Was ist ein Paketfilter?[1]

Ein Paketfilter ist ein Stück Software, das sich die Header von passierenden Paketen ansieht und über das Schicksal des vollständigen Pakets entscheidet. Er könnte entscheiden, das Paket zu DROPPEN (ich meine das Paket zu verwerfen, als wäre es niemals empfangen worden), es zu akzeptieren (ACCEPT, ich meine das Paket durchzulassen), oder etwas Komplizierteres. Unter Linux ist Paketfiltern im Kernel selbst enthalten (als ein Kernelmodul oder direkt eingebaut), und es gibt noch ein paar trickreichere Dinge, die wir mit Paketen anstellen können, aber das generelle Prinzip vom Ansehen der Header und über das Schicksal der Pakete Entscheiden ist immer noch da.

Warum sollte ich einen Paketfilter wollen?

Kontrolle. Sicherheit. Wachsamkeit.

Kontrolle:
Wenn du einen Linuxrechner benutzt, um dich aus Deinem internen Netzwerk mit einem anderen Netzwerk (wie dem Internet) zu verbinden, hast du die Möglichkeit, bestimmte Arten von Traffic zu erlauben, und andere zu verbieten. Zum Beispiel enthält der Header eines Pakets die Zieladresse des Pakets, also kannst du verhindern, dass Pakete zu einem bestimmten Teil des äußeren Netzwerks gehen. Ein anderes Beispiel: Ich benutze Netscape, um die Dilbert-Archive zu besuchen. Es gibt Werbung von doubleclick.net auf der Seite, und Netscape verschwendet meine Zeit, um sie alle fröhlich herunterzuladen. Man kann das Problem lösen, indem man den Paketfilter beauftragt, keine Pakete von oder zu Adressen, die doubleclick.net gehören, zuzulassen (es gibt hierfür auch bessere Wege: siehe Junkbuster).

Sicherheit:
Wenn dein Linuxrechner das einzige zwischen dem Chaos des Internet und deinem netten, ordentlichen Netzwerk ist, ist es schön, zu wissen, dass du einschränken kannst, was für Dinge durch deine Türe kommen. Zum Beispiel kannst du alles erlauben, was aus deinem Netzwerk rausgeht, aber du könntest besorgt sein über den wohlbekannten 'Ping of Death', der von bösen Außenstehenden hereinkommen könnte. Als ein anderes Beispiel möchtest du vielleicht nicht, dass Fremde zu deinem Linuxrechner telnetten können, obwohl all deine Accounts Passwörter haben. Vielleicht möchtest du auch (wie die meisten) im Internet eher ein Beobachter sein, als jemand, der Dienste (gewollt oder nicht) anbietet. Erlaube einfach niemandem, eine Verbindung zu dir aufzubauen, indem der Paketfilter eingehende Pakete, die eine Verbindung aufbauen wollen, verwirft.

Wachsamkeit:
Manchmal könnte eine schlecht konfigurierte Maschine im lokalen Netz entscheiden, Pakete regelrecht in die Außenwelt zu spucken. Es ist nett, wenn man dem Paketfilter sagen kann, dass er dir melden soll, sobald etwas Abnormales vorfällt; vielleicht kannst du dann etwas daran ändern, oder vielleicht bist du bloß von Natur aus neugierig.

[1] Quelle: **http://www.netfilter.org** Linux 2.4 Packet Filtering HOWTO von Rusty Russell, ins Deutsche übersetzt von Melanie Berg, bearbeitet von Ivo Plohnke.

Wie filtere ich Pakete unter Linux?

1999 gab es für Linux 2.4 eine komplette Überarbeitung des Kernels und somit das Tool für die vierte Generation: 'iptables'. Es ist dieses iptables, auf das sich dieses HOWTO konzentriert. Du brauchst einen Kernel mit der Netfilter-Infrastruktur: Netfilter ist ein genereller Rahmen im Linuxkernel, auf dem andere Dinge (wie die iptables Module) aufbauen können. Das bedeutet, dass du Kernel 2.3.15 oder höher brauchst. Das Tool iptables spricht mit dem Kernel und sagt ihm, welche Pakete zu filtern sind. Wenn du kein Programmierer und auch nicht überaus neugierig bist, ist das der Weg, auf dem du den Paketfilter kontrollieren wirst.

iptables

Das iptables Tool fügt Regeln in die Filtertabellen des Kernels ein und löscht andere. Das bedeutet, dass die Regeln, wie immer du sie aufsetzt, beim Neustart des Rechners verloren sein werden. Lies „Regeln dauerhaft erstellen", um sicherzugehen, dass sie beim nächsten Neustart wieder neu aufgesetzt werden. iptables ist ein Ersatz für ipfwadm und ipchains:

Regeln dauerhaft erstellen

Deine jetzige Firewall-Konfiguration ist im Kernel gespeichert und geht also beim Neustart verloren. iptables-save und iptables-restore schreiben steht auf meiner TODO-Liste. Wenn es sie geben wird, werden sie cool sein, ich verspreche es. In der Zwischenzeit schreib die Befehle, die nötig sind, um deine Regeln zu erstellen, in ein Init-Script. Versichere dich, dass du etwas Intelligentes tust, falls einer der Befehle nicht ausgeführt werden kann (normalerweise 'exec /sbin/sulogin').

Eine wirklich schnelle Anleitung zum Paketfiltern

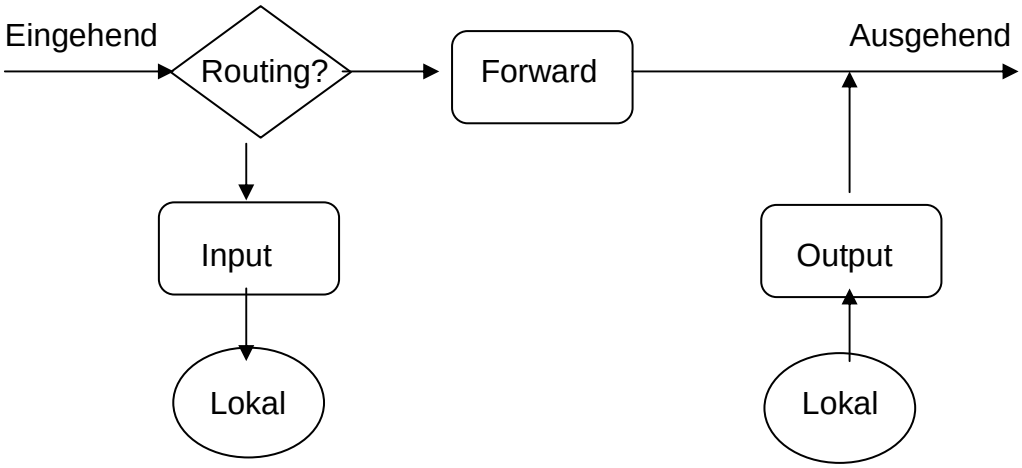
Die meisten Leute haben nur eine einfach PPP-Verbindung zum Internet und wollen nicht, dass irgendjemand in ihr Netzwerk oder in die Firewall kommen kann:

```
## Verbindungsaufspürende Module einfügen (Wenn nicht schon im Kernel).
# insmod ip_conntrack
# insmod ip_conntrack_ftp

## Kette erstellen, die neue Verbindung blockt, es sei denn, sie kommen
## von innen
# iptables -N block
# iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A block -j DROP

## Von INPUT und FORWARD Ketten zu dieser Kette springen
# iptables -A INPUT -j block
# iptables -A FORWARD -j block
```

Wie Pakete die Filter passieren

Der Kernel beginnt mit drei Listen von Regeln in der Filtertabelle; diese Listen werden Firewall-Ketten oder nur Ketten genannt. Diese drei Ketten heißen INPUT, OUTPUT und FORWARD.

Die Ketten sind so organisiert:



Die drei abgerundeten Rechtecke repräsentieren die drei oben erwähnten Ketten. Wenn ein Paket einen Kreis im Diagramm erreicht, wird diese Kette untersucht, um über das Schicksal des Pakets zu entscheiden. Wenn die Kette besagt, dass das Paket zu DROPPEN ist, wird das Paket hier gekillt, wenn die Kette jedoch sagt, dass das Paket zu akzeptieren (ACCEPT) ist, kann es weiter durch das Diagramm reisen. Eine Kette ist eine Checkliste von Regeln. Jede Regel sagt, was mit dem Paket zu tun ist. Wenn die Regel nicht auf das Paket zutrifft, wird die nächste Regel befragt. Wenn es endlich keine Regeln zum Befragen mehr gibt, sieht sich der Kernel die Policy der Kette an und entscheidet, was zu tun ist. In einem sicherheitsbewussten System sagt diese Policy dem Kernel normalerweise, dass er das Paket DROPPEN soll.

1. Wenn ein Paket eingeht (z.B. durch die Netzwerkkarte), sieht sich der Kernel zunächst die Zieladresse des Pakets an: das wird 'Routing' genannt.

2. Wenn das Paket für diesen Rechner bestimmt ist, wandert es im Diagramm an die INPUT-Kette. Wenn es diese passiert, wird es der auf dieses Paket wartende Prozess erhalten.

3. Andernfalls, wenn der Kernel Forwarding nicht aktiviert hat, oder er nicht weiß, wie er das Paket weiterleiten soll, wird das Paket verworfen. Wenn Forwarding aktiviert ist und das Paket für eine andere Netzwerkschnittstelle (wenn du eine hast) bestimmt ist, geht das Paket in unserm Diagramm direkt zur FORWARD-Kette. Wenn es dort akzeptiert (ACCEPT) wird, wird es weitergeleitet.

4. Schließlich kann ein Programm, das auf dem Rechner läuft, auch Netzwerkpakete verschicken. Diese Pakete gehen direkt zur OUTPUT-Kette. Wenn diese das Paket akzeptiert, wandert es weiter zu welcher Schnitt- stelle es auch immer bestimmt ist.

iptables verwenden

iptables hat eine recht detaillierte Man-page (man iptables), wenn du mehr Informationen über Besonderheiten brauchst. Es gibt verschiedene Dinge, die du mit iptables machen kannst. Du fängst an mit den drei eingebauten Ketten INPUT, OUTPUT FORWARD, die du nicht löschen kannst. Lass uns einen Blick auf die Operationen werfen, mit denen man auf ganzen Ketten arbeiten kann:

1. **Eine neue Kette erstellen (-N).**
2. **Eine leere Kette löschen (-X).**
3. **Die Policy für eine eingebaute Kette ändern (-P).**
4. **Die Regeln einer Kette auflisten (-L).**
5. **Die Regeln aus einer Kette ausspülen (flush) (-F).**
6. **Paket- und Bytezähler aller Regeln einer Kette auf Null stellen (-Z).**

Es gibt verschiedene Wege, die Regeln in einer Kette zu manipulieren:

1. **Eine neue Regel an eine Kette anhängen (-A).**
2. **Eine neue Regel an eine bestimmte Position in der Kette einfügen (-I).**
3. **Eine Regel an bestimmter Position in der Kette ersetzen (-R).**
4. **Eine Regel an einer bestimmten Position in der Kette löschen (-D).**
5. **Die erste passende Regel in einer Kette löschen (-D).**

Operationen auf einer einzelnen Regel

Dies ist das A-und-O des Paketfilterns: Regeln manipulieren. Meistens wirst du vermutlich den Befehl zum Anhängen (-A) oder Löschen (-D) einer Regel verwenden. Die anderen (-I zum Einfügen und -R zum Ersetzen) sind einfache Erweiterungen dieser Konzepte. Jede Regel bestimmt eine Reihe von Bedingungen, die ein eintreffendes Paket durchlaufen muss und was weiterhin mit dem Paket geschehen soll ('target'). Zum Beispiel möchtest du vielleicht alle ICMP-Pakete, die von der IP-Adresse 127.0.0.1 kommen, verwerfen. Unsere Bedingungen sind in diesem Fall also, dass das Protokoll ICMP und die Quelladresse 127.0.0.1 sein müssen. Das Ziel ist Verwerfen (DROP). 127.0.0.1 ist die Loopback-Schnittstelle, welche du auch dann hast, wenn du keine wirkliche Netzwerkverbindung hast. Du kannst das 'ping' Programm verwenden, um solche Pakete zu generieren (es sendet einfach ein ICMP Typ 8 (echo request), auf das alle kooperierenden Hosts verbindlich mit einem ICMP Typ 0 Paket (echo reply) antworten sollten). Das macht es nützlich für einen Test.

```
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms

# iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

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

Du kannst hier sehen, dass der erste Ping ankommt (das -c 1 sagt dem Programm, dass es nur ein einziges Paket schicken soll). Dann erweitern wir die INPUT-Kette mit einer Regel ('-A'), die besagt, dass Pakete, die von 127.0.0.1 ('-s 127.0.0.1') kommen und das Protokoll ICMP ('-p icmp') verwenden, zu verwerfen sind ('-j DROP'). Dann testen wir unsere Regel mit einem zweiten Ping. Es wird eine Pause geben, bevor das Programm aufgibt, auf ein Paket zu warten, das niemals ankommen wird. Wir können die Regel mit einer von zwei Möglichkeiten löschen. Erstens, da wir wissen, dass es die einzige Regel in der INPUT-Kette ist, können wir sie nach der Nummerierung löschen, so wie:

```
# iptables -D INPUT 1
#
```

Dies löscht Regel Nummer 1 in der INPUT-Kette. Der zweite Weg ist wie das -A-Kommando, man muss nur das -A durch ein -D ersetzen. Dies ist nützlich, wenn du eine komplexe Kette von Regeln hast und nicht alle erst durchzählen möchtest, um herauszufinden, dass es Regel 37 ist, die Du loswerden willst. In diesem Fall würden wir folgendes verwenden:

```
# iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
#
```

Die Syntax von -A muss genau dieselben Optionen haben wie die Syntax des -A (oder -R) Befehls. Wenn es mehrere identische Regeln in derselben Kette gibt, wird nur die erste gelöscht.

Der statische Treffer

Die nützlichsten Treff-Kriterien kommen mit der 'statischen Erweiterung', welche die verbindungsaufspürende Analyse des 'ip_conntrack' Moduls interpretiert. Dies wird stark empfohlen. Das Bestimmen von '-m state' erlaubt eine zusätzliche --state Option, welche eine durch Komma getrennte Liste von Zuständen ist, die auf ein Paket zutreffen können (das '!' Flag besagt, dass sie nicht zutreffen). Diese Zustände sind:

NEW
Ein Paket, das eine neue Verbindung aufbaut.

ESTABLISHED
Ein Paket, das zu einer bereits existierenden Verbindung gehört (ich meine eins, das Antwortpakete hatte).

RELATED
Ein Paket, das verwandt mit, aber nicht Teil von einer bestehenden Verbindung ist, so wie ein ICMP Fehler, oder (mit dem eingefügten FTP-Modul) ein Paket, das eine FTP Datenverbindung aufbaut.

INVALID
Ein Paket, das aus welchem Grund auch immer nicht identifiziert werden konnte: Das beinhaltet Speicherknappheit und ICMP Fehler, welche nicht mit einer bekannten Verbindung korrespondieren. In der Regel sollten diese Pakete verworfen werden.

Das Ziel bestimmen

Jetzt, wo wir wissen, was für Untersuchungen wir mit einem Paket machen können, brauchen wir noch einen Weg, um zu sagen, was mit den Paketen geschehen soll, die auf unsere Tests zutreffen. Das wird das Ziel einer Regel (target) genannt.

Es gibt zwei sehr einfach eingebaute Ziele: DROP und ACCEPT. Wir sind ihnen bereits begegnet. Wenn eine Regel auf ein Paket zutrifft und ihr Ziel eins von beiden ist, werden keine weiteren Regeln konsultiert: Das Schicksal des Pakets ist entschieden.

Es gibt zwei weitere Typen von anderen als den eingebauten Zielen: Erweiterungen und benutzerdefinierte Ketten.

Benutzerdefinierte Ketten

Ein mächtiges Merkmal, das iptables von ipchains geerbt hat, ist die Möglichkeit für den Benutzer, neue Ketten zusätzlich zu den drei eingebauten (INPUT, OUTPUT und FORWARD) zu erstellen. Der Konvention nach schreibt man benutzerdefinierte Ketten in Kleinbuchstaben, um sie von den anderen zu unterscheiden (wir werden weiter unten, im Absatz „Operationen auf einer vollständigen Kette" erklären, wie man benutzerdefinierte Ketten erstellt).

Wenn ein Paket, das auf eine Regel zutrifft, deren Ziel eine benutzerdefinierte Kette ist, beginnt das Paket, die Regeln in der benutzerdefinierten Kette zu durchlaufen. Wenn diese Kette nicht über das Schicksal des Pakets entscheidet, beginnt das Paket, die Regeln der aktuellen Kette weiter zu durchlaufen, sobald die Regeln jener (benutzerdefinierten) Kette fertig durchlaufen sind.

Benutzerdefinierte Ketten können zu anderen benutzerdefinierten Ketten springen (machen aber keine Schleifen: Deine Pakete werden verworfen, wenn erkannt wird, dass sie in einer Schleife stecken).

Erweiterungen zu iptables: Neue Ziele

Der andere Typ eines Ziels ist eine Erweiterung. Eine Zielerweiterung besteht aus einem Kernelmodul und einer optionalen Erweiterung zu iptables, um neue Kommandozeilenoptionen zu bieten. Es gibt in der Standard netfilter-Distribution verschiedene Erweiterungen:

LOG

Dieses Modul bietet Kernel-Logging für zutreffende Pakete. Es kommt mit diesen zusätzlichen Optionen:

--log-level
Gefolgt von einer Level-Nummer oder einem Namen. Erlaubte Namen sind (achte auf Groß- und Kleinschreibung) 'debug', 'info', entsprechend dazu die Nummern 7 bis 0. Lies die Man-Page von syslog.conf für eine Erklärung dieser Level.

--log-prefix
Gefolgt von einem String von bis zu 30 Zeichen, wird diese Botschaft zu Beginn der Logmeldung gesendet. Dies erlaubt, dass sie eindeutig identifiziert wird.

Dieses Modul ist am nützlichsten nach einem limit target, damit deine Logs nicht überflutet werden.

REJECT

Dieses Modul hat denselben Effekt wie 'DROP', außer, dass dem Sender eine ICMP 'port unreachable' Fehlermeldung geschickt wird. Beachte, dass keine ICMP Fehlermeldung geschickt wird (siehe RFC 1122), wenn: Das gefilterte Paket als erstes eine ICMP Fehlermeldung war, oder ein unbekannter ICMP Typ.

Das gefilterte Paket ein non-head Fragment war.

Wir in der letzten Zeit zu viele ICMP Fehlermeldungen an diese Adresse geschickt haben.

REJECT kann auch mit einem optionalen '--reject-with' Argument versehen werden, welches das Antwortpaket verändert: Siehe die Man-Page.

Operationen auf einer vollständigen Kette

Ein sehr nützliches Merkmal von iptables ist die Fähigkeit, verwandte Regeln zu Ketten zu gruppieren. Du kannst die Ketten nennen, wie du willst, aber ich empfehle, Kleinbuchstaben zu verwenden, um Verwirrungen mit den eingebauten Ketten und Zielen zu vermeiden. Kettennamen können bis zu 31 Buchstaben lang sein

Eine neue Kette erstellen

Lass uns eine neue Kette erstellen. Weil ich so ein phantasievoller Typ bin, werde ich sie test nennen. Wir benutzen die '-N' oder '--new-chain' Option:

```
# iptables -N test
#
```

Es ist so einfach. Jetzt kannst du Regeln einfügen wie oben beschrieben.

Eine Kette löschen

Eine Kette löschen ist auch einfach. Man verwendet die '-X' Option und schon ist sie weg.

```
# iptables -X test
#
```

Es gibt eine Reihe von Einschränkungen beim Löschen von Ketten: Sie müssen leer sein (Siehe „Eine Kette 'flushen'" weiter unten) und sie dürfen nicht das Ziel irgendeiner Regel sein. Du kannst keine der drei eingebauten Ketten löschen. Wenn du keine Kette angibst, werden, wenn möglich, alle benutzerdefinierten Ketten gelöscht.

Eine Kette 'flushen'

Es gibt eine einfache Möglichkeit, alle Regeln aus einer Kette zu entfernen, indem man das '-F' (oder '--flush') Kommando benutzt.

```
# iptables -F forward
#
```

Wenn du keine Kette angibst, werden **alle** Ketten entleert.

Eine Kette anzeigen lassen

Du kannst dir alle Regeln einer Kette mit dem '-L' (oder '--list') Befehl anzeigen lassen.

Der für jede benutzerdefinierte Kette aufgelistete 'refcnt' ist die Anzahl von Regeln, die diese Kette als ihr Ziel haben. Dieser muss auf Null stehen (und die Kette muss leer sein), ehe sie gelöscht werden kann. Wenn kein Kettenname angegeben wird, werden alle, sogar leere Ketten, aufgelistet.

Es gibt drei Optionen, welche das '-L' begleiten können.

Die '-n' (numeric) Option ist sehr nützlich, weil sie verhindert, dass iptables versucht, die IP-Adressen aufzulösen. Dies wird, (wenn du, wie die meisten Leute, DNS

verwendest) große Wartezeiten verursachen, wenn dein DNS nicht richtig eingerichtet ist, oder wenn du DNS-Anfragen ausgefiltert Außerdem bewirkt diese Option, dass TCP und UDP Ports als Zahlen und nicht als Namen ausgedruckt werden.

Die '-v' Option zeigt Dir alle Details einer Regel, sowie Paket- und Byte- Zähler, die TOS Vergleiche und die Schnittstellen. Andernfalls werden diese Werte weggelassen

Die Policy bestimmen

Als wir erklärt haben, wie ein Paket durch eine Kette läuft, haben wir uns kurz angesehen, was passiert, wenn ein Paket das Ende einer zutreffenden Kette erreicht. In diesem Fall bestimmt die Policy der Kette das weitere Schicksal des Pakets. Nur eingebaute Ketten (INPUT, OUTPUT,FORWARD) haben Policies, da ein Paket, wenn es das Ende einer benutzerdefinierten Kette erreicht, die Reise in der vorherigen Kette wieder aufnimmt.

Die Policy kann entweder ACCEPT (akzeptieren) oder DROP (verwerfen) sein.

.

Tipps für das Design von Paketfiltern

Eine bekannte Weisheit im Computer-Sicherheits-Bereich ist es, alles zu blocken, dann einzelne Löcher, wenn nötig, zu öffnen. Das wird auch oft gesagt als 'Alles, was nicht explizit erlaubt ist, ist verboten.'. Wenn Sicherheit Dein Hauptziel ist, empfehle ich dieses Vorgehen. Lass keine Dienste laufen, die Du nicht brauchst, auch, wenn Du denkst, dass Du den Zugang hierzu geblockt hast. Wenn Du anfängst, eine Firewall zu bauen, fang mit nichts an und blocke alle Pakete, füge dann Dienste hinzu und lass die benötigten Pakete durch. Ich empfehle mehrfache Sicherheit: kombiniere tcp-wrapper (für Verbindungen zu dem Paketfilter selbst), Proxies (für Verbindungen durch die Firewall) Route Verification und Paketfilter. Route Verification bedeutet, dass ein Paket, das von einer unerwarteten Schnittstelle kommt, verworfen wird: Wenn Dein internes Netzwerk zum Beispiel die Adressen 10.1.1.0/24 hat und ein mit dieser Quelladresse zu einer externen Schnittstelle kommt, wird es verworfen. Dies kann für die ppp0-Schnitt- stelle wie folgt aktiviert werden:

```
# echo 1 > /proc/sys/net/ipv4/conf/ppp0/rp_filter
#
```

Oder für alle existierenden und in Zukunft existierenden Schnittstellen so:

```
# for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
#     echo 1 > $f
# done
#
```

Debian tut dies, wo immer es möglich ist, standardmäßig. Wenn Du asymmetrisches Routing einsetzt (ich meine, wenn Du Pakete aus seltsamen Richtungen erwartest), wirst Du dieses Filtern auf diesen Schnittstellen deaktivieren wollen. Logging ist

nützlich, wenn man eine Firewall aufsetzt und irgendetwas nicht funktioniert, auf einer produktiven Firewall solltest Du es jedoch mit der 'limit' Option verwenden, um jemanden davon abzuhalten, Deine Logs zu überfluten. Für sichere Systeme empfehle ich 'connection tracking': Es bietet einen Overhead, da alle Verbindungen verfolgt werden, aber es ist sehr nützlich für kontrollierten Zugang zu Deinem Netzwerk.

```
# iptables -N no-conns-from-ppp0
# iptables -A no-conns-from-ppp0 -m state --state ESTABLISHED,RELATED –j
  ACCEPT
# iptables -A no-conns-from-ppp0 -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A no-conns-from-ppp0 -i ppp0 -m limit -j LOG --log-prefix "Bad packet
  from ppp0:"
# iptables -A no-conns-from-ppp0 -i ! ppp0 -m limit -j LOG --log-prefix "Bad packet
  not from ppp0:"
# iptables -A no-conns-from-ppp0 -j DROP

# iptables -A INPUT -j no-conns-from-ppp0
# iptables -A FORWARD -j no-conns-from-ppp0
```

Eine gute Firewall zu bauen liegt jenseits der Möglichkeiten dieses HOWTOs, aber mein Ratschlag ist: Sei immer Minimalist. Lies das Security-HOWTO für mehr Informationen über das Testen und Prüfen Deines Rechners.