzen, wenn man einfach alle eingehenden Pakete akzeptiert, die im Status `ESTABLISHED` oder `RELATED` sind.

Dies hat zwei Gründe. Zum einen gehört es zum Konzept der hier vorgestellten Firewall, explizit bestimmte Protokolle zu erlauben, anstatt einfach nur alle ausgehenden Verbindungen zu erlauben und alle eingehenden zu verbieten.

Zum anderen ist das Stateful Packet Filtering eine ziemlich junge Errungenschaft des Linux-Kernels. Es hat eine recht komplizierte Aufgabe zu erfüllen und ist nicht leicht zu realisieren. Während die normale Paket-Filterung im Prinzip nur die ersten Bytes eines Datenpaketes zu betrachten braucht, um zu entscheiden, ob ein Paket passieren darf oder nicht, hat es das Stateful Packet Filtering deutlich schwerer. Es muß das zu filternde Anwendungsprotokoll zum Teil nachbilden.

Um zu entscheiden, ob z. B. eine eingehende Verbindung von Port 20 eines Servers eine FTP-Datenverbindung ist, die zu einer bestehenden FTP-Kontrollverbindung gehört, beobachtet ein Kernelmodul alle ausgehenden TCP-Verbindungen zu Port 21 anderer Rechner. Es wartet darauf, daß der Befehl `PORT` übertragen wird, und ermittelt daraus, von welchem Rechner eine Verbindung zu welchem Port der Firewall geöffnet werden wird.

Dabei kann einiges schiefgehen, was dann dazu führt, daß beliebige Verbindungen freigeschaltet werden und vorher durch die Firewall geschützte Serverdienste frei aus dem Internet zugreifbar sind. Dies passiert nicht nur CISCO [61], sondern auch den Linux-Kernel-Entwicklern [60].

Auch andere Protokolle kennen zusätzliche Datenverbindungen, die dynamisch geöffnet werden. So kennt Internet Relay Chat (IRC) den Direct Client Connect (DCC), bei dem Dateien direkt zwischen zwei Chat-Teilnehmern ausgetauscht werden. Auch hier existierte im Linux-Kernel schon einmal ein Bug, der dazu führte, daß beliebige Verbindungen zu beliebigen Ports eines Klienten geöffnet werden konnten [62].

Neben der schieren Komplexität der Aufgabe, ein Anwendungsprotokoll im Kernel nachzubilden, besteht auch noch das Problem, daß dies für jedes Protokoll geschehen muß, für das ein Stateful Packet Filtering durchgeführt wird. Damit existieren diverse Stellen im Kernel, an denen sich ein Fehler in die Programmierung eingeschlichen haben könnte.

Ich halte es daher für durchaus möglich, daß sich auch weiterhin Fehler im Stateful Packet Filtering finden werden. Daher werden Sie im folgenden einige Regeln finden, die überflüssig sind, wenn man davon ausgeht, daß das Stateful Packet Filtering funktioniert. Sie stellen eine zweite Verteidigungslinie dar, falls sich diese Annahme einmal als falsch erweist.

Das Loopback-Interface

Wird von der Firewall aus auf Dienste zugegriffen, die auf der Firewall laufen, so wird dazu ein spezielles Netzwerk-Interface namens `lo` benutzt. Da keine externen Pakete über dieses Interface geroutet werden, existiert kein Grund, Verkehr über dieses Interface nicht zuzulassen. Dies betrifft allerdings nur die Chains `INPUT` und `OUTPUT`. Die Chain `FORWARD` wird für die lokale Paketvermittlung niemals benutzt.

```
# Lokale Pakete

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

NetBIOS

Obwohl NetBIOS sicherlich nicht zu den Protokollen gehört, die eine Firewall weiterleiten sollte, so kann es doch sinnvoll sein, dafür eigene Regeln zu definieren, die die Flut von NetBIOS-Paketen, die in jedem Netz mit Windows-Rechnern kursieren, ohne Logeinträge entsorgen. NetBIOS benutzt die Ports 137, 138 und 139, wobei diese nicht nur als Zielports, sondern auch als Quellports in DNS-Abfragen auftauchen.

Anfragen aus dem Internet an lokale NetBIOS-Dienste sollten dagegen sehr wohl protokolliert werden, da diese Anzeichen für einen Angriff darstellen könnten.

```
# NetBIOS ueber TCP/IP

iptables -A INPUT -p UDP -s 192.168.20.0/24 --sport 137:139 -j DROP
iptables -A INPUT -p UDP -s 192.168.20.0/24 --dport 137:139 -j DROP
iptables -A INPUT -p TCP -s 192.168.20.0/24 --sport 137:139 -j DROP
iptables -A INPUT -p TCP -s 192.168.20.0/24 --dport 137:139 -j DROP

iptables -A FORWARD -p UDP -s 192.168.20.0/24 --sport 137:139 -j DROP
iptables -A FORWARD -p UDP -s 192.168.20.0/24 --dport 137:139 -j DROP
iptables -A FORWARD -p TCP -s 192.168.20.0/24 --sport 137:139 -j DROP
iptables -A FORWARD -p TCP -s 192.168.20.0/24 --dport 137:139 -j DROP
```

ICMP

Fügen wir keine Regeln für ICMP ein, so führt dies dazu, daß die Firewall keinerlei Fehlermeldungen von anderen Rechnern erhält. Dies bedeutet, daß Zugriffe auf nicht existierende Rechner oder Dienste der Firewall nicht mitgeteilt werden können, weshalb die Firewall eine geraume Zeit auf Antwortpakete wartet, die niemals kommen werden. Darüber hinaus benutzt Linux ICMP-Pakete, um herauszufinden, was die maximale Paketgröße ist, die über eine bestimmte Strecke übertragen werden kann. Auch hierbei wird die Nutzung des Internets bei einem Mißlingen des Vorganges nicht unmöglich, die Leistung der Firewall leidet aber darunter.

Es ist also sinnvoll, bestimmte ICMP-Pakete an die Firewall zuzulassen. Allerdings können andere Pakete zu Angriffen mißbraucht werden (siehe Kapitel 4, Unterabschnitt *Angriffe mittels ICMP*, ab Seite 30). Es ist daher nicht empfehlenswert, alles zu erlauben.

In Tabelle 12-1 finden Sie eine Übersicht gebräuchlicher ICMP-Meldungen. Wirklich gefährlich sind dabei vor allem »Redirect«, »Router Advertisement« und »Router Solicitation«, da das Potential von Angriffen, die auf ihnen basieren, über einfache Denial-of-Service-Angriffe deutlich hinausgeht. Während man mit »Source Quench« einen Rechner vielleicht am Senden von Paketen hindern kann, kann man mit diesen Nachrichten dafür sorgen, daß Pakete einen bestimmten Weg nehmen. Dies erlaubt es einem Angreifer unter anderem, bestimmte Verbindungen über den eigenen Rechner zu routen und zu manipulieren.

Tabelle 12-1: Gebräuchliche ICMP-Meldungen

Nummer	Bedeutung	Bewertung
0	Antwort auf ein Ping (Echo Reply)	theoretisch DoS, wird allerdings für `ping` benötigt
3	Empfänger nicht erreichbar (Destination Unreachable)	theoretisch DoS, bei Filterung Timeout statt Fehlermeldung (s. o.)
4	Nachrichtenunterdrückung (Source Quench)	kann zu DoS-Angriffen genutzt werden
5	Paketumleitung (Redirect)	Kontrolle des Routing durch den Angreifer
8	Ping (Bitte um Echo) (Echo Request)	theoretisch DoS, ermöglicht aber Netzwerkdiagnosen
9	Router-Bekanntgabe (Router Advertisement)	Kontrolle des Routing durch den Angreifer
10	Router-Auswahl (Router Solicitation)	Kontrolle des Routing durch den Angreifer
11	Zeitüberschreitung (Time Exceeded)	theoretisch DoS, ansonsten eher nützlich
12	Ungültiger Header (Parameter Problem)	unproblematisch

»Destination Unreachable« sollte möglichst nicht gefiltert werden, da es sich hier um eine ganze Klasse von Fehlermeldungen handelt, die immer dann gesendet werden, wenn eine Verbindung nicht zustande kommt. Auch »Time Exceeded« ist nützlich, da es erlaubt, den Weg, den Pakete durch das Netz nehmen, mit z. B. `traceroute` nachzuvollziehen. Aus dem gleichen Grund sollte man »Echo Reply« annehmen. Es bildet die Basis für `ping`. Wenn man sein Gegenstück »Echo Request« filtert, so nimmt man anderen die Möglichkeit zu überprüfen, ob die Firewall erreichbar ist. Auch bei »Parameter Problem« handelt es sich um eine normale Fehlermeldung, die nicht gefiltert werden sollte.

Eine Weiterleitung von ICMP-Paketen ist dagegen generell nicht zu empfehlen. Es existieren Trojaner, die ICMP-Nachrichten benutzen, um heimlich Informationen zu übertragen.

Hinzu kommt, daß bestimmte DoS-Angriffe auf ICMP basieren. Indem wir also ICMP nicht routen, verhindern wir, daß derartige Angriffe den Verkehr innerhalb unseres LAN beeinträchtigen. Zwar können wir so nicht verhindern, daß ein DoS-Angriff uns vom

Internet abschneidet, die Kommunikation im lokalen Netz wird aber zumindest nicht beeinträchtigt.

Auch ist es nicht wirklich notwendig, daß normale Anwender auf den Klientenrechnern in die Lage versetzt werden, Rechner im Internet anzupingen. Hier greift das grundlegende Prinzip, Dinge nur zu erlauben, wenn sie wirklich notwendig sind.

Der letzte Punkt, der zu beachten wäre, ist schließlich das Senden von ICMP-Paketen. Hier ist es nicht nötig zu filtern. Da wir ICMP-Pakete nicht routen, können die fraglichen Pakete nur von der Firewall selbst stammen und sollten daher kein Problem darstellen.

Hier ein Beispiel:

```
# ICMP

iptables -A INPUT  -p icmp --icmp-type  0 -j ACCEPT
iptables -A INPUT  -p icmp --icmp-type  3 -j ACCEPT
iptables -A INPUT  -p icmp --icmp-type  8 -j ACCEPT
iptables -A INPUT  -p icmp --icmp-type 11 -j ACCEPT
iptables -A INPUT  -p icmp --icmp-type 12 -j ACCEPT
iptables -A OUTPUT -p icmp                -j ACCEPT

iptables -I states -p icmp --icmp-type  8 -j RETURN
```

Die letzte Regel ist nötig, damit »Echo Request«-Pakete nicht durch das Stateful Packet Filtering blockiert werden. Ping-Anfragen werden nämlich von der Paketfilterung auch als Verbindungsaufbauten angesehen. Hingegen gelten »Echo Reply«-Pakete und Fehlermeldungen als Antworten und können ungehindert passieren.

Da wir das Stateful Packet Filtering in eine eigene Chain verlegt haben, brauchen wir nun nur eine Regel **vor** den allgemeinen Regeln einzufügen, welche die Bearbeitung der Regeln der Chain abbricht. Der Einfachheit halber fügen wir sie gleich als allererste Regel ein (`-I` <*chain*>), so daß wir sicher wissen, daß sie auf jeden Fall vor den allgemeinen Regeln bearbeitet wird.

Eigene Chains

Obwohl die folgenden Regeln auch problemlos für die Standard-Chains definiert werden könnten, hat die Aufteilung in verschiedene Chains durchaus ihre Vorteile. Zum einen müssen vom Kernel weniger Regeln überprüft werden, da nur die Regeln der jeweils anwendbaren Chain untersucht werden müssen, zum anderen können die Regeln so in logische Bereiche aufgeteilt werden. Dies macht die Formulierung einfacher und hilft Fehler zu vermeiden, wenn mehr als zwei Netzwerkstränge verwaltet werden müssen.

```
# Eigene Chains

# - Externes Interface

iptables -N ext-in
iptables -N ext-fw
iptables -N ext-out
```

```
# - Internes Interface

iptables -N int-in
iptables -N int-fw
iptables -N int-out

# - Verteilung der Pakete auf die Chains

iptables -A INPUT   -i eth0 -s 192.168.20.0/24 -j int-in
iptables -A INPUT   -i "$EXTIF" -j ext-in
iptables -A FORWARD -i eth0 -o "$EXTIF" -s 192.168.20.0/24 -j int-fw
iptables -A FORWARD -i "$EXTIF" -o eth0 -j ext-fw
iptables -A OUTPUT  -o eth0 -j int-out
iptables -A OUTPUT  -o "$EXTIF" -j ext-out
```

Blockieren des Zugriffs auf lokale Serverdienste

Grundsätzlich sollten unaufgefordert aufgebaute Verbindungen schon durch das Stateful Packet Filtering abgefangen werden. Es kann aber nicht schaden, noch einmal gezielt nachzusehen, welche Dienste auf der Firewall aktiv sind, und den Zugang zu diesen mit eigenen Filterregeln zu verbieten.

Dabei interessieren uns weniger die Dienste, die auf den sogenannten *privilegierten Ports* unter 1024 angesiedelt sind. Für diese ist normalerweise sowieso keine Filterregel eingerichtet, die Pakete akzeptiert. Anders sieht es mit Diensten auf *hohen Ports* über 1023 aus. Der Zugriff ist – vom Stateful Packet Filtering einmal abgesehen – normalerweise erlaubt, um FTP zu ermöglichen.

Eine wichtige Hilfe bei dieser Suche bietet der Aufruf

```
netstat -an | grep -v '^unix'
```

In einer Installation, in der auf dem Rechner sowohl ssh (Port 22 TCP), ein Mailserver (Port 25 TCP), ein DNS-Server (Port 53 UDP/TCP), Druckdienste (Port 515 TCP) als auch ein `squid` als Webproxy (in der Standardinstallation: Port 3128 TCP) betrieben wird, könnte dies so aussehen:

```
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:3128            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:515             0.0.0.0:*               LISTEN
tcp        0      0 10.0.0.1:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
udp        0      0 10.0.0.1:53             0.0.0.0:*
udp        0      0 127.0.0.1:53            0.0.0.0:*
raw        0      0 0.0.0.0:1               0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
```

Hier sticht vor allem der `squid` hervor. Er benutzt einen hohen Port. Falls wir also nicht den Wunsch haben, einen frei zugänglichen Proxyserver zu betreiben[5], sollte dieser Port besser vor Zugriffen von außen geschützt werden.[6]

Folgende Filterregel bietet sich daher an:

```
# Zugriffe auf Server der Firewall
# - squid

iptables -A ext-in -p tcp --dport 3128 -j LOG \
--log-prefix "Externer Zugriff auf den Squid:"
iptables -A ext-in -p tcp --dport 3128 -j DROP
```

Da wir jetzt noch nicht endgültig wissen, auf welchen Ports später Serverdienste auf Anfragen warten werden, sollten wir den hier beschriebenen Schritt nach der Installation aller Serverdienste wiederholen. Auch nach eventuellen Upgrades des Systems sollten wir sicherstellen, daß dadurch nicht versehentlich neue Ports geöffnet wurden.

DNS

Bei DNS werden Pakete von einem hohen Port (> 1023) des Klienten an Port 53 des DNS-Servers gesendet. Hierzu wird in der Regel UDP verwendet. Ist die Antwort des Servers allerdings länger als 512 Bytes, so wird TCP verwendet.

Beginnen wir unsere Regeln also damit, daß wir als erstes unserer Firewall selbst erlauben, auf den DNS-Server zuzugreifen:

```
# DNS
# - Zugriff auf den externen Server

iptables -A ext-in  -p udp -s 10.0.0.77 --sport 53 --dport 1024:65535 \
-j ACCEPT
iptables -A ext-out -p udp -d 10.0.0.77 --dport 53 --sport 1024:65535 \
-j ACCEPT

iptables -A ext-in  -p tcp -s 10.0.0.77 --sport 53 --dport 1024:65535 \
! --syn -j ACCEPT
iptables -A ext-out -p tcp -d 10.0.0.77 --dport 53 --sport 1024:65535 \
-j ACCEPT
```

Wir wollen nun aber auch, daß Klienten im lokalen Netz den DNS-Server erreichen können. Hierzu müssen wir die Pakete aus dem lokalen Netz entgegennehmen und ins Internet weiterleiten:

5 Selbst wenn wir dies wollten, wäre es besser, dafür einen eigenen Rechner abzustellen und ihn wie einen Webserver in einer DMZ oder vor der Firewall zu betreiben.

6 Scans nach frei zugänglichen Proxies machen im Moment einen Großteil der beobachteten Port Scans aus. Warum dies so ist, ist nicht mit letzter Sicherheit geklärt, aber es steht zu befürchten, daß ein gewisser Anteil von Crackern herrührt, die eine Methode suchen, einen Webserver anzugreifen, ohne dabei die Adresse ihres eigenen Rechners zu verraten. Dies ist schade, da es durchaus auch ehrbare Gründe gibt, anonymisierende Proxies zu benutzen. Allerdings sollten dies dann Proxies sein, die von ihrem Eigentümer bewußt der Öffentlichkeit zur Verfügung gestellt wurden.

```
# - Forwarding durch die Firewall

iptables -A int-fw  -p udp -d 10.0.0.77 --dport 53 -j ACCEPT
iptables -A ext-fw  -p udp -s 10.0.0.77 --sport 53 -j ACCEPT

iptables -A int-fw  -p tcp -d 10.0.0.77 --dport 53 -j ACCEPT
iptables -A ext-fw  -p tcp -s 10.0.0.77 --sport 53 -j ACCEPT
```

Betreiben wir auf der Firewall einen eigenen DNS-Server, so sollten die Klienten nicht auf den externen DNS-Server, sondern auf den auf der Firewall zugreifen. In diesem Fall müssen wir die Filterregeln im Abschnitt »Forwarding durch die Firewall« durch die folgenden ersetzen:

```
# - DNS-Server auf der Firewall

iptables -A int-in  -p udp -d 192.168.20.15 --dport 53 -j ACCEPT
iptables -A int-out -p udp -s 192.168.20.15 --sport 53 -j ACCEPT
iptables -A int-in  -p tcp -d 192.168.20.15 --dport 53 -j ACCEPT
iptables -A int-out -p tcp -s 192.168.20.15 --sport 53 ! --syn \
-j ACCEPT
```

Einfache TCP-basierte Protokolle

Viele der bekannteren Netzwerkprotokolle bestehen aus Sicht der Paketfilterung aus einer einfachen Netzwerkverbindung von einem mehr oder weniger zufällig gewählten TCP-Port größer 1023 des Klienten an einen vordefinierten Port kleiner 1024 des Servers.

Da TCP verbindungsorientiert arbeitet, kann zwischen Anfragen und Antworten unterschieden werden. Verhindert man, daß Pakete mit CTL=SYN angenommen werden (Option `--syn`), so können keine Verbindungen von außen aufgebaut werden.

Daraus ergeben sich die folgenden Regeln, bei denen wieder zwischen dem Zugriff der Firewall selbst und den Regeln unterschieden wird, die für das Annehmen und die Weiterleitung von Paketen aus dem lokalen Netz nötig sind:

```
# - Zugriff der Firewall

iptables -A ext-in -p tcp --dport 1024:65535 --sport <Port> \
! --syn -j ACCEPT
iptables -A ext-out -p tcp --sport 1024:65535 --dport <Port> \
-j ACCEPT


# - Forwarding durch die Firewall

iptables -A int-fw -p tcp --sport 1024:65535 --dport <Port> \
-j ACCEPT
iptables -A ext-fw -p tcp --dport 1024:65535 --sport <Port> \
! --syn -j ACCEPT
```

Sie müssen nur noch für <*Port*> die Portnummer des betreffenden Protokolls einsetzen. Diese habe ich bei den im folgenden besprochenen Protokollen der Einfachheit halber jeweils am Anfang des Abschnitts angegeben.

HTTP

Port: 80

HTTP ist das grundlegende Protokoll des World Wide Web. Wann immer Sie mit dem Browser eine Adresse besuchen, die mit *http:* beginnt, kommt es zum Einsatz. Ihr Browser öffnet dabei eine TCP-Verbindung von einem »hohen« Port (> 1023) zu Port 80 eines fremden Rechners.

Dabei handelt es sich aber nur um unverschlüsselte Verbindungen. »Sichere« HTTP-Verbindungen über SSL/TLS (HTTPS) benutzen einen anderen Port. Da SSL bzw. TLS allerdings prinzipiell nicht auf HTTP beschränkt ist, habe ich diesem Protokoll ab Seite 329 einen eigenen Abschnitt gewidmet.

Schließlich existieren noch Server, die nicht den Standardport benutzen, sondern z. B. auf Ports wie 8000, 8008, 8080 oder 8888 Verbindungen entgegennehmen. Will man derartige Verbindungen erlauben, so sind zusätzliche Regeln nötig. Diese erhält man, indem man in obigen Regeln den Port 80 durch den jeweils gewünschten Port ersetzt.

Glücklicherweise ist das Problem aber nicht weit verbreitet. Die meisten Server halten sich an den Standard. Aus diesem Grund empfiehlt es sich abzuwarten, bis ein konkreter Bedarf besteht, einen bestimmten Port freizuschalten. Auch sollte man überlegen, ob man die Ausnahmen nicht auf bestimmte Rechner begrenzt, indem man zusätzlich zu

```
--sport Port auch -s Server und zu
--dport Port auch -d Server benutzt.
```

Erlaubt man schließlich das Protokoll FTP, so braucht man sich über Server auf hohen Ports keine Gedanken mehr zu machen. Diese Zugriffe sind dann sowieso erlaubt, um passives FTP zu ermöglichen (siehe Kapitel 12, Unterabschnitt *FTP*, ab Seite 330).

SMTP

Port: 25

Wollen Sie E-Mails versenden, so werden Sie, falls Sie nicht einen Server mit Web-Interface (z. B. Hotmail, GMX, Web.de . . .) benutzen, dies über das Simple Mail Transfer Protocol (SMTP) tun. Dieses Protokoll baut eine TCP-Verbindung von einem hohen Port des lokalen Rechners zu Port 25 des Mailservers auf.

Bei der Konfiguration der E-Mail-Clients sollten Sie aber darauf achten, daß diese den Mailserver Ihres Providers benutzen und eine korrekte Antwortadresse eingetragen haben. Zwar wäre es auch möglich, den Mailserver des Empfängers direkt anzusprechen, dies ist aber nicht unbedingt sinnvoll. So ist der Mailserver Ihres Providers in der Regel für Sie erreichbar[7]. Für den zuständigen Mailserver des Empfängers muß dies aber nicht gelten. Senden Sie die E-Mail über Ihren Provider, so wird sie dort zwischengelagert, während der Mailserver über einen längeren Zeitraum versucht, sie zuzustellen. Wollen Sie eine direkte Verbindung herstellen, so müssen Sie dies selbst wiederholen,

7 Andernfalls wäre sein Netz so überlastet, daß Ihre Chance, überhaupt einen Rechner im Internet zu erreichen, nur äußerst gering wäre.

bis sich ein Erfolg einstellt. Dies kostet nicht nur Zeit und Nerven, Sie müssen auch die Verbindungsgebühren zahlen.

POP3

Port: 110

Das Post Office Protocol Version 3 (POP3) ist gegenwärtig das gängigste Protokoll, um E-Mails von einem Mailserver abzuholen. Hierbei macht der Client eine TCP-Verbindung von einem hohen Port zu Port 110 des Mailservers auf.

Allerdings ist das POP3-Protokoll nicht ganz unproblematisch. Um seine E-Mails ausgehändigt zu bekommen, muß man sich in der Regel am Zielsystem authentisieren. Dazu werden standardmäßig ein Name und ein Paßwort benutzt. Beide werden im Klartext übertragen und können daher abgehört werden.

Zwar existieren Alternativen wie das Protokoll APOP, bei dem statt des Paßworts ein Challenge-Response-Verfahren verwendet wird[8], oder POP3S, bei dem die Verbindung mittels SSL gesichert wird (siehe Kapitel 12, Unterabschnitt *SSL und TLS*, ab Seite 329), diese werden aber nur von wenigen Servern und E-Mail-Clients unterstützt. Sollte Ihr Provider Ihnen allerdings anbieten, einen solchen Zugriff zu benutzen, so sollten Sie von diesem Angebot unbedingt Gebrauch machen.

Auch die eigentlichen E-Mails werden im Klartext übertragen, dies ist aber weniger ein Problem von POP3, sondern ein generelles Problem der Versendung von E-Mails. E-Mails sind grundsätzlich weniger gegen fremde Blicke geschützt als eine Postkarte, da selbst der Versand zu einem Bekannten im selben Ort im schlimmsten Fall über diverse Server auf der ganzen Welt erfolgt, wenn Sie nicht beim selben Provider sind. Auf der ganzen Strecke findet keine Verschlüsselung, ja nicht einmal ein Schutz vor Manipulation statt.

Wollen Sie daher in einer E-Mail irgend etwas sagen, was nicht auch in einer überregionalen Tageszeitung unter Ihrem Namen erscheinen dürfte, sollten Sie ein gutes Verschlüsselungsprodukt verwenden. Pretty Good Privacy (PGP)[9] oder GNU Privacy Guard (GPG)[10] sind hier sicherlich einer näheren Betrachtung wert.

IMAP

Port: 143

Das Internet Message Access Protocol (IMAP) dient wie POP3 dem Herunterladen der eigenen E-Mails von einem zentralen Mailserver. Es ist jünger als POP3 und bietet deutlich mehr Möglichkeiten, hat sich aber noch nicht im gleichen Maße durchgesetzt. Aus Sicht des Firewalling beschränken sich die Unterschiede zwischen den Protokollen auf die Tatsache, daß IMAP den Port 143 benutzt.

8 Hier sind keine Änderungen der Filterregeln erforderlich, da es sich nur um eine Protokollerweiterung handelt, nicht aber um ein eigenständiges Protokoll.

9 Eine Version für den privaten Gebrauch kann von *http://www.pgp.com* heruntergeladen werden. Die kommerzielle Nutzung der Software erfordert den Erwerb einer Lizenz.

10 Erhältlich unter *http://www.gnupg.org*.

Was die Sicherheit des Protokolls angeht, bietet sich hier das gleiche Bild wie bei POP3. Auch hier werden Name und Paßwort unverschlüsselt gesendet. Zwar existiert auch eine Möglichkeit, Einweg-Paßwörter zu benutzen, Sie sollten aber bis zum Beweis des Gegenteils nicht davon ausgehen, daß dies sowohl von Ihrem E-Mail-Client als auch von Ihrem Mailserver unterstützt und genutzt wird. Eine andere Möglichkeit, Ihre Verbindung zum Mailserver zu sichern, besteht im Einsatz von SSL/TLS. Sollte das in Ihrem Fall möglich sein, so sollten Sie die Gelegenheit unbedingt nutzen.

NNTP

Port: 119

Das Network News Transfer Protocol (NNTP) dient zum Transport von Usenet News. Hierbei handelt es sich um ein System, bei dem man Nachrichten schreibt, die dann in sogenannte Newsgroups gestellt werden, wo sie von jedermann gelesen und gegebenenfalls kommentiert und beantwortet werden können. Die Summe aller Server bildet das Usenet. Die Server sind dabei alle miteinander vernetzt. Erhält ein Server eine neue Nachricht, so wird er sie seinen Nachbarservern anbieten. Auf diese Weise entstehen Diskussionsforen, die es ermöglichen, daß Benutzer auf der ganzen Welt miteinander kommunizieren.

Zwar hat die Bedeutung des Usenet mit dem Aufkommen des World Wide Web etwas abgenommen, ein Posting in die richtige Newsgroup kann aber immer noch die letzte Hoffnung sein, wenn man z. B. ein schwieriges Problem mit einer Software hat und der Hersteller nicht bereit ist, einem zu helfen. Unter den oft Tausenden von Lesern weltweit findet sich dann schon einmal der eine oder andere, der dasselbe Problem hatte und die Lösung kennt. Aber auch sonst kann das Usenet eine angenehme Möglichkeit sein, sich mit Gleichgesinnten auszutauschen.

Bei NNTP wird eine TCP-Verbindung von einem hohen Port des Klienten zu Port 119 des Newsservers benutzt.

Gopher und WAIS

Ports: 70 (Gopher), 210 (WAIS)

Diese beiden Protokolle sind hier eigentlich nur der Vollständigkeit halber aufgeführt. Obwohl sie früher recht beliebt waren, werden sie heute kaum noch eingesetzt.

Gopher ist ein menübasiertes textorientiertes Protokoll, das es schon vor dem WWW erlaubte, verschiedene Rechner in einer Art Hypertext zu verbinden. Startete man eine Gopher-Sitzung, so erschien auf dem Bildschirm ein Menü, dessen Menüpunkte wieder zu weiteren Menüs oder zu Dateien führten. Führte ein Menüpunkt zu einem anderen Menü, so mußte dieses nicht auf demselben Server liegen. Genau wie im WWW konnte man sich bei Gopher durch Links um die ganze Welt bewegen. Sollten Sie dies einmal ausprobieren wollen oder kennen Sie noch Gopher-Server, die Sie benutzen möchten, so müssen Sie Port 70 freischalten.

Das WAIS-Protokoll erlaubt es, komplizierte Datenbankabfragen zu stellen, und wurde früher gerne eingesetzt, um z. B. Bibliothekskataloge abzufragen. Seitdem es aber Suchmaschinen im WWW gibt, wird dieses Protokoll kaum noch eingesetzt. Sollten Sie aber noch einen WAIS-Server kennen, den Sie benutzen möchten, so sollten Sie Port 210 freischalten.

Prinzipiell kann es sowohl bei Gopher als auch bei WAIS vorkommen, daß auch andere Ports verwendet werden. In diesem Falle beachten Sie bitte die Ausführungen zu HTTP ab Seite 326.

SSL und TLS

Ports: z. B. 443 (HTTPS), 995 (POP3S), 993 (IMAPS), 563 (NNTPS), 636 (LDAPS), 992 (Telnets)

Wenn Webseiten sicher übertragen werden sollen (Bankgeschäfte, Übertragung von Kreditkartendaten bei Online-Einkäufen . . .), kommt in der Regel HTTPS zum Einsatz. Dabei wird ein Protokoll eingesetzt, das von Netscape entwickelt und Secure Socket Layer (SSL) getauft wurde. Als es dann zu einem internationalen Standard wurde, hat man es in Transport Layer Security (TLS) umbenannt.

Dieses Protokoll ist zwischen dem Anwendungsprotokoll (hier: HTTP) und TCP/IP angesiedelt und dient dazu, beliebigen Protokollen Mechanismen zur Verschlüsselung und Authentisierung zur Verfügung zu stellen. Genauso wie TCP höherliegende Anwendungen davon befreit, überprüfen zu müssen, ob alle Pakete genau einmal und in der richtigen Reihenfolge angekommen sind, befreit SSL/TLS die Anwendung davon, selbst eine Methode zur kryptographischen Sicherung einer Verbindung mit der Gegenstelle auszuhandeln.

Will man nun SSL benutzen, so darf man die Verbindung nicht direkt zum normalen Port des Anwendungsprotokolls herstellen (hier: 80), da dort Anfragen in dem eigentlichen Anwendungsprotokoll erwartet würden und der Server nicht darauf gefaßt ist, SSL/TLS-Pakete entgegenzunehmen. Statt dessen sind für die meisten Protokolle spezielle Ports reserviert, die für die gesicherte Variante des Protokolls benutzt werden sollen. Dies sind z. B. 443 für HTTPS, 563 für NNTPS, 636 für LDAPS, 989 für die Datenverbindung von FTPS (siehe Kapitel 12, Unterabschnitt *FTP*, ab Seite 330), 990 für die Kontrollverbindung von FTPS, 992 für TELNETS, 993 für IMAPS, 994 für IRCS, 995 für POP3S.

Für die meisten Protokolle wird SSL/TLS allerdings nur in ganz geringem Maße eingesetzt. Man kann sich daher mit dem Einrichten spezieller Regeln Zeit lassen, bis man tatsächlich einen Server findet, der es auch unterstützt. Eine besondere Ausnahme stellt allerdings HTTPS dar. HTTPS ist schon deutlich weiter verbreitet und sollte daher auch von den Filterregeln erlaubt werden.

Auch für die Protokolle POP3 und IMAP ist eine Sicherung der Verbindung sinnvoll, schließlich muß man beim Abholen seiner E-Mail normalerweise Name und Paßwort angeben. Diese werden in den Protokollen aber standardmäßig im Klartext übertragen und können durch Sniffer ausgespäht werden. Bedauerlicherweise unterstützen weder

alle E-Mail-Provider noch alle E-Mail-Programme den Gebrauch von SSL. Gehören Sie zu den Glücklichen, die eine Ausnahme von dieser Regel darstellen, sollten Sie diese Protokolle unbedingt freischalten.

FTP

Ohne Stateful Packet Filtering ist FTP aus der Sicht des Firewalling ein wahrer Alptraum.

Das Protokoll basiert auf TCP und baut zunächst eine Kontrollverbindung vom lokalen Rechner zu Port 21 des FTP-Servers auf. Sollen nun Daten übertragen werden, so wird dafür eine eigene Verbindung geöffnet.

Beim *aktiven FTP* tut dies der FTP-Server. Wir haben also hier die Situation, daß ein Rechner im Internet eine Verbindung zu unserer Firewall aufbaut! Dies widerspricht allem, was wir bisher bei der Einrichtung beachtet haben. Wir müssen dabei darauf achten, daß die Verbindungen von Port 20 kommen und an einen hohen Port gerichtet sind. Hierbei ist es besonders wichtig, daß sich auf den hohen Ports keine Server befinden, die nicht durch spezielle Regeln gegen den Zugriff abgeschirmt wurden.

Die Alternative besteht im *passiven FTP*, bei dem der Klient eine weitere Verbindung öffnet. Dazu teilt der Server dem Klienten vorher eine Portnummer mit, auf der er auf den Verbindungsaufbau wartet. Hier ist die Datenverbindung nun eine ausgehende Verbindung von einem Port > 1024 zu einem Port > 1024. Dies ist zwar ebenfalls schwierig zu filtern, da beide Portnummern unbekannt sind, immerhin wird aber keine Verbindung mehr zur Firewall geöffnet. Leider wird es nicht von allen FTP-Servern unterstützt.

Hier nun die notwendigen Regeln:

```
# FTP

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

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

# - - Aktives FTP

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

# - - Passives FTP

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

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

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

# - - Aktives FTP

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


# - - Passives FTP

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

Proxies auf der Firewall

Wollen wir den Zugriff für ein bestimmtes Protokoll über einen Proxy realisieren, so brauchen wir die gleichen Regeln, um von der Firewall aus auf den Zielrechner zuzugreifen, wie wir sie auch ohne Proxies verwenden. Die Regeln im Abschnitt »Zugriff auf den Server« bleiben also gleich. Der Proxy agiert hier wie ein normaler Klient, der von einem Benutzer auf der Firewall gestartet wurde.

Die Regeln im Abschnitt »Forwarding durch die Firewall« entfallen dagegen und werden durch neue ersetzt, die nur den Zugriff auf den jeweiligen Proxy auf der Firewall erlauben. Statt in `int-fw` und `ext-fw` werden die Regeln nun in `int-in` und `int-out` definiert, da es sich um Zugriffe auf die Firewall selber handelt.

Einfache TCP-basierte Protokolle

Wir wollen die nötigen Regeln nun einmal am Beispiel des Webproxys `squid` betrachten. Dieser benutzt den Port 3128. Für andere Proxies müssen Sie natürlich den verwendeten Port anpassen:

```
# - Proxy auf Port 3128

iptables -A int-in -p tcp --sport 1024:65535 -d 192.168.20.15 \
--dport 3128 -j ACCEPT
iptables -A int-out -p tcp --dport 1024:65535 -s 192.168.20.15 \
--sport 3128 ! --syn -j ACCEPT
```

Webbrowser verwenden grundsätzlich HTTP, um den Proxy anzusprechen, obwohl der Zielrechner eigentlich ein anderes Protokoll (z. B. Gopher oder FTP) benutzt. Wenn der Proxy entsprechende Protokolle unterstützt, wird er gegenüber dem Browser als Webserver, gegenüber dem Zielrechner aber als Client für das jeweilige Protokoll auftreten.

Wollen wir also auf bestimmte Protokolle ausschließlich mit einem Webbrowser zugreifen, so reicht uns für das Protokoll nur der Bereich »Zugriff auf den Server«. Die Regeln für eine Weiterleitung von direkten Verbindungen entfallen, und der Zugriff der Klienten aus dem lokalen Netz ist bereits für HTTP-Zugriffe erlaubt worden.

Hier ein Beispiel für Gopher:

```
# Gopher
# - Zugriff auf den Server

iptables -A ext-in  -p TCP --sport 70 --dport 1024:65535 \
! --syn -j ACCEPT
iptables -A ext-out -p TCP --dport 70 --sport 1024:65535 \
-j ACCEPT
```

DNS

Betreibt man auf der Firewall einen DNS-Server als Proxy für das lokale Netz, so gilt das bereits Gesagte. Hier muß man allerdings beachten, daß DNS neben TCP auch UDP benutzt:

```
# - - Server auf der Firewall

iptables -A int-in  -p udp -d 192.168.20.15 --dport 53 -j ACCEPT
iptables -A int-out -p udp -s 192.168.20.15 --sport 53 -j ACCEPT
iptables -A int-in  -p tcp -d 192.168.20.15 --dport 53 -j ACCEPT
iptables -A int-out -p tcp -s 192.168.20.15 --sport 53 ! --syn -j ACCEPT
```

FTP

Für FTP werden ähnliche Regeln verwendet. Hier muß man allerdings beachten, daß neben der Kontrollverbindung auch eine Datenverbindung geöffnet wird. Dadurch sind sechs statt der bisherigen zwei Regeln nötig:

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

iptables -A int-in  -p tcp --sport 1024:65535 -d 192.168.20.15 --dport 21 \
-j ACCEPT
iptables -A int-out -p tcp --dport 1024:65535 -s 192.168.20.15 --sport 21 \
! --syn -j ACCEPT

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

iptables -A int-in  -p tcp --sport 1024:65535 -d 192.168.20.15 --dport 20 \
! --syn -j ACCEPT
iptables -A int-out -p tcp --dport 1024:65535 -s 192.168.20.15 --sport 20 \
-j ACCEPT
```

```
# - - Passives FTP

iptables -A int-in  -p tcp --sport 1024:65535 -d 192.168.20.15 --dport 1024:65535 \
-j ACCEPT
iptables -A int-out -p tcp --dport 1024:65535 -s 192.168.20.15 --sport 1024:65535 \
! --syn -j ACCEPT
```

Falls Sie den `ftp-gw` von SuSE verwenden, müssen Sie noch beachten, daß Sie hier als Quellport für Datenverbindungen unter bestimmten Umständen nicht Port 20 verwenden können. Dies ist dann der Fall, wenn Sie den Proxy in einem `chroot`-Käfig verwenden wollen. In so einer Konfiguration kommen nur Ports oberhalb von 1023 in Frage. Wir wollen hier 2020 verwenden:

```
# - - Aktives FTP
# - - - SuSE ftp-proxy im chroot

iptables -A int-in  -p tcp --sport 1024:65535 -d 192.168.20.15 --dport 2020 \
! --syn -j ACCEPT
iptables -A int-out -p tcp --dport 1024:65535 -s 192.168.20.15 --sport 2020 \
-j ACCEPT
```

Die Regeln im Abschnitt »Forwarding durch die Firewall« entfallen wie gehabt.

Transparente Proxies

In einigen Fällen kann es sinnvoll sein, einen Benutzer, der einen direkten Zugriff auf einen Server im Internet versucht, auf einen Proxy »umzuleiten«. Ich habe beispielsweise einmal eine Firewall betreut, auf der E-Mails nur über den Mailserver der Firewall versandt werden durften. Dies stellte ein Problem für einige Benutzer dar, die ihren Laptop von zu Hause mitbrachten und dort normalerweise direkt den Zielrechner kontaktierten, bei uns aber jedesmal den lokalen Mailserver einstellen mußten. Um ihnen das ständige Umkonfigurieren zu ersparen, erstellte ich die nötigen Regeln auf der Firewall, damit ihre Pakete automatisch bei unserem Mailserver landeten und nicht wie zuvor verworfen wurden.

Dies ist für ein Protokoll wie SMTP relativ problemlos, da hier die Zieladresse im Protokoll übermittelt wird. Es gehört zur Klasse der *Store and Forward-Protokolle*, die davon ausgehen, daß die übertragenen Nachrichten bei Bedarf auch über mehrere Zwischenrechner gesendet werden können, wo sie jeweils abgelegt (»Store«) und zu einem Zeitpunkt, an dem wo die Bedingungen günstig sind[11], weitergeleitet werden (»Forward«). Für den Proxy ist es damit kein Problem, zu entscheiden, wohin er eine E-Mail weiterleiten muß.

In die gleiche Klasse gehört z. B. auch NNTP, wo im Prinzip jede Nachricht an alle Server weitergeleitet wird. Zwar ist in der Praxis die Datenflut so groß, daß nicht jeder Newsserver auch wirklich alle Newsgroups vorhält. Man wird aber in der Regel dem Proxy trotzdem einen festen Server vorgeben, an den alle Anfragen weitergeleitet werden.

11 Ein günstiger Telefontarif, der Zielrechner hat sich in das Netz eingewählt und ist bereit, die Daten zu übernehmen ...

Andere Protokolle sind dagegen für das Betreiben eines transparenten Proxys deutlich problematischer. Hierzu gehören vor allem Protokolle, bei denen sich ein Benutzer gezielt an einem System anmeldet, im Protokoll aber nicht der Name des Zielsystems übertragen wird, da man davon ausgeht, daß alle Verbindungen direkt hergestellt werden. Beispiele hierfür sind FTP, POP3 und IMAP.

Klassische Proxies wie zum Beispiel der `ftp-gw` aus dem Firewalling Toolkit wissen hier nicht, für welches Zielsystem die Verbindung bestimmt ist, weswegen der gewünschte Server in vorhandenen Feldern untergebracht werden muß. Bei manchen FTP-Proxies gibt man daher z. B. als Benutzername »Kennung@Zielsystem« an. Dies verhindert natürlich jeden Versuch, den Benutzer auf einen Proxy umzuleiten, ohne daß dieser etwas davon mitbekommt.

Moderne Proxies wie der `ftp-proxy` von SuSE arbeiten hier mit der Network Address Translation zusammen, wodurch sie in der Lage sind, die eigentliche Zieladresse herauszufinden. Mit ihnen stellt transparentes Proxying kein Problem mehr dar.

HTTP stellt schließlich einen Mittelweg zwischen diesen Extremen dar. Obwohl das Protokoll eigentlich auch direkte Verbindungen benutzt, wenn dem Browser nicht explizit mitgeteilt wurde, daß ein Proxy benutzt werden soll, so senden neuere Browser einen zusätzlichen Header »Host:«, in dem der Zielrechner angegeben ist. Wertet der Proxy diesen Header aus, so kann er auch dann eine Seite anfordern, wenn in der eigentlichen URL kein Zielrechner angegeben ist. Nicht jeder Webproxy ist allerdings dazu in der Lage.

Wollen wir einen transparenten Proxy einrichten, so bekommen wir es nun zum ersten Mal mit der Network Address Translation (Table `nat`) zu tun. Wir richten in der Chain `PREROUTING` eine Regel ein, die greift, bevor die Routing-Entscheidung getroffen wird. In dieser ändern wir die Zieladresse und den Zielport, so daß das Paket wie eine Proxy-Anfrage für den Rechner selbst aussieht und durch das nachfolgende Firewalling auch so behandelt wird. Dadurch sind keine zusätzlichen Regeln im eigentlichen Firewalling nötig. Dort genügen die normalen Proxyregeln.

Für die Umleitung von HTTP-Zugriffen auf einen `squid` genügt es z. B., die folgende Regel zu definieren:

```
# Transparente Umleitung von HTTP-Zugriffen auf den Proxy auf
# Port 3128

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
-j REDIRECT --to-port 3128
```

Für DNS ist das Vorgehen ähnlich, man muß allerdings beachten, daß hier sowohl TCP als auch UDP verwendet wird:

```
# - Umleitung von DNS-Serverzugriffen auf den lokalen Server

iptables -t nat -A PREROUTING -i eth0 -p udp --dport 53 \
-j REDIRECT --to-port 53
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 53 \
-j REDIRECT --to-port 53
```

Auch FTP macht diesmal keine Ausnahme. Lediglich die Kontrollverbindung muß umgeleitet werden:

```
# - Umleitung von FTP-Serverzugriffen (TCP 21) auf den
#   lokalen ftp-proxy

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 21 \
-j REDIRECT --to-port 21
```

In einzelnen Fällen ist es allerdings vorgekommen, daß die transparente Umleitung mit `REDIRECT` nicht gelang, obwohl der direkte Zugriff auf den Proxy problemlos funktionierte. In diesen Fällen kann es helfen, wenn man statt dessen `DNAT` verwendet und explizit als Zieladresse die IP-Adresse des internen Interfaces vorgibt.

Für obiges Beispiel einer Umleitung auf den `squid` würde das folgendermaßen aussehen:

```
# Transparente Umleitung von HTTP-Zugriffen auf den Proxy auf
# Port 3128

iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
-j DNAT --to-destination 192.168.20.15:3128
```

Man kann `DNAT` auch benutzen, um die Anfrage auf einen anderen Rechner umzuleiten. Dies kann z. B. sinnvoll sein, wenn man einen Webproxy in einer DMZ betreibt, statt ihn auf der Firewall selbst laufen zu lassen. Ich verwende den Mechanismus auch, um zentral festzulegen, welcher DNS-Server von meinen Klienten benutzt wird. So brauche ich bei einem Providerwechsel nur die Firewall umzukonfigurieren und erspare es mir, die Klienten einzeln anzupassen.

Einen Spezialfall stellt hier allerdings FTP dar. Für dieses Protokoll sollten Sie keine transparente Umleitung auf einen anderen Rechner probieren. Hier hätte der Proxy sonst keine Chance herauszufinden, welchen Rechner Sie eigentlich erreichen wollten.

Logging ungewöhnlicher Pakete

Wird für ein Paket keine passende Regel gefunden, so treten schließlich die Policies der Chain in Kraft. Diese wurden von uns so gewählt, daß alle unbekannten Pakete ohne Rückmeldung an den Sender entsorgt werden. Auf diese Weise können viele Angriffe abgewehrt werden. Wir erhalten aber keinen Eintrag im Systemlog, der uns auf sie aufmerksam macht, da dies nur für Regeln, nicht aber für die Policy definiert werden kann.

Aus diesem Grund ist es sinnvoll, für jede Chain eine letzte Regel zu definieren, die diesen Mangel behebt:

```
# Pakete, die nicht von den normalen Regeln abgedeckt werden

iptables -A INPUT   -s 0.0.0.0/0 -j LOG \
--log-prefix "INPUT (default): "
iptables -A OUTPUT  -s 0.0.0.0/0 -j LOG \
--log-prefix ÖUTPUT (default): "
iptables -A FORWARD -s 0.0.0.0/0 -j LOG \
--log-prefix "FORWARD (default): "
iptables -A int-in  -s 0.0.0.0/0 -j LOG \
--log-prefix "int-in (default): "
iptables -A int-out -s 0.0.0.0/0 -j LOG \
--log-prefix "int-out (default): "
iptables -A int-fw  -s 0.0.0.0/0 -j LOG \
--log-prefix "int-fw (default): "
iptables -A ext-in  -s 0.0.0.0/0 -j LOG \
--log-prefix "ext-in (default): "
iptables -A ext-fw  -s 0.0.0.0/0 -j LOG \
--log-prefix "ext-fw (default): "
iptables -A ext-out -s 0.0.0.0/0 -j LOG \
--log-prefix "ext-out (default): "

iptables -A INPUT   -s 0.0.0.0/0 -j DROP
iptables -A OUTPUT  -s 0.0.0.0/0 -j DROP
iptables -A FORWARD -s 0.0.0.0/0 -j DROP
iptables -A int-in  -s 0.0.0.0/0 -j DROP
iptables -A int-out -s 0.0.0.0/0 -j DROP
iptables -A int-fw  -s 0.0.0.0/0 -j DROP
iptables -A ext-in  -s 0.0.0.0/0 -j DROP
iptables -A ext-fw  -s 0.0.0.0/0 -j DROP
iptables -A ext-out -s 0.0.0.0/0 -j DROP
```

Vielleicht ist Ihnen aufgefallen, daß hier die Chain `states` fehlt. Diese Chain ist ein Spezialfall. Sie dient dazu, bestimmte Pakete zu verwerfen. Pakete, auf die keine Regel paßt, werden in der aufrufenden Chain weiter bearbeitet. Hier würde das beschriebene Vorgehen dazu führen, daß die meisten legitimen Pakete verworfen werden.

Masquerading

Im Firewalling haben wir die Rechner im lokalen Netz so behandelt, als ob sie gültige Internet-Adressen hätten. Dies ist in unserem Beispiel aber gar nicht der Fall. Alle Rechner im lokalen Netz haben private Adressen, die im Internet nicht geroutet werden. Der einzige Rechner mit einer gültigen Adresse ist die Firewall selbst.

Es bietet sich daher an, mit Network Address Translation (Table `nat`) die Absenderadresse in den Paketen so zu verändern, daß sie von der Firewall selbst zu stammen scheinen. Idealerweise geschieht dies nach dem Firewalling (Chain `POSTROUTING`), das so nicht durch den Vorgang beeinflußt wird, sondern die Pakete als ganz normale Pakete des lokalen Netzes erkennt.

Hier die Regel:

```
# Alle Pakete des lokalen Netzes maskieren

iptables -t nat -A POSTROUTING -o "$EXTIF" -j MASQUERADE
```

Regeln in Systemdateien eintragen

Die folgenden Dateien bieten sich für den Eintrag von Firewallregeln an:

- ein selbsterstelltes Runlevel-Skript
- `/etc/ppp/ip-up`
- `/etc/ppp/ip-down`

Ein Runlevel-Skript sollte die Regeln enthalten, die schon gelten sollen, bevor eine Verbindung zum Internet besteht. Wichtig ist dabei, daß das Skript schon vor dem Einrichten der Netzwerk-Interfaces gestartet bzw. erst nach dem Herunterfahren der Interfaces gestoppt wird. Unter SuSE-Linux können wir dies über entsprechende Kommentare erreichen. Das Skript installieren wir dann einfach mit `insserv`.

Hier ein Beispielskript, das eine einfache Firewall realisiert. Regeln für Webproxies auf Port 3128, 8000, 8080 und 8118, einen FTP-Proxy und einen DNS-Server auf der Firewall sind im Skript vorhanden, aber auskommentiert. Soll ein Protokoll statt durch Masquerading durch einen Proxy geschützt werden, so muß nur das Kommentarzeichen »#« aus der betreffenden Zeile entfernt und der Abschnitt »Maskieren durch die Firewall« für das betreffende Protokoll auskommentiert werden.

Das gleiche gilt auch für die transparente Umleitung von Web-, FTP- und DNS-Zugriffen auf einen Proxy der Firewall. Auch hier ist der nötige Befehl vorhanden, aber auskommentiert. Auch abgelehnte Ident-Anfragen werden nur protokolliert, wenn Sie das entsprechende Kommentarzeichen entfernen.

Am Anfang des Skripts stehen mehrere Variablen, die noch an die konkrete Anwendungssituation angepaßt werden müssen:

EXTIP Die Adresse unseres externen Interfaces.

EXTIF Der Name des externen Interfaces. Übliche Namen sind `ppp0` (Modem) und `ippp0` (ISDN).

INTIP Die Adresse der Firewall im lokalen Netz.

INTIF Der Name des internen Interfaces. Üblicherweise handelt es sich um ein Ethernet-Interface, z. B. `eth0`.

INTBITS Eine andere Art, eine Netzwerkmaske anzugeben. Hat Ihr internes Interface z. B. die Adresse *192.168.20.15* und es sollen im LAN alle Adressen der Art *192.x.x.x* möglich sein, so interessieren uns nur die ersten 8 Bit einer Adresse, wenn wir entscheiden wollen, ob ein Rechner sich im selben Subnetz wie die Firewall befindet. Wir geben dann also eine `8` an. Für einen Adreßbereich *192.168.x.x* nehmen wir dementsprechend `16`, und für *192.168.20.x* ist der zugehörige Wert `24`.

TCPPROTECTED Die Nummern der TCP-Ports über 1023, auf denen Server aktiv sind, auf die man nicht aus dem Internet zugreifen können soll (z. B. Proxies).

UDPPROTECTED Die Nummern der UDP-Ports über 1023, auf denen Server aktiv sind, die nicht aus dem Internet zu erreichen sein sollen (z. B. Proxies).

Zwei weitere Variablen werden automatisch gefüllt. Sie brauchen sie daher nicht zu ändern:

INTNET Der Adreßbereich, der im lokalen Netz verwendet wird. Diesen Wert brauchen Sie nicht zu ändern, er wird aus *INTIP* und *INTBITS* berechnet.

DNSSERVER Die IP-Adressen der DNS-Server. Die Adressen werden aus der Datei */etc/resolv.conf* ausgelesen. Sie brauchen sie daher hier nicht noch ein zweites Mal anzugeben.

Hier nun das Skript:

```
#!/bin/sh
###########################################################################
#
# pfilter.iptables
#
#     Filterregeln für IP-Pakete (iptables-Version)
#
# Usage: pfilter {start|stop}
#
# Copyright (C) 2003 Andreas 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:          pf-ipt
# Required-Start:    $local_fs
# Should-Start:
# Required-Stop:     procconf
# Should-Stop:
# Default-Start:     2 3 5
# Default-Stop:      0 1 6
# Short-Description: Firewalling
# Description:       Firewalling
### END INIT INFO
```

```
# ------------------------------------------------------------------------
# Grundeinstellungen
# ------------------------------------------------------------------------

# Eine Kurzbezeichnung für iptables

# Warnfarbe

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


# Eine Kurzbezeichnung für 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 grundsätzliche Daten

# - Externes Interface

EXTIF="ppp0"

# - Internes Interface

INTIP="192.168.20.15"
INTIF="eth0"
INTBITS=24
INTNET="$INTIP"/"$INTBITS"

# - 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'
```

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

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


# ------------------------------------------------------------------------
# Regeln, die immer gelten
# ------------------------------------------------------------------------

# Alle Regeln und selbstdefinierten Chains löschen

$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

$R -A FORWARD -i ! "$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
```

```
# 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


# Unaufgeforderte Verbindungsaufbauten (NEW) und ungültige Pakete
# (INVALID) des externen Interfaces.

# - Eine eigene Chain für 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

$R -A states -m state --state NEW,INVALID -j LOG \
--log-prefix Ünerwuenschte 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 über 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 ungewöhnlicher 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 ÖUTPUT (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)

# 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

# - Verteilung der Pakete auf die Chains

$R -A INPUT   -i "$INTIF" -s "$INTNET" -j int-in
$R -A INPUT   -i "$EXTIF" -j ext-in
$R -A FORWARD -i "$INTIF" -o "$EXTIF" -s "$INTNET" -j int-fw
$R -A FORWARD -i "$EXTIF" -o "$INTIF" -j ext-fw
$R -A OUTPUT  -o "$INTIF" -j int-out
$R -A OUTPUT  -o "$EXTIF" -j ext-out
```

```
# 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

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

# - Forwarding durch die Firewall

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


# HTTP-Proxies

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

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

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

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


# HTTPS
# - Zugriff auf den Server

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

# - Forwarding durch die Firewall

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


# SMTP
# - Zugriff auf den Server

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

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

# - Forwarding durch die Firewall

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


# Gopher
# - Zugriff auf den Server

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

# - Forwarding durch die Firewall

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


# WAIS
# - Zugriff auf den Server

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

# - Forwarding durch die Firewall

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


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

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

# - - Aktives FTP

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

# - - Passives FTP

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



# Protokollierung ungewöhnlicher Pakete

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

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



# Network Address Translation
# - Masquerading

$R -t nat -A POSTROUTING -o "$EXTIF" -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
```

Regeln, die sich auf die Adresse des externen Interfaces beziehen, können unter Umständen erst später definiert werden, wenn uns diese Adresse vom Provider zugewiesen wurde. Hierbei bieten sich die Skripte `/etc/ppp/ip-up` und `/etc/ppp/ip-down` an. Sind diese schon vorhanden, so sollten wir die vorhandenen Dateien umbenennen und unsere eigenen erzeugen[12]:

```
# cd /etc/ppp
# mv ip-up ip-up.orig
# mv ip-down ip-down.orig
# echo '#!/bin/sh' > ip-up
# echo '#!/bin/sh' > ip-down
# chmod 700 ip-up ip-down
```

In der Datei `ip-up` können wir all jene Regeln eintragen, die erst definiert werden können, wenn eine Verbindung besteht. Ein Beispiel dafür wäre die Anti-Spoofing-Regel für das externe Interface:

```
EXTIP=$4

# gespoofte Pakete der Firewall
iptables -I INPUT 1 -i !lo -s $EXTIP -j DENY -l
iptables -I FORWARD 2 -i !lo -s $EXTIP -j DENY -l
```

`ip-down` sollte das System dagegen wieder in den Zustand bringen, der für den Fall gewünscht wird, in dem keine Internetverbindung besteht. Insbesondere sollte dabei die Regel mit der nunmehr ungültigen Adresse gelöscht werden:

```
EXTIP=$4

# gespoofte Pakete der Firewall
iptables -D INPUT -i !lo -s $EXTIP -j DENY -l
iptables -D FORWARD -i !lo -s $EXTIP -j DENY -l
```

12 Dies gilt natürlich nicht, wenn wir die Dateien selbst erzeugt haben (z. B. für ISDN).