Linux 2.4 Packet Filtering HOWTO

Rusty Russell, Mailingliste `netfilter@lists.samba.org` v1.0.2 Mon May 1 18:09:31 CST 2000

Ins Deutsche uebersetzt von Melanie Berg (mel@sekurity.de)

Contents

1 Einleitung

Willkommen, geschaetzer Leser.

Es wird angenommen, dass Du weisst, was eine IP-Adresse, eine Netzwerk- adresse, Routing und DNS sind. Wenn Du das nicht wissen solltest, empfehle ich Dir, das Networking-Concepts-HOWTO zu lesen.

Dieses HOWTO wechselt zwischen einer leichten Einfuehrung (mit welcher Du Dich jetzt warm und verschwommen fuehlen wirst, aber ungeschuetzt in der wirklichen Welt) und rohen Enthuellungen, welche wohl alle ausser den haertesten unter uns verwirrt, paranoid und starkes Geschuetz suchend hinterlassen werden.

Dein Netzwerk ist nicht **sicher**. Das Problem, schnelle, bequeme Kommunikation durch eine Einschraenkung von guten, und nicht schlechten, Absichten zuzulassen, ist kongruent zu dem anderen schwer zu behandelnden Problem, auf der einen Seite Freie Sprache zu erlauben und auf der anderen Seite den Schrei nach "Feuer!" in einem ueberfuellten Theater zu verbieten. Dieses Problem wird im Zuge dieses HOWTOs nicht geloest werden.

Also kannst Du nur entscheiden, wo der Kompromiss liegen wird. Ich werde versuchen, Dir eine Einfuehrung ueber ein paar erhaeltliche Tools und einige Unsicherheiten, derer man sich bewusst sein soll, zu geben, in der Hoffnung, dass Du sie fuer gute, und nicht fuer boese Zwecke gebrauchen wirst. Ein anderes aequivalentes Problem.

2 Wo ist die offizielle Website? Gibt es eine Mailingliste?

Es gibt drei offizielle Seiten:

- Dank an *Penguin Computing* `http://netfilter.filewatcher.org/` .
- Dank an *The Samba Team and SGI* `http://netfilter.samba.org/` .
- Dank an *Harald Welte* `http://netfilter.gnumonks.org/` .

Fuer die offizielle Netfilter-Mailingliste siehe Sambas Listserver

Netfilter List `http://www.netfilter.org/contact.html#list` .

3 Also was ist ein Paketfilter?

Ein Paketfilter ist ein Stueck Software, das sich die *Header* von passierenden Paketen ansieht und ueber das Schicksal des vollstaendigen Pakets entscheidet. Es koennte entscheiden, das Paket zu DROPPEN (ich meine das Paket zu verwerfen, als waere 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 koennen, aber das generelle Prinzip vom Ansehen der Header und ueber das Schicksal der Pakete entscheiden ist immernoch da.

3.1 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 Moeglichkeit, bestimmte Arten von Traffic zu erlauben, und andere zu verbieten. Zum Beispiel enthaelt der Header eines Pakets die Zieladresse des Pakets, also kannst Du verhindern, dass Pakete zu einem bestimmten Teil des aeusseren 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 froehlich herunterzuladen. Man kann das Problem loesen, indem man den Paketfilter beauftragt, keine Pakete von oder zu Adressen, die doubleclick.net gehoeren, zuzulassen (es gibt hierfuer auch bessere Wege: siehe Junkbuster).

Sicherheit:

Wenn Dein Linuxrechner das einzige zwischen dem Chaos des Internet und Deinem netten, ordentlichen Netzwerk ist, ist es schoen, zu wissen, dass Du einschraenken kannst, was fuer Dinge durch Deine Tuer kommen. Zum Beispiel kannst Du alles erlauben, was aus Deinem Netzwerk rausgeht, aber Du koenntest besorgt sein ueber den wohlbekannten 'Ping of Death', der von boesen Aussenstehenden hereinkommen koennte. Als ein anderes Beispiel moechtest Du vielleicht nicht, dass Fremde zu Deinem Linuxrechner telnetten koennen, obwohl all Deine Accounts Passwoerter haben. Vielleicht moechtest 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 koennte 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 vorfaellt; vielleicht kannst Du dann etwas daran aendern, oder vielleicht bist Du bloss von Natur aus neugierig.

3.2 Wie filtere ich Pakete unter Linux?

Linuxkernel hatten Paketfilter seit der 1.1er Serie. Die erste Version, ba- sierend auf 'ipfw' von BSD, wurde von Alan Cox Ende 1994 portiert. Dies wurde von Jos Vos und anderen fuer Linux 2.0 weiterentwickelt; das tool 'ipfwadm' kontrollierte die Filterregeln des Kernels. Mitte 1998, fuer Linux 2.2, habe ich den Kernel mit Hilfe von Michael Neuling sehr stark ueberarbeitet und das tool 'ipchains' eingefuehrt. Mitte 1999, endlich, gab es fuer Linux 2.4 eine komplette Ueberarbeitung des Kernels und somit das Tool fuer 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 iptables Modul im Linuxkernel, auf dem andere Dinge (wie die iptables Module) aufbauen koennen. Das bedeutet, dass Du Kernel 2.3.15 oder hoeher brauchst und CONFIG_NETFILTER in der Kernel-Konfiguration mit 'Y' beantworten musst.

Das Tool `iptables` spricht mit dem Kernel und sagt ihm, welche Pakete zu filtern sind. Wenn Du kein Programmierer und auch nicht ueberaus neugierig bist, ist das der Weg, auf dem Du den Paketfilter kontrollieren wirst.

3.2.1 iptables

Das `iptables` Tool fuegt Regeln in die Filtertabellen des Kernels ein und loescht andere. Das bedeutet, dass die Regeln, wie immer Du sie aufsetzt, beim Neustart des Rechners verloren sein werden. Lies 3.2.2 (Regeln dauerhaft erstellen), um sicherzugehen, dass sie beim naechsten Neustart wieder neu aufgesetzt werden.

`iptables` ist ein Ersatz fuer `ipfwadm` und `ipchains`: Lies 8 (ipchains und ipfwadm verwenden), um den Gebrauch von iptables schmerzfrei zu vermeiden, wenn Du eins der beiden Tools verwendest.

3.2.2 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 noetig sind, um Deine Regeln zu erstellen, in ein Init-Script. Versichere Dich, dass Du etwas Intelli- gentes tust, falls einer der Befehle nicht ausgefuehrt werden kann (normalerweise 'exec /sbin/sulogin').

4 Wer zum Teufel bist Du, und wieso spielst Du mit meinem Kernel rum?

Ich bin Rusty, der Linux-IP-Firewall-Maintainer und nur ein anderer Programmierer, der zufaellig zur richtigen Zeit am richtigen Ort war. Ich schrieb ipchains (siehe 3.2 (Wie filtere ich Pakte unter Linux?) weiter oben fuer den Verdienst der Leute, die wirklich gearbeitet haben) und habe genug gelernt, um Paketfilter dieses Mal richtig zu machen. Hoffe ich.

WatchGuard `http://www.watchguard.com` , ein exzellentes Firewall-Unternehmen , das die wirklich nette plug-in Firebox verkauft, hat mir angeboten, mich fuer's Nichtstun zu bezahlen, also konnte ich all meine Zeit darauf verwenden, dieses Zeug hier zu schreiben und mich um mein frueheres Zeug zu kuemmern. Ich habe 6 Monate vorhergesagt, und es dauerte 12, aber am Ende habe ich gefuehlt, dass es richtig gemacht wurde. Viele Ueberarbeitungen, einen Festplatten-Crash, einen gestolenen Laptop, eine Reihe von kaputten Filesystemen und einen zerbrochenen Bildschirm spaeter, ist es endlich fertig.

Wo ich gerade hier bin, moechte ich die falschen Auffassungen von einigen Leuten richtigstellen: Ich bin kein Kernel-Guru. Ich weiss das, weil mich meine Arbeit am Kernel mit einigen von Ihnen in Kontakt gebracht hat: David S. Miller, Alexey Kuznetsov, Andi Cleen, Alan Cox. Wie auch immer, sie sind mit der wahren Magie beschaeftigt, waehrend ich nur im seichten Ende wate, wo es sicher ist.

5 Rustys 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: (Anm.d.Uebersetzerin: Die Rauten am Zeilen- anfang muessen fuer Copy and Paste entfernt werden)

```
## Verbindungsaufspuerende Module einfuegen (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
```

6 Wie Pakete die Filter passieren

Der Kernel beginnt mit drei Listen von Regeln in der Filtertabelle; diese Listen werden **Firewall-Ketten** oder nur **Ketten** (von englisch "chain" = Kette, And.d.Uebersetzerin) genannt. Diese drei Ketten heissen **INPUT**, **OUTPUT** und **FORWARD**.

Das unterscheidet sich sehr davon, wie der 2.0er oder 2.2er Kernel funktionierte!

Fuer Ascii-Art Fans, die Ketten sind so arrangiert:

```
                        -----
 Eingehend             /     \         Ausgehend
      -->[Routing ]   ---> |FORWARD|------->
         [Entscheidung]     \_____/        ^
                |                          |
                v                         ____
               ___                       /    \
              /   \                     |OUTPUT|
             |INPUT|                     \____/
              \___/                        ^
                |                          |
                 ----> Lokaler Prozess ---
```

Die drei Kreise repraesentieren die drei oben erwaehnten Ketten. Wenn ein Paket einen Kreis im Diagramm erreicht, wird diese Kette untersucht, um ueber 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 'Wenn der Paket-header so-und-so aussieht, ist das-und-das mit dem Paket zu tun'. Wenn die Regel nicht auf das Paket zutrifft, wird die naechste 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 zunaechst die Zieladresse des Pakets an: Das wird 'Routing' genannt.
2. Wenn das Paket fuer 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 fuer 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 laeuft, auch Netzwerkpakete verschicken. Diese Pakete gehen direkt zur OUTPUT-Kette. Wenn diese das Paket akzeptiert, wandert es weiter zu der Schnitt- stelle, fuer die es bestimmt ist.

7 iptables verwenden

iptables hat eine recht detaillierte Man-page (`man iptables`), wenn Du mehr Informationen ueber Besonderheiten brauchst. Diejenigen von Euch, die mit ipchains vertraut sind, werden vielleicht einfach einen Blick in 10 (Unterschiede zwischen iptables and ipchains) werfen wollen, sie sind sehr aehnlich.

Es gibt verschiedene Dinge, die Du mit `iptables` machen kannst. Du faengst an mit den drei eingebauten Ketten `INPUT`, `OUTPUT` und `FORWARD`, die Du nicht loeschen 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 loeschen (-X).
3. Die Policy fuer eine eingebaute Kette aendern (-P).
4. Die Regeln einer Kette auflisten (-L).
5. Die Regeln aus einer Kette ausspuelen (flush) (-F).
6. Paket- und Bytezaehler 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 anhaengen (-A).
2. Eine neue Regel an eine bestimmte Position in der Kette einfuegen (-I).
3. Eine Regel an bestimmter Position in der Kette ersetzen (-R).
4. Eine Regel an einer bestimmten Position in der Kette loeschen (-D).
5. Die erste passende Regel in einer Kette loeschen (-D).

7.1 Was Du siehst, wenn Dein Computer hochfaehrt

iptables kann ein Modul (mit dem Namen iptables_filter.o) sein, welches automatisch geladen werden sollte, sobald Du `iptables` das erste Mal startest. Es kann auch permanent in den Kernel hineinkompiliert sein.

Bevor irgendwelche iptables-Befehle ausgefuehrt werden (Sei vorsichtig: Manche Distributionen werden iptables in den Init-Scripts haben), wird es keine Regeln in einer der eingebauten Ketten (INPUT, OUTPUT, FORWARD) geben, und die Policy aller Ketten wird auf 'ACCEPT' stehen. Du kannst die Standard-Policy der FORWARD-Kette aendern, indem Du die Option 'forward=0' im iptables_filter module mitgibst.

7.2 Operationen auf einer einzelnen Regel

Dies ist das A-und-O des Paketfilterns: Regeln manipulieren. Meistens wirst Du vermutlich den Befehl zum Anhaengen (-A) oder Loeschen (-D) einer Regel verwenden. Die anderen (-I zum Einfuegen 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 moechtest 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 muessen. 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 nuetzlich fuer 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 koennen die Regel mit einer von zwei Moeglichkeiten loeschen. Erstens, da wir wissen, dass es die einzige Regel in der INPUT-Kette ist, koennen wir sie nach der Nummerierung loeschen, so wie:

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

Dies loescht 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 nuetzlich, wenn Du eine komplexe Kette von Regeln hast und nicht alle erst durchzaehlen moechtest, um herauszufinden, dass es Regel 37 ist, die Du loswerden willst. In diesem Fall wuerden 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 -D) Befehls. Wenn es mehrere identische Regeln in derselben Kette gibt, wird nur die erste geloescht.

7.3 Filterbestimmungen

Wir haben die Verwendung von -p, um das Protokoll, und -s, um die Quell- adresse zu bestimmen, gesehen, aber es gibt noch andere Optionen, die wir benutzen koennen, um die Charakteristika des Pakets zu bestimmen. Was nun folgt, ist ein erschoepfendes Kompendium.

7.3.1 Quell- und Ziel-IP-Adresse bestimmen

Quell- ('-s', '–source' oder '-src') und Ziel- ('-d', '–destination' oder '–dst') IP-Adresse koennen auf vier Arten bestimmt werden. Die meistverbreitete Methode besteht darin, den vollen Namen zu verwenden, wie 'localhost' oder 'www.linuxhq.com'. Der zweite Weg ist, die IP-Adresse so wie '127.0.0.1' zu bestimmen.

Der dritte und der vierte Weg erlauben Bestimmungen einer Gruppe von IP-Adressen, so wie '199.95.207.0/24' oder '199.95.207.0/255.255.255.0'. Beide bestimmen alle IP-Adressen von 199.95.207.0 bist 199.95.207.255. Die Zahlen hinter dem '/'sagen, welcher Teil der IP-Adresse signifikant ist. '/32' oder '/255.255.255.255' ist der Standard (trifft auf alle IP-Adressen zu). Um ueberhaupt keine IP-Adresse zu bestimmen, kann '/0' verwendet werden, zum Beispiel so:

```
[BEACHTE: '-s 0/0' ist hier redundant. ]
# iptables -A INPUT -s 0/0 -j DROP
#
```

Dies wird selten verwendet, da der obige Effekt derselbe ist, wie die '-s' Option ueberhaupt nicht zu bestimmen.

7.3.2 Inversion bestimmen

Viele Flags (Optionen), wie '-s' (oder '–source') und '-d' (oder '–destination') koennen vor ihren Argumenten ein vorangestelltes '!' (Gesprochen: 'NICHT') haben, um auf **nicht** gegebene Adressen zuzutreffen. Zum Beispiel trifft '-s ! localhost' auf alle Pakete zu, die nicht von localhost kommen.

7.3.3 Das Protokoll bestimmen

Das Protokoll kann mit der '-p' (oder '–protocol') Option bestimmt werden. Protokoll kann eine Zahl sein (wenn Du die numerischen Protokollwerte fuer IP kennst) oder ein Name fuer die speziellen Faelle von 'TCP', 'UDP' oder 'ICMP'. Gross- und Kleinschreibung ist egal, also funktioniert 'tcp' genauso wie 'TCP'.

Dem Protokollnamen kann ein '!' vorangestellt werden, um ihn zu invertieren. Zum Beispiel bezieht sich '-p ! TCP' auf alle **nicht**-TCP-Pakete.

7.3.4 Eine Schnittstelle bestimmen

Die '-i' (oder '–in-interface') und die '-o' (oder '–out-interface') Optionen bestimmen den Namen einer **Schnittstelle**. Eine Schnittstelle ist eine physikalische Komponente, an der Pakete eingehen ('-i') oder ausgehen ('-o') koennen. Du kannst das **ifconfig**-Kommando benutzen, um alle Schnittstellen zu sehen, die im Moment 'up' (ich meine aktiv) sind.

Pakete, die die `INPUT`-Kette durchwandern, haben keine Output-Schnittstelle, also wird jegliche Regel, die in dieser Kette die '-o' Option benutzt, niemals zutreffen. Ebenso haben Pakete, die die `OUTPUT`-Kette durchwandern, keine input-Schnittstelle, also wird jegliche Regel, die in dieser Kette die '-i' Option benutzt, niemals zutreffen.

Nur Pakete, die die `FORWARD`-Kette durchwandern, koennen beides haben, Input- und Output-Schnittstelle.

Es ist vollkommen in Ordung, eine Schnittstelle zu bestimmen, die im Moment noch nicht existiert; nichts wird auf die Regel zutreffen, bis die Schnitt- stelle aktiviert wird. Dies ist extrem nuetzlich fuer Dialup-PPP Verbin- dungen (gewoehnlich Schnittstelle `ppp0`) und aehnliches.

Als ein Sonderfall wird ein Schnittstellenname, der mit einem '+' endet, auf alle Schnittstellen zutreffen, welche mit diesem String beginnen (ob sie in dem Moment existieren oder nicht). Um zum Beispiel eine Regel zu bestimmen, die auf alle PPP-Schnittstellen zutrifft, wuerde man die `-i ppp+` Option verwenden.

Dem Namen der Schnittstelle kann ein '!' vorausgehen, um Pakete zu erfassen, die **nicht** auf die angegebene Schnittstelle passen.

7.3.5 Fragmente bestimmen

Manchmal ist ein Paket zu gross, um an einem Stueck durch eine Leitung zu gehen. Wenn das geschieht, wird das Paket in **Fragmente** aufgeteilt und in mehreren Paketen weiterverschickt. Die andere Seite setzt diese Fragmente wieder zusammen, um das gesamte Paket zu rekonstruieren.

Das Problem mit Fragmenten ist, dass das erste Fragment die kompletten zu untersuchenden Header-Felder (IP + TCP, UDP und ICMP) enthaelt, waehrend die nachfolgenden Fragmente nur Teilstuecke der Header (IP ohne die zusaetzlichen Protokoll-Felder) enthalten. Somit ist es nicht moeglich, in nachfolgenden Fragmenten nach Protokoll-Headern zu suchen (wie es zum Beispiel bei TCP, UDP und ICMP Erweiterungen getan wird).

Wenn Du 'connection tracking' oder NAT verwendest, werden alle Fragmente wieder miteinander verschmolzen werden, bevor sie den Paketfilter erreichen, also wirst Du Dir nie Sorgen um Fragmente machen muessen.

Andernfalls ist es wichtig, zu verstehen, wie Fragmente von den Filterregeln behandelt werden. *Keine* Filterregel, die nach nicht verfuegbaren Informationen fragt, wird zutreffen. Dies bedeutet, dass das erste Fragment wie jedes andere Paket auch behandelt wird, das zweite und weitere Fragmente aber nicht. Eine Regel wie `-p TCP -sport www` (die einen Quellport von www bestimmt) wird niemals auf ein (anderes als als das erste) Fragment zutreffen. Gleiches gilt fuer die gegenteilige Regel `-p TCP -sport ! www`.

Wie auch immer, Du kannst eine besondere Regel fuer zweite und weitere Fragmente bestimmen, wenn Du die '-f' (oder '–fragment') Option verwendest. Es ist auch erlaubt, eine Regel zu bestimmen, die *nicht* auf zweite oder weitere Fragmente zutrifft, indem man dem '-f' ein '!' voranstellt.

Gewoehnlich wird es als sicher angesehen, zweite und weitere Fragmente durchzulassen, da das erste Fragment ausgefiltert werden kann und somit eine Wieder-Zusammensetzung beim Zielhost verhindert wird. Andererseits gibt es Software-Bugs, durch die Maschinen zum Absturz gebracht werden koennen, wenn man auf bestimmte Weise zusammengesetzte Fragmente an sie sendet. Deine Entscheidung.

Bemerkung fuer Netzwerk-Leiter: Missgeformte Pakete (TCP, UDP und ICMP Pakete, die zu klein fuer den Firewall-Code sind, um Ports oder ICMP Code oder Type zu lesen) werden verworfen, wenn solche

Untersuchungen versucht werden. TCP-Fragmente starten also bei Position 8.

Als ein Beispiel verwirft folgende Regel jegliche Fragmente, die an 192.168.1.1 gehen:

```
# iptables -A OUTPUT -f -d 192.168.1.1 -j DROP
#
```

7.3.6 iptables Erweiterungen: Neue Treffer

`iptables` ist **erweiterbar**, was bedeutet, dass beides, sowohl Kernel als auch das iptables Tool erweitert werden koennen, um neue Moeglichkeiten anzubieten.

Manche dieser Erweiterungen sind Standard, andere sind eher exotisch. Erweiterungen koennen von anderen Leuten gemacht werden und separat fuer nette User verbreitet werden.

Kernelerweiterungen leben gewoehnlich im Unterverzeichnis fuer Kernelmodule, so wie /lib/modules/2.3.15/net. Sie werden bei Bedarf geladen, wenn Dein Kernel mit der CONFIG_KMOD Option kompiliert wurde, also solltest Du sie nicht manuell einfuegen muessen.

Erweiterungen fuer das iptables-Programm sind shared libraries, welche gewoehnlich in /usr/local/lib/iptables leben, obwohl eine Distribution sie auch nach /lib/iptables oder /usr/lib/iptables legen koennte.

Erweiterungen kommen in zwei Arten: neue Ziele ('targets') und neue Treffer ('matches'), wir werden weiter unten ueber neue Ziele sprechen. Manche Protokolle bieten automatisch neue Tests an: Im Moment sind das TCP, UDP und ICMP, wie unten gezeigt werden wird.

Fuer diese wirst Du die Moeglichkeit haben, die neuen Tests auf der Kommandozeile nach der '-p' Option zu bestimmen, was die Erweiterungen laden wird. Benutze fuer explizite neue Tests die '-m' Option, um die Erweiterungen zu laden. Danach werden die erweiterten Optionen verfuegbar sein.

Um Hilfe fuer eine Erweiterung zu bekommen, benutze die Option, um es zu laden ('-p', '-j' oder '-m') gefolgt von einem '-h' oder '–help'. Zum Beispiel:

```
# iptables -p tcp --help
#
```

TCP Erweiterungen Die TCP Erweiterungen werden automatisch geladen, wenn '-p tcp' angegeben ist. Sie bieten folgende Optionen (keine davon passt auf Fragmente):

–tcp-flags

Gefolgt von einem optionalen '!', dann zwei Zeichenketten von Flags, erlaubt Dir, nach speziellen TCP-Flags zu filtern. Die erste Zeichenkette von Flags ist die Maske: eine Liste von Flags, die Du untersuchen willst. Die zweite Zeichenkette besagt, welche Flags gesetzt sein sollen. Zum Beispiel:

```
# iptables -A INPUT --protocol tcp --tcp-flags ALL SYN,ACK -j DENY
```

Dies besagt, dass alle Flags untersucht werden sollen ('ALL' ist synonym mit 'SYN, ACK, FIN, RST, URG, PSH'), dass aber nur SYN und ACK gesetzt sein sollen. Es gibt auch ein Argument 'NONE', was 'Keine Flags' bedeutet.

–syn

Mit einem optionalen vorangestellten '!', ist dies eine kurze Schreibweise fuer '–tcp-flags SYN,RST,ACK SYN'.

–source-port

Gefolgt von einem optionalen '!', dann entweder ein TCP-Port oder eine Reihe von Ports. Ports koennen Portnamen sein, so wie in /etc/services aufgelistet, oder numerisch angegeben. (So waeren fuer den SMTP-Port folgende Eintraege moeglich: '–source-port smtp' oder '–source-port 25', Anm.d.Uebersetzerin)

Reihen sind entweder zwei Portnamen, getrennt durch ein '-', oder (um groesser als oder gleich ein gegebener Port zu bestimmen) ein Port gefolgt von einem '-', oder (um kleiner oder gleich einem gegebene Port zu bestimmen), ein '-' gefolgt von dem Port.

–sport

Ist synonym mit '–source-port'.

–destination-port

und

–dport

ist dasselbe wie oben, nur dass sie den Zielport bestimmen, und nicht den Quellport, der zutreffend sein muss.

–tcp-option

Gefolgt von einem optionales '!' und einer Nummer, trifft auf ein TCP- Paket zu, bei dem die TCP-Option gleich der Nummer ist. Ein TCP-Paket, welches keinen komplettein Header hat, wird automatisch verworfen, wenn ein Versuch gestartet wird, die TCP-Optionen zu untersuchen.

Erklaerung der TCP-Flags Es ist manchmal nuetzlich, TCP-Verbindungen in die eine Richtung zu erlauben, in die andere jedoch nicht. Zum Beispiel moechtest Du vielleicht alle Verbindungen zu einem externen WWW-Server erlauben, aber keine Verbindungen von diesem Server.

Die einfache Annaeherung daran wuerde sein, alle einkommenden TCP-Pakete von diesem Server zu blocken. Leider benoetigen TCP-Verbindungen Pakete in beide Richtungen, um ueberhaupt zu funktionieren.

Die Loesung liegt darin, nur die Pakete zu blocken, die verwendet werden, um eine Verbindung aufzubauen. Diese Pakete werden **SYN**-Pakete genannt (Ok, technisch sind es Pakete, die das SYN-Flag gesetzt haben, und das FIN und ACK Flag nicht, aber wir nennen sie kurz SYN-Pakete). Indem man nur diese Pakete blockt, koennen schon Verbindungsversuche von der Gegenseite unterbunden werden.

Hierfuer wird das '–syn' Flag benutzt: Es gilt nur fuer Regeln, die TCP als Protokoll bestimmen. Um zum Beispiel TCP-Verbindungs-Versuche von 192.168.1.1 zu bestimmen:

```
-p TCP -s 192.168.1.1 --syn
```

Dieses Flag kann invertiert werden, indem man ein '!' voranstellt, welches sich dann auf jedes andere Paket bezieht ausser auf das zum Verbindungs- aufbau.

UDP Erweiterungen Diese Erweiterungen werden automatisch geladen, wenn '-p udp' bestimmt wird. Sie bieten die Optionen '–source-port', '–sport', '–destination-port' und '–dport', wie sie oben fuer TCP detailliert beschrieben wurden.

ICMP Erweiterungen Diese Erweiterungen werden automatisch geladen, wenn '-p icmp' bestimmt wird. Sie bieten nur eine neue Option:

–icmp-type

Gefolgt von einem optionalen '!', dann entweder ein ICMP-Typname (zum Beispiel 'host-unreachable'), oder ein numerischer Typ (zum Beispiel '3'), oder ein numerischer Typ und Code, getrennt durch ein '/' (zum Beispiel '3/3'. Eine Liste von verfuegbaren ICMP Typname zeigt das Kommando '-p icmp –help'.

Andere gueltige Erweiterungen Die anderen Erweiterungen im netfilter-Paket sind Demonstrations-Erweiterungen, die (wenn installiert) mit der Option '-m' aufgerufen werden koennen.

mac

Dieses Modul muss explizit mit der '-m mac' oder '–match-mac' Option bestimmt werden. Es wird verwendet, um auf die MAC-Adresse einkommender Pakete zuzutreffen, und ist somit nur nuetzlich fuer Pakete, die die PREROUTING und die INPUT-Kette durchlaufen.

–mac-source

Gefolgt von einem optionalen '!', dann eine Netzwerkadresse in durch Doppelpunkte getrennter Hex-Notation, zum Beispiel '–mac-source 00:60:08:91:CC:B7'.

limit

Dieses Modul muss explizit mit der '-m limit ' oder '–match-limit' Option bestimmt werden. Es wird verwendet, um die Rate der Treffer einzuschraenken, sowie das Unterdruecken von Log-Meldungen. Es wird nur auf eine vorgegebene Anzahl von Malen pro Sekunde zutreffen (Standardmaessig 3 Treffer pro Stunde, mit einer Grenze von 5). Es hat zwei optionale Argumente:

–limit

Gefolgt von einer Zahl. Dies bestimmt die maximale Anzahl von erlaubten Treffern pro Zeiteinheit (Standard: pro Sekunde). Die Nummer kann explizit Einheiten bestimmen, indem '/second/', '/minute/', '/hour/' oder '/day/' oder Teile davon benutzt werden (so ist '5/second' dasselbe wie '5/s').

–limit-burst

Gefolgt von einer Zahl, die die maximale Grenze angibt, bevor das obere Limit erreicht wird.

Dieses Argument kann oft mit dem LOG Ziel verwendet werden, um begrenzt zu loggen. Um zu verstehen, wie das funktioniert, lass uns einen Blick auf die folgende Regel werfen, welche Pakete mit den standardmaessigen Limit-Parametern loggt:

```
# iptables -A FORWARD -m limit -j LOG
```

Das erste Mal, wenn diese Regel erreicht wird, wird das Paket geloggt; tatsaechlich werden, da die Standardgrenze 5 ist, die ersten 5 Pakete geloggt. Danach wird es zwanzig Minuten dauern, ehe ein Paket wieder von dieser Regel geloggt wird, ungeachtet dessen, wieviele Pakete wirklich ankommen. Ausserdem, wenn zwanzig Minuten vergehen, ohne dass ein treffendes Paket ankommt, wird die Grenze um eins zurueckgeschraubt: Wenn 100 Minuten lang kein Paket auf die Regel trifft, steht der Zaehler wieder auf Null, da, wo wir angefangen haben.

Zur Zeit kannst Du keine Regel mit einer Ladezeit groesser als 59 Stunden erstellen. Wenn Du also eine Standard-Rate von 1 pro Tag setzt, muss Deine Grenzrate bei weniger als 3 liegen.

Du kannst dieses Modul auch verwenden, um verschiedene Denial Of Service Attacken (DoS) zu verhindern, um mit einer schnelleren Rate Deine Erreichbarkeit zu verstaerken.

Syn-flood Schutz:

```
# iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
```

Verstohlene Portscanner:

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT
```

Ping of death:

```
# iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
```

Dieses Modul arbeitet wie eine "hysteresis door", wie in dem untenstehenden Graphen gezeigt wird.

```
          Rate (pkt/s)
                ^        .---.
                |       / DoS \
                |      /       \
Kante des DoS  -|.....:.........\.......................
    = (limit *  |    /:          \
   limit-burst) |   / :           \         .-.
                |  /  :            \       /   \
                | /   :             \     /     \
 Ende des DoS  -|/....:..............:.../.......\..../.
    = limit     |     :              :`-'         `--'
   -------------+-----+--------------+-------------------> Zeit (s)
      LOGIK =>  Trifft | Trifft nicht|   Trifft
```

Sagen wir, es trifft ein Paket pro Sekunde, mit einer 5 Paket Grenze, aber es kommen 4 Pakete pro Sekunde ein, dann drei Sekunden, und fangen dann in drei Sekunden wieder an:

```
    <--Flood 1-->           <---Flood 2--->
Total  ^                   Line  __--      YNNN
Packets|               Rate  __--      YNNN
       |            mum  __--      YNNN
    10 |        Maxi __--         Y
       |         __--            Y
       |     __--               Y
       | __--    YNNN
       |-    YNNN
     5 |    Y
       |   Y                    Schluessel: Y -> Regel trifft zu
       |  Y                                 N -> Regel trifft nicht zu
       | Y
       |Y
     0 +-------------------------------------------------->Zeit (Sekunden)

       0   1   2   3   4   5   6   7   8   9   10  11  12
```

Du kannst sehen, dass die ersten 5 Pakete das eine Paket pro Sekunde ueberschreiten duerfen, dann kommt die Grenze. Wenn es eine Pause gibt, kann eine neue Grenze gesetzt werden, aber nicht jenseits der maximalen Rate, die von dieser Regel bestimmt wurde (1 Paket pro Sekunde bis die Grenze erreicht ist).

owner

Dieses Modul versucht fuer lokal generierte Pakete, auf verschiedene Charakteristika zuzutreffen, wie das Paket generiert wurde. Es ist nur in der OUTPUT-Kette erlaubt, und sogar da koennen manche Pakete (wie ICMP Ping-Antworten) keinen Besitzer (owner) haben, und werden so niemals treffend sein.

–uid-owner userid

Trifft zu, wenn das Paket von einem Prozess mit der gegebenen, effektiven (numerischen) User ID generiert wurde.

–uid-owner groupid

Trifft zu, wenn das Paket von einem Prozess mit der gegebenen, effektiven (numerischen) Gruppen ID generiert wurde.

–pid-owner processid

Trifft zu, wenn das Paket von einem Prozess mit der gegebenen, effektiven (numerischen) Prozess ID generiert wurde.

–sid-owner processid

Trifft zu, wenn ein Paket von einem Prozess in der gegebenen session group generiert wurde.

unclean

Dieses experimentielle Modul muss explizit mit der '-m unclean' oder '–match unclean' Option bestimmt werden. Es macht verschiedene, zufaellige Gesundheits-Checks auf Paketen. Dieses Modul ist nicht getestet worden und sollte nicht als Sicherheitsloesung verwendet werden (wahrscheinlich macht es die Dinge noch schlimmer, da es selbst viele Bugs beinhalten kann). Es bietet keine Optionen an.

Zustandsbezogene Treffer Die nuetzlichsten Treff-Kriterien kommen mit der 'Zustands-Erweiterung', welche die verbindungsaufspuerende Analyse des 'ip_conntrack' Moduls interpretiert. Dies wird stark empfohlen.

Das Bestimmen von '-m state' erlaubt eine zusaetzliche Option –state, welche eine durch Komma getrennte Liste von Zustaenden ist, die auf ein Paket zutreffen koennen (das '!' Flag besagt, dass sie **nicht** zutreffen). Diese Zustaende sind:

NEW

Ein Paket, das eine neue Verbindung aufbaut.

ESTABLISHED

Ein Paket, das zu einer bereits existierenden Verbindung gehoert (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 eingefuegten 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.

7.4 Das Ziel bestimmen

Jetzt, wo wir wissen, was fuer Untersuchungen wir mit einem Paket machen koennen, 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.

7.4.1 Benutzerdefinierte Ketten

Ein maechtiges Merkmal, das `iptables` von `ipchains` geerbt hat, ist die Moeglichkeit fuer den Benutzer, neue Ketten zusaetzlich 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 7.5 (Operationen auf einer vollstaendigen Kette) erklaeren, wie man benutzerdefinierte Ketten erstellt).

Wenn auf ein Paket eine nutzerdefinierte Regel zutrifft, wird es an die entsprechende Kette uebergeben und durchlaeuft alle Regeln darin. Wenn diese Kette nicht ueber 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.

Zeit fuer etwas mehr Ascii-Art. Stell Dir zwei (dumme) Ketten vor: `INPUT` (die eingebaute Kette) und `test` (eine benutzerdefinierte Kette).

```
 ‘INPUT’                         ‘test’
----------------------------    ----------------------------
|Regel1: -p ICMP -j DROP   |    |Regel1: -s 192.168.1.1    |
|--------------------------|    |--------------------------|
|Regel2: -p TCP -j test    |    |Regel2: -d 192.168.1.1    |
|--------------------------|    ----------------------------
|Regel3: -p UDP -j DROP    |
----------------------------
```

Stell Dir jetzt ein TCP Paket vor, dass von 192.168.1.1 kommt und zu 1.2.3.4 geht. Es kommt in die `INPUT`-Kette und wird auf Regel1 geprueft - kein Treffer. Regel2 trifft aber zu, und das Ziel ist hier `test`, also ist die naechste Regel, auf die geprueft wird, der Anfang von `test`. Regel1 in `test` trifft zu, bestimmt aber kein Ziel, also wird Regel2 untersucht. Diese trifft nicht zu, und wir haben das Ende der Kette erreicht. Wir kehren zur `INPUT`-Kette zurueck, wo wir gerade Regel2 untersucht hatten, also pruefen wir jetzt auf Regel3, welche ebenfalls nicht zutrifft.

Der Weg des Pakets ist also folgender:

```
                        v    __________________________
 ‘INPUT’                |   /    ‘test’               v
------------------------|--/    -----------------------|----
|Regel1                 | /|    |Regel1                |   |
|-----------------------|/-|    |----------------------|---|
|Regel2                 /  |    |Regel2                |   |
|--------------------------|    -----------------------v----
|Regel3                 /--+___________________________/
```

```
    ------------------------|---
                            v
```

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

7.4.2 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 fuer zutreffende Pakete. Es kommt mit diesen zusaetzlichen Optionen:

–log-level

Gefolgt von einer Level-Nummer oder einem Namen. Erlaubte Namen sind (achte auf Gross- und Kleinschreibung) 'debug', 'info', 'notice', 'warning', 'err', 'crit', 'alert' und 'emerg', entsprechend dazu die Nummern 7 bis 0. Lies die Man-Page von syslog.conf fuer eine Erklaerung 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 nuetzlichsten nach einem "limit target", damit Deine Logs nicht ueberflutet 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 veraendert: Siehe die Man-Page.

7.4.3 Spezielle eingebaute Ziele

Es gibt zwei spezielle eingebaute Ziele: `RETURN` und `QUEUE`.

`RETURN` hat denselben Effekt, wie vom Ende einer Kette fallen: Fuer eine Regel in einer eingebauten Kette wird das Ziel der Regel ausgefuehrt. Fuer eine Regel in einer benutzerdefinierten Kette geht die Reise des Pakets in der vorangegangenen Kette weiter, direkt nach der Regel, die zu der benutzerdefinierten Kette gesprungen ist.

`QUEUE` ist ein besonderes Ziel, welches das Paket an die Reihe der Benutzerprozesse anhaengt. Damit das nuetzlich ist, werden zwei weitere Komponenten benoetigt:

- Ein "queue handler", der mit dem wirklichen Mechanismus vom Reichen der Pakete vom Kernel zur Anwenderseite arbeitet; und
- eine Anwendung, die Pakete empfaengt, moeglicherweise manipuliert und Urteile darueber faellt.

Der Standard "queue handler" fuer IPv4 iptables ist das ip_queue Modul, das mit dem Kernel geliefert wird und als EXPERIMENTAL gekennzeichnet ist.

Das Folgende ist ein kleines Beispiel dafuer, wie man iptables verwendet, um Pakete fuer den Userspace einzureihen:

```
# modprobe iptable_filter
# modprobe ip_queue
# iptables -A OUTPUT -p icmp -j QUEUE
```

Mit dieser Regel werden lokal generierte, ausgehende ICMP Pakete (wie sie zum Beispiel mit ping erstellt werden) dem ip_queue Modul uebergeben, welches dann versucht, die Pakete an eine Anwendung zu liefern. Wenn es keine Anwend- ung gibt, die darauf wartet, werden die Pakete verworfen.

Um eine Anwendung zu schreiben, solltest Du die libipq API verwenden. Sie wird mit iptables geliefert. Beispielcode (z.B. redirect.c) kann in der Testsuite Tools auf dem CVS Server gefunden werden.

Der Status von ip_queue kann wie folgt ueberprueft werden:

```
/proc/net/ip_queue
```

Die maximale Laenge der Queue (ich meine die Anzahl der an eine Anwendung gelieferten Pakete, ueber die kein Urteil gefaellt wurde) kann wie folgt kontrolliert werden:

```
/proc/sys/net/ipv4/ip_queue_maxlen
```

Der Standardwert der maximalen Laenge der Queue liegt bei 1024. Sobald dieses Limit erreicht wird, werden neue Pakete solange verworfen werden, bis die Laenge der Queue wieder unter das Limit faellt. Nette Protokolle, wie TCP, interpretieren verworfenen Pakete als 'Ueberfuellung' und werden hoffentlich das Senden einstellen, wenn die Queue sich fuellt. Wie auch immer, wenn der Standardwert in einer gegebenen Situation zu klein sein sollte, koennen ein paar Experimente erforderlich sein, um die ideale maximale Laenge der Queue zu bestimmen.

7.5 Operationen auf einer vollstaendigen Kette

Ein sehr nuetzliches Merkmal von `iptables` ist die Faehigkeit, 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 koennen bis zu 31 Buchstaben lang sein

7.5.1 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 einfuegen wie oben beschrieben.

7.5.2 Eine Kette loeschen

Eine Kette loeschen ist auch einfach. Man verwendet die '-X' oder '–delete-chain' Option. Wieso 'X'? Naja, alle guten Buchstaben waren schon weg.

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

Es gibt eine Reihe von Einschraenkungen beim Loeschen von Ketten: Sie muessen leer sein (Siehe 7.5.3 (Eine Kette 'flushen') weiter unten) und sie duerfen nicht das Ziel irgendeiner Regel sein. Du kannst keine der drei eingebauten Ketten loeschen.

Wenn Du keine Kette angibst, werden, wenn moeglich, alle benutzerdefinierten Ketten geloescht.

7.5.3 Eine Kette 'flushen'

Es gibt eine einfache Moeglichkeit, 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.

7.5.4 Eine Kette anzeigen lassen

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

Der fuer 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 geloescht werden kann.

Wenn kein Kettenname angegeben wird, werden alle, sogar leere Ketten, aufgelistet.

Es gibt drei Optionen, welche das '-L' begleiten koennen. Die '-n' (numeric) Option ist sehr nuetzlich, weil sie verhindert, dass `iptables` versucht, die IP-Adressen aufzuloesen. 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 hast. 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- zaehler, die TOS Vergleiche und die Schnittstellen. Andernfalls werden diese Werte weggelassen.

Beachte, dass Paket- und Bytezaehler jeweils mit dem Suffix 'K', 'M' oder 'G' fuer 1000, 1,000,000 und 1,000,000,000 ausgedruckt werden. Das '-x' Flag (expand numbers) gibt auch ganze Zahlen aus, egal wie gross sie sind.

7.5.5 Zaehler resetten (auf Null stellen)

Es ist eine nuetzliche Moeglichkeit, die Zaehler auf Null zu setzen. Dies kann mit der '-Z' (oder '–zero) Option getan werden.

Das Problem hierbei ist, dass Du manchmal die Werte der Zaehler kennen musst, direkt bevor sie zurueckgestellt werden. Im obigen Beispiel koennen zwischen dem '-L' und dem '-Z' Befehl bereits einige Pakete durchgegangen sein. Aus diesem Grund kannst Du die '-L' und die '-Z' Option gemeinsam verwenden, um die Zaehler auf Null zu setzen, *waehrend* Du sie liest.

7.5.6 Die Policy bestimmen

Als wir erklaert haben, wie ein Paket durch eine Kette laeuft, 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.

8 ipchains und ipfwadm verwenden

In der netfilter-Distribution gibt es Module mit dem Namen ipchains.o und ipfwadm.o. Fueg eins von beiden in den Kernel ein (Beachte: Sie sind nicht kompatibel mit iptables.o, ip_conntrack.o und ip_nat.o!). Dann kannst Du ipchains oder ipfwadm wie in den guten alten Zeiten verwenden.

Dies wird noch eine zeitlang unterstuetzt werden. Ich denke eine begruen- dete Formel hierzu ist 2 * [Hinweis auf Ersatz - Erster stabilen Release], jenseits des Datums, an dem eine stabile Release fuer einen Ersatz erhaeltlich ist.

Fuer ipfwadm bedeutet das, das das Ende des Supports hier liegt:

```
2 * [Oktober 1997 (2.1.102 release) - Maerz 1995 (ipfwadm 1.0)]
        + Januar 1999 (2.2.0 release)
    = November 2003.
```

Fuer ipchains bedeutet das, das das Ende des Supports hier liegt:

```
2 * [August 1999 (2.3.15 release) - Oktober 1997 (2.2.0 release)]
        + Januar 2000 (2.3.0 release?)
    = September 2003.
```

Bis 2004 brauchst Du Dir also keine Sorgen zu machen.

9 Kombinieren von NAT und Paketfiltern

Es ist weit verbreitet, dass man Network Adress Translation (siehe das NAT-HOWTO) und Paketfilter machen will. Die guten Neuigkeiten sind, dass man sie extrem gut miteinander kombinieren kann.

Du entwirfst Deine Paketfilter und kannst das NAT, das Du machst, dabei komplett ignorieren. Die Quellen und Ziele, die die Paketfilter sehen, sind die 'wirklichen' Quellen und Ziele. Wenn Du z.B. DNAT machst, um irgendeine Verbindung an 1.2.3.4 Port 80 ueber 10.1.1.1 Port 8080 zu schicken, sieht der Paketfilter Pakete an 10.1.1.1 Port 8080 (das wirkliche Ziel), nicht an 1.2.3.4 Port 80. Aehnlich kannst Du Masquerading ignorieren: Pakete scheinen von ihrer wirklichen internen IP-Adresse zu kommen (sagen wir 10.1.1.1), und Antworten werden scheinbar auch dorthin zurueckgehen.

Du kannst die Treffer-Erweiterungen fuer Zustaende verwenden, ohne die Paket- filter extra arbeiten zu lassen, da NAT sowieso 'connection tracking' erfordert. Um dem simplen Masquerading Beispiel im NAT-HOWTO zu verbieten, neue ankommende Verbindungen an der ppp0-Schnittstelle anzunehmen, wuerde folgendes reichen:

```
# Maskiere ppp0
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

# Verbiete NEW und INVALID ankommende oder weitergeleitete
# Pakete von ppp0.
iptables -A INPUT -i ppp0 -m state --state NEW,INVALID -j DROP
iptables -A FORWARD -i ppp0 0 -m state --state NEW,INVALID -j DROP

# IP-Forwarding aktivieren
echo 1 > /proc/sys/net/ipv4/ip_forward
```

10 Unterschiede zwischen iptables und ipchains

- Zuerst wurden die Namen der eingebauten Ketten von Kleinbuchstaben zu Grossbuchstaben geaendert, weil die INPUT und die OUTPUT-Kette jetzt nur lokal-bestimmte und lokal-generierte Pakete bekommen. Frueher war es so, dass sie jeweils alle eingehenden und alle ausgehenden Pakete gesehen haben.
- Die '-i' Option bedeutet jetzt eintreffendes Interface und funktioniert nur bei der INPUT und bei der FORWARD-Kette. Regeln in der FORWARD oder OUTPUT-Kette, die '-i' verwenden, sollten zu '-o' geaendert werden.
- TCP und UDP Ports muessen jetzt mit der –source-port oder –sport (oder –destination-port/–dport) Option ausgeschrieben werden, da dies jeweils die TCP oder UDP Erweiterungen laedt.
- Das TCP Flag -y ist jetzt –syn, und muss nach '-p tcp' stehen.
- Das DENY-Ziel ist jetzt, endgueltig, DROP.
- Einzelne Ketten waehrend des Auflistens auf Null setzten funktioniert.
- Eingebaute Ketten auf Null setzen leert auch Policy-Zaehler.
- Das Auflisten von Ketten gibt Dir die Zaehler als atomischen Schnapp- schuss.
- REJECT und LOG sind jetzt erweiterte Ziele, was bedeutet, dass sie separate Kernelmodule sind.
- Kettennamen koennen bis zu 31 Zeichen lang sein.
- MASQ ist jetzt MASQUERADE und hat eine andere Syntax. REDIRECT, obwohl es den Namen behaelt, musste auch eine Syntax-Aenderung durchlaufen. Lies das NAT-HOWTO fuer Informationen ueber die Konfiguration von beiden.
- Die -o Option wird nicht weiter dazu benutzt, Pakete an eine Anwendung zu senden (siehe -i weiter oben). Pakete werden jetzt mit dem QUEUE Target an den Userspace geschickt.
- Wahrscheinlich Tonnen von anderen Sachen, die ich vergessen habe.

11 Tips fuer das Design von Paketfiltern

Eine bekannte Weisheit im Computer-Sicherheits-Bereich ist es, alles zu blocken, dann einzelne Loecher, wenn noetig, zu oeffnen. Das wird auch oft gesagt als 'Alles, was nicht explizit erlaubt ist, ist ver- boten.'. 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 anfaengst, eine Firewall zu bauen, fang mit nichts an und blocke alle Pakete, fuege dann Dienste hinzu und lass die benoetigten Pakete durch.

Ich empfehle stark Sicherheit: kombiniere tcp-wrapper (fuer Verbindungen zu dem Paketfilter selbst), Proxies (fuer 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 Paket mit dieser Quelladresse zu einer externen Schnittstelle kommt, wird es verworfen. Dies kann fuer die Schnittstelle ppp0 wie folgt aktiviert werden: (Auch hier muessen fuer ein Script natuerlich wieder die Rauten am Zeilenanfang entfernt werden, Anm.d. Uebersetzerin)

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

Oder fuer 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 moeglich ist, standardmaessig. 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 nuetzlich, 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 ueberfluten.

Fuer sichere Systeme empfehle ich staerkstens 'connection tracking': Es bietet einen Overhead, da alle Verbindungen verfolgt werden, aber es ist sehr nuetzlich fuer 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 Moeglichkeiten dieses HOWTOs, aber mein Ratschlag ist: Sei immer Minimalist. Lies das Security-HOWTO fuer mehr Informationen ueber das Testen und Pruefen Deines Rechners.