

Ralf Spenneberg

Linux-Firewalls mit iptables & Co.

Sicherheit mit Kernel 2.4 und 2.6
für Linux-Server und -Netzwerke



 ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam



18 Patch-O-Matic

Der Netfilter/Iptables-Code befindet sich in konstanter Entwicklung. Die von den Distributoren verteilten Kernel und die auf <http://www.kernel.org> zur Verfügung gestellten Kernel enthalten nicht alle verfügbaren Funktionen. Einige Funktionen sind sehr nützlich, einige sind interessant, einige unsinnig und viele fehlerhaft. Dennoch sollten Sie sich die in Patch-O-Matic zusätzlich zur Verfügung gestellten Funktionen ansehen, da sie teilweise einige Probleme sehr elegant lösen können und morgen vielleicht auch im Standard-Kernel enthalten sind.

18.1 Was ist Patch-O-Matic?

Bei dem Patch-O-Matic-Paket handelt es sich um ein Patch-Paket, das seit der Iptables-Version 1.2.7 in einem eigenen Paket vertrieben wird. Dieses Paket heißt `patch-o-matic-ng` oder kurz `p-o-m`. Dieses Paket enthält Patches vom Netfilter-Team und unabhängigen Entwicklern, die noch nicht Teil des Linux-Kernels und des Iptables-Befehls sind.

Patch-O-Matic enthält viele sehr interessante Funktionen, die teilweise auch von den Distributoren in ihren Kernen integriert werden. Diese Funktionen sind aber entweder noch nicht ausreichend getestet, nicht ausreichend stabil oder nicht gut genug dokumentiert, so dass sie noch nicht in den Standard-Kernel aufgenommen wurden. Viele der Patches reifen aber in Patch-O-Matic und werden zu einem späteren Zeitpunkt tatsächlich Teil des Linux-Kernels. Der TCP-Window-Tracking-Patch von Jozsef Kadlecik ist ein Beispiel für einen derartigen Patch. Nach vielen anfänglichen Problemen ist dieser Patch aus Patch-O-Matic seit der Version 2.6.9 in dem Linux-Kernel enthalten.

18.2 Wie bekomme ich Patch-O-Matic, und wie wende ich es an?

Früher war Patch-O-Matic Teil des Iptables-Befehls. Dann wurde es als eigenständiges Paket von dem Netfilter-Team in regelmäßigen Abständen veröffentlicht. Seit dem 30. Januar 2005 erhalten Sie nur noch tägliche Snapshots des Patch-O-Matic-Subversion-Servers zum Download. Entweder Sie checken den Quelltext aus dem Subversion-Server aus, oder Sie laden einen derartigen Snapshot herunter. Die Snapshots finden Sie täglich aktualisiert auf <http://ftp.netfilter.org/pub/patch-o-matic-ng/snapshot/>.

Wenn Sie den Quelltext aus dem Subversion-Server auschecken möchten, können Sie die folgenden Befehle nutzen:

```
$ svn co https://svn.netfilter.org/netfilter/trunk/patch-o-matic-ng
...
A patch-o-matic-ng/pom2patch/pom2patch
Ausgecheckt, Revision 4354.
```

Wenn Sie später prüfen möchten, ob Updates verfügbar sind, wechseln Sie einfach in das entsprechende Verzeichnis und geben Sie folgenden Befehl ein:

```
$ svn update
Revision 4354.
```

Falls Änderungen erfolgt sind, wird Ihr Quelltext-Baum aktualisiert.

Um nun Patch-O-Matic anwenden zu können, benötigen Sie sowohl den Quelltext-Baum eines Linux-Kernels als auch des Iptables-Befehls. Patch-O-Matic muss viele Patches in beiden Software-Paketen durchführen. Den Linux-Kernel erhalten Sie auf <http://www.kernel.org>, und den Iptables-Quelltext erhalten Sie auf <http://www.Iptables.org>. Wenn Sie diese ausgepackt haben, können Sie Patch-O-Matic anwenden. Hierfür gibt es mehrere Möglichkeiten:

- `./runme pending`: Dieses Kommando wendet alle zum aktuellen Zeitpunkt bekannten Bugfixes auf den Kernel und den Iptables-Befehl an. Neue Funktionen werden nur hinzugefügt, wenn sie auch auf die Aufnahme in den Kernel warten.
- `./runme base`: Viele Patches sind zueinander inkompatibel. Dieser Befehl erlaubt Ihnen nur die Anwendung kompatibler Patches. Anschließend funktioniert noch alles.
- `./runme extra`: Hier müssen Sie wissen, was Sie tun. Dieser Befehl bietet Ihnen alle Patches an. Wenn Ihr Auto anschließend einen Kolbenfresser bekommt, ist das Ihre Schuld.
- `./runme obsolete`: Hiermit können Sie alte und überflüssige Patches anwenden. Warum sollten Sie das tun wollen?

Hinweis



Die Distributoren unter den Lesern werden diese Vorgehensweise nicht besonders mögen. Wenn Sie ein Kernel- oder Iptables-Paket bauen und einen bestimmten Patch hinzufügen möchten, bevorzugen Sie genau das, einen Patch, und nicht einen `runme`-Befehl. Hierfür gibt es den Befehl `pom2patch`. Dieser Befehl erzeugt aus Patch-O-Matic den entsprechenden Patch, den Sie mit dem `patch`-Kommando anwenden können.

Damit der Befehl `runme` die Patches auch anwenden kann, sollten Sie ihm mitteilen, wo sich der Kernel- und der Iptables-Quelltext befindet. Geben Sie beim Aufruf einfach den Ort mit den folgenden Variablen an:

18.2 Wie bekomme ich Patch-O-Matic, und wie wende ich es an?

```
$ KERNEL_DIR=/usr/local/src/linux-2.6.13 \
> IPTABLES_DIR=/usr/local/src/iptables-1.3.3 \
> ./runme pending
Loading patchlet definitions..... done
```

Welcome to Patch-o-matic (\$Revision: 4088 \$)!

```
Kernel: 2.6.13, /usr/local/src/linux-2.6.13/
Iptables: 1.3.3, /usr/local/src/iptables-1.3.3/
Each patch is a new feature: many have minimal impact, some do not.
Almost every one has bugs, so don't apply what you don't need!
```

Already applied:

```
Testing comment... not applied
The comment patch:
  Author: Brad Fisher <brad@info-link.net>
  Status: Part of 2.6.x mainline
```

This option adds CONFIG_IP_NF_MATCH_COMMENT, which supplies a comment match module. This match allows you to add comments (up to 256 characters) to any rule.

Supported options:
--comment COMMENT

Example:
-A INPUT -s 192.168.0.0/16 -m comment --comment "A privatized IP block"

Do you want to apply this patch [N/y/t/f/a/r/b/w/q/?] y

Excellent! Source trees are ready for compilation.

Recompile the kernel image (if there are non-modular netfilter modules).
Recompile the netfilter kernel modules.
Recompile the iptables binaries.

Am Ende jedes Patches können Sie entscheiden, ob dieser Patch zur Anwendung kommen soll oder nicht. Sie haben folgende Möglichkeiten zur Auswahl:

- n: No. Dies ist auch der Default, daher können Sie auch drücken.
- y: Yes. Es wird zunächst geprüft, ob der Patch anwendbar ist, und dann wird er ausgeführt. Wenn beim Test ein Fehler auftritt, wird der Patch nicht angewendet.

- t: Test. Es wird geprüft, ob der Patch anwendbar ist.
- f: Force. Der Patch wird angewendet, obwohl dabei Fehler auftreten.
- a: Restart in apply mode. Patch-O-Matic wird neu gestartet, um Patches anzuwenden.
- r: Restart in reverse mode. Patch-O-Matic wird neu gestartet, um Patches zu entfernen.
- b: Back. Einen Patch zurück.
- w: Walk. Einen Patch vorwärts.
- q: Quit. Patch-O-Matic verlassen.
- ?: Hiermit zeigt Patch-O-Matic eine Hilfe an.

Sie werden nun Patch für Patch gefragt, ob Sie den Patch anwenden möchten. Wenn Sie alle Patches durchgearbeitet haben (oben war es nur ein einziger), werden Sie aufgefordert, nun Ihren Kernel und Iptables zu übersetzen und zu installieren.

18.3 Base-Patches

Ich werde hier die Base-Patches in Patch-O-Matic vom Oktober 2005 behandeln. Wenn Sie dieses Buch lesen, hat es hier sicherlich schon Veränderungen gegeben. Vielleicht sind einige hinzugekommen und einige in den Kernel übernommen worden. Die wichtigen Patches sollen aber ein wenig ausführlicher zur Sprache kommen.

18.3.1 IPV4OPTSSTRIP

Dieser Patch fügt ein Mangle-Target `IPV4OPTSSTRIP` hinzu, das sämtliche IP-Optionen von einem Paket entfernen kann. Typische IP-Optionen sind zum Beispiel Source-Routing. Das Target verfügt über keine weiteren Optionen und kann daher sehr einfach eingesetzt werden:

```
# iptables -t mangle -A PREROUTING -j IPV4OPTSSTRIP
# iptables -t mangle -nL
Chain PREROUTING (policy ACCEPT)
target     prot opt source                destination
IPV4OPTSSTRIP all  --  0.0.0.0/0             0.0.0.0/0
```

18.3.2 connlimit

Dieser Patch fügt einen Test hinzu, mit dem Sie die Anzahl der Verbindungen zu einem Server pro Client oder Client-Netzwerk beschränken können. Der Test verfügt über zwei Optionen:

- `--connlimit-above <Limit>`. Hiermit geben Sie die maximale Anzahl der Verbindungen an.

- `--connlimit-mask <Maske>`. Hiermit beschränken Sie die Anzahl der Verbindungen nicht pro Client, sondern pro Netzwerk.

```
# Maximal 25 HTTP-Verbindungen pro Client
iptables -A FORWARD -p tcp --syn --dport 80 -m connlimit --connlimit-above 25 -j REJECT
```

```
# Maximal 100 HTTP-Verbindungen pro /24 Netzwerk
iptables -A FORWARD -p tcp --syn --dport 80 -m connlimit --connlimit-above 100 \
    --connlimit-mask 24 -j REJECT
```

18.3.3 expire

Dieser Patch gibt Ihnen die Möglichkeit, Ihren Regeln eine Lebensdauer zu geben. Sobald die angegebene Zeit abgelaufen ist, wird die Regel aus dem Regelwerk entfernt. Dies ist zum Beispiel interessant, um für einen kurzen Zeitraum einen bestimmten Zugang freizuschalten. Sie müssen nicht selbst daran denken, die Regel zu entfernen oder einen Cron-Job zu erzeugen. Der Kernel kümmert sich selbst darum.

```
iptables -A INPUT -p tcp --dport 22 -m expire --expiration +5 -j ACCEPT
```

18.3.4 NETMAP

Wenn Sie komplette Netzwerke natten möchten, ist dieser Patch möglicherweise etwas für Sie. Dieser Patch fügt ein NAT-Target hinzu, mit dem Sie sehr einfach 1:1 NAT für komplette Netzwerke implementieren können. Anstatt für jeden Rechner eine eigene Regel zu erzeugen, nutzen Sie nun eine Regel für das ganze Netz. Das `NETMAP`-Target kann sowohl als `SNAT`-Target in der `POSTROUTING`-Kette als auch als `DNAT`-Target in der `PREROUTING`-Kette angewendet werden. In dem folgenden Beispiel wird jeder IP-Adresse aus dem 1.2.3.0/24-Netz die entsprechende IP-Adresse aus dem 5.6.7.0/24-Netz zugewiesen. Das heißt, 1.2.3.10 wird 5.6.7.10, und 1.2.3.128 wird 5.6.7.128.

```
iptables -t nat -A PREROUTING -d 1.2.3.0/24 -j NETMAP --to 5.6.7.0/24
```

18.3.5 fuzzy

Dieser Patch implementiert einen Test, der die übertragenen Pakete/Sekunde mit einem Fuzzy Logic Controller prüft. Dabei ist der Test dem Limit-Test ähnlich, jedoch etwas ungenauer und somit toleranter gegenüber plötzlichen Abweichungen nach oben und unten. Er kann somit besser zur Bekämpfung von Denial-of-Service-Angriffen eingesetzt werden. Der Patch bietet die folgenden Optionen:

- `--lower-limit <packets/second>`
- `--upper-limit <packets/second>`

18.3.6 ipv4options

Dieser Patch fügt einen Test hinzu, mit dem Sie prüfen können, ob IP-Optionen in dem IPv4-Header gesetzt sind. Sie können im Einzelnen die folgenden IPv4-Optionen testen:

- `--ssrr`. Strict Source Routing.
- `--lsrr`. Loose Source Routing.
- `--no-srr`. Kein Source Routing.
- `--rr`. Record Route.
- `--ts`. Time Stamp.
- `--ra`. Router Alert.
- `--any-opt`. Mindestens eine beliebige Option.

Damit können Sie zum Beispiel Source-Routing-Pakete verwerfen:

```
iptables -t mangle -A PREROUTING -m ipv4options --ssrr -j DROP
iptables -t mangle -A PREROUTING -m ipv4options --lsrr -j DROP
```

18.3.7 nth

Dieser Patch erlaubt es Ihnen, jeweils das n-te Paket zu testen. Dabei können Sie bis zu 16 verschiedene voneinander unabhängige Zähler einsetzen. Dieser Test verfügt über einige zusätzliche Optionen:

- `--every <Nth>`. Jedes n-te Paket wird von dieser Regel ausgewertet.
- `--counter <Zahl>`. Dabei können Sie einen bestimmten Zähler (0-15) wählen. Default ist 0.
- `--start <Zahl>`. Der Zähler kann mit einer Zahl kleiner N initialisiert werden.
- `--packet <Zahl>`. Hiermit können Sie angeben, was mit jedem Paket passiert. Wenn Sie zum Beispiel `--every 2` verwenden, gibt es zwei Gruppen von Paketen, je nachdem, ob Sie bei 0 oder bei 1 anfangen zu zählen. Sie können nun zwei Regeln definieren, einmal mit `--packet 0` und einmal mit `--packet 1`, die jeweils auf diese Gruppen zutreffen.

```
iptables -t nat -A POSTROUTING -o eth0 -m nth --counter 1 \
  --every 2 --packet 0 -j SNAT --to-source 10.0.0.5
iptables -t nat -A POSTROUTING -o eth0 -m nth --counter 1 \
  --every 2 --packet 1 -j SNAT --to-source 10.0.0.6
```

18.3.8 osf

OpenBSD hat es vorgemacht, und nun ist es auch unter Linux möglich: passives Betriebssystem-Fingerprinting in den Firewall-Regeln. Sie können mit diesem Patch

Regeln aufsetzen, die nur für bestimmte Betriebssysteme gelten. Einigen Linux-Fanatikern mag die folgende Regel gefallen:

```
iptables -A FORWARD -p tcp -m osf --genre Windows -j REJECT
```

Dieser TCP-Patch kennt die Optionen:

- `--genre <os>`
- `--log <0|1>`. Wenn Sie diese Option angeben, protokolliert die Regel das Betriebssystem, wenn es nicht mit dem in der Regel angegebenen übereinstimmt. Ist der Wert 1, so wird nur das erste nicht übereinstimmende Betriebssystem protokolliert:

```
ipt_osf: Windows [2000:SP3:Windows XP Pro SP1, 2000 SP3]:
        11.22.33.55:4024 -> 11.22.33.44:139
```

- `--smart`. OSF verwendet den TTL-Wert nur, wenn das lokale Netzwerk der Ausgangsort des Pakets ist.
- `--netlink`. Hiermit kann die Protokollierung über Nlogd erfolgen.

Die notwendigen Fingerprints können Sie von <http://www.openbsd.org/cgi-bin/cvsweb/src/etc/pf.os> herunterladen. Diese Fingerprints müssen dann über das Proc-Dateisystem in den Kernel geladen werden (`/proc/sys/net/ipv4/osf`).

18.3.9 psd

Dieser Patch implementiert einen Portscan-Detektor in dem Linux-Kernel. Der Algorithmus wurde von Solar Designer's Portscan-Detektor `scanlogd` (<http://www.openwall.com>) übernommen. Ich selbst halte die Anwendung für recht kritisch, denn eine Reaktion auf einen falschen Portscan kann von einem Angreifer für einen Denial-of-Service genutzt werden. Die Erkennung durch die aktuellen Portscan-Detektoren zum Beispiel in Snort ist auch wesentlich genauer. Dennoch möchte ich kurz die Optionen auflisten und erklären.

- `--psd-weight-threshold <threshold>`. Ab diesem Schwellenwert wird ein Portscan gemeldet.
- `--psd-delay-threshold <threshold>`. Pakete, deren Abstand geringer ist als die hier angegebene Zeitspanne in Hundertstel-Sekunden, werden als Portscan-Sequenz erkannt. Dabei müssen die Pakete von der identischen Quell-IP-Adresse stammen und unterschiedliche Zielpports aufweisen.
- `--psd-lo-ports-weight <weight>`. Wenn die erkannten Portscan-Pakete an einen niedrigen Port (≤ 1024) gerichtet sind, werden die Pakete mit diesem Gewicht addiert.
- `--psd-hi-ports-weight <weight>`. Pakete an einen hohen Port werden mit diesem Gewicht addiert.

Sobald die Summe der erkannten und gewichteten Pakete den Schwellenwert überschreitet, wird ein Portscan erkannt.

18.3.10 quota

Mit diesem Patch können Sie Netzwerkquoten vergeben. Dazu definieren Sie eine Quote mit der Option `--quota <bytes>`. Jedes Paket verringert die Quote. Nach Ablauf der Quote werden keine weiteren Pakete mehr akzeptiert. Aktuelle Versionen sollten auch, entgegen der Dokumentation, auf SMP-Systemen einsetzbar sein.

18.3.11 random

Dieser Patch trifft zufällig auf Pakete zu. Sie können mit der Option `--average <Prozent>` die Wahrscheinlichkeit des Zutreffens variieren (Default: 50%). Dieser Patch ist besonders geeignet, um Fehler durch fehlerhafte Netzwerkhardware zu simulieren.

18.3.12 set

Dies ist der Nachfolger des obsoleten `pool`-Patches. Sie benötigen für die Anwendung zusätzlich den `ipset`-Befehl, den Sie ebenfalls auf der Netfilter-Homepage finden (<http://ipset.netfilter.org>). Dieser Patch erlaubt es Ihnen, Gruppen von IP-Adressen zu definieren und diese in einer Regel zu testen. Die IP-Adressen können auch von Iptables-Regeln dynamisch zu diesen Gruppen hinzugefügt werden. Dieser Patch ist so interessant und wichtig, dass er in einem eigenen Kapitel besprochen wird (Kapitel 25).

18.3.13 time

Mit diesem Patch können Sie Regeln definieren, die nur zu bestimmten Zeiten aktiv sind. Anstatt die Regelsätze per Cron austauschen zu lassen, können Sie die Uhrzeiten direkt in den Regeln hinterlegen. Zusätzlich erlaubt der Test folgende Optionen:

- `--timestart HH:MM`
- `--timestop HH:MM`
- `--days Mon,Tue,Wed,Thu,Fri,Sat,Sun`
- `--datestart YYYY[:MM[:DD[:hh[:mm[:ss]]]]]`
- `--datestop YYYY[:MM[:DD[:hh[:mm[:ss]]]]]`

Als Zeitbasis wird die lokale Uhrzeit verwendet.

18.3.14 u32

Wenn Sie immer schon mal etwas in einem Paket testen wollten, aber bisher nicht der Test dafür in Netfilter enthalten war, ist dies der Test für Sie. Hiermit können Sie ein beliebiges Bit an einer beliebigen Stelle in einem IP-Paket prüfen.

Hierzu stellt das Modul den Test `--u32` zur Verfügung. Diese Option erlaubt Ihnen die Beschreibung des Pakets in einer sehr mächtigen Sprache. Die Syntax wird im Kernel-Quelltext wie folgt angegeben:

```
--u32 tests
tests := location = value | tests && location = value
value := range | value , range
range := number | number : number
location := number | location operator number
operator := & | << | >> | @
```

Hinweis



Es gibt ein paar Beschränkungen bei der Anwendung dieses Moduls:

- Sie dürfen nur maximal 10 = (Gleichheitszeichen) und damit 9 && als Argument verwenden.
- Pro Wert (value) dürfen Sie nur maximal 10 Bereiche (range) verwenden.
- Pro Ort (location) dürfen Sie ebenfalls nur maximal 10 Nummern (number) angeben.

Es werden immer 4 Bytes von dem angegebenen Ort gelesen. Zunächst ein einfaches Beispiel. Im IP-Header befindet sich das Längelfeld in den Bytes 2 und 3 des Headers. Um nun die ersten 4 Bytes des Pakets zu lesen, verwenden Sie als Ortsangabe 0. Da Sie sich jedoch nur für die Bytes 2 und 3 interessieren, wird dieses Ergebnis mit 0x0000FFFF Und-verknüpft. Damit fallen die beiden Bytes 0 und 1 weg: 0&0xFFFF. Wenn Sie prüfen möchten, ob die Länge des Pakets kleiner als 512 Bytes ist, so können Sie dann den folgenden Test verwenden: 0&0xFFFF=0x0:0x200. Die Angabe 0x0:0x200 prüft, ob der Wert innerhalb dieses Bereichs (0x200=512) liegt.

Ein weiteres kompliziertes Beispiel prüft, ob das Byte 0-3 in den TCP-Daten den Wert 25 oder 50 hat. Der komplette Test lautet: 6&0xFF=6 && 0>>22&0x3C@12>>26&0x3C@0=25, 50. Schauen wir uns ihn der Reihe nach an. Zunächst ermitteln wir, ob es sich überhaupt um ein TCP-Paket handelt. Dies erkennen wir im IP-Header an dem Next-Protocol-Feld. Dieses Feld ist das Byte 9 im Header. Wir lesen ab dem Byte 6 die Länge von 4 Bytes und Und-verknüpfen dies mit 0xFF. Ist der Wert 6, handelt es sich um ein TCP-Paket, und wir machen weiter in dem Test.

Hinweis



Nun müssen wir eigentlich prüfen, ob es sich um ein IP-Fragment handelt. Da aber Iptables meist unter Einsatz der Connection Tracking-Funktion verwendet wird, sind die Pakete in den Tabellen Mangle, NAT und Filter bereits defragmentiert. Fragmente treffen wir hier nicht mehr an. Sie könnten aber diesen Test einfügen: 4&0x3FFF=0. Er liest ab dem Byte 4 die Länge von 4 Bytes und extrahiert die letzten 6 Bits von Byte 6 und das komplette Byte 7. Hier werden das MF-Bit und der Fragment-Offset gespeichert. Sind diese 0, so handelt es sich um ein komplettes Paket. Möchten Sie das

erste Fragment genauso prüfen wie ein komplettes Paket, müssen Sie einfach nur die letzten 5 Bits des Byte 6 analysieren: $4 \& 0x1FFF = 0$. Im ersten Fragment ist das MF schon gesetzt, aber der Offset noch null.

Um die gesuchten Bytes in den TCP-Daten zu finden, müssen Sie nun die Länge des IP-Headers und die Länge des TCP-Headers ermitteln und entsprechend im Paket weiterspringen. Leider wird die Länge im IP-Header in 4-Byte-Worten angegeben. Das bedeutet, dass Sie diese Zahl noch mit 4 multiplizieren müssen. Die Länge wird in den letzten 4 Bits des ersten Bytes gespeichert. Dazu liest der Test `0>>22&0x3c@` die Bytes 0-3 und verschiebt diese um 22 Bits nach rechts. Es bleiben 10 Bits. Wäre die Verschiebung um 24 Bits erfolgt, so wäre nur das erste Byte übrig geblieben. Da wir um 2 Bits weniger schieben, wird dieses Byte aber automatisch mit $2^2=4$ multipliziert. Jetzt muss der Test nur noch die restlichen gesetzten Bits entfernen. Hierfür führt er eine Und-Verknüpfung mit `0x3c` (0000111100) durch. Stellen Sie sich vor, der IP-Header hätte seine Standardlänge von 20 Bytes. Dann befindet sich bei einem IPv4-Paket im ersten Byte die Zahl `0x45`, binär: 01000101. Nach der Verschiebung bleiben 10 Bits: 01000101yy. Der Wert der Bits yy ist unbekannt. Nun führen Sie eine Und-Verknüpfung mit `0x3c` (0000111100) durch und es bleibt: $0000010100 = 0x14 = 20$. Das @ im Test definiert diese Zahl als neuen Offset für den nächsten Test. Dieser funktioniert analog `12>>26&0x3c@`. Die Länge des TCP-Headers wird wieder in 4-Byte-Worten in der linken Hälfte des Bytes 12 im TCP-Header angegeben. Sie lesen also ab Byte 12 die Länge von 4 Bytes und verschieben nun um 26 Bits nach rechts. Dadurch bleiben 6 Bits übrig. Die 4 höchstwertigen Bits sind automatisch mit 4 multipliziert worden. Nun setzen Sie die beiden niedrigwertigen Bits auf null, indem Sie eine Und-Verknüpfung mit `0x3c` durchführen. Dies können Sie wieder als neuen Offset verwenden. Jetzt liest der Test ab diesem Offset wieder 4 Bytes und prüft, ob deren Inhalt entweder 25 oder 50 ist ($0=25,50$).

Wie Sie sehen, können Sie mit diesem Test alle nur denkbar möglichen Tests durchführen. Um dies auf ganze Verbindungen anzuwenden, wäre es zum Beispiel möglich, in Abhängigkeit von diesem Test eine Verbindung zu markieren, um sie anschließend anders zu behandeln.

18.4 Extra-Patches

Ich werde hier die zusätzlichen Extra-Patches in Patch-O-Matic vom Oktober 2005 behandeln. Wenn Sie dieses Buch lesen, hat es hier sicherlich schon Veränderungen gegeben. Vielleicht sind einige hinzugekommen und einige in die Base-Patches oder den Kernel übernommen worden. Die wichtigen Patches sollen aber ein wenig ausführlicher zu Sprache kommen.

**Achtung**

Viele der Patches im Extra-Bereich sind nicht mit den aktuellsten Kernel lauffähig. Sie müssen im Grunde jeden einzelnen testen.

18.4.1 ACCOUNT

Dieser Patch fügt ein sehr schnelles Accounting-System dem Kernel hinzu. Hierfür benötigen Sie zusätzlich doch die `libipt_ACCOUNT`-Bibliothek, mit der Sie die Informationen anschließend auslesen können. In dem Paket ist auch das `ipaccount`-Werkzeug enthalten, mit dem Sie die Zahlen auf der Kommandozeile auswerten können.

Der Patch fügt ein `ACCOUNT`-Target dem Kernel hinzu. Dieses Target kennt zwei Optionen:

- `--addr <network/mask>`: Für dieses Netzwerk wird die Zählung durchgeführt.
- `--tname <name>`: Hier werden die Zahlen gespeichert.

Weitere Informationen über den Patch und die zugehörigen Pakete erhalten Sie auf http://www.intra2net.com/opensource/ipt_account.

18.4.2 IPMARK

Die Firewall-Markierung ist eine häufige Methode, um Pakete verschiedenen Quality-of-Service-Queues zuzuordnen. Bei einem Service-Provider ist es häufig erforderlich, jedem Client eine eigene Queue zuzuordnen, in der dessen Verkehr reglementiert wird. Handelt es sich um viele Clients, ist das sehr aufwendig, da Sie für jede IP-Adresse eine eigene Regel definieren müssen. Dieses Target erleichtert die Definition ungemein, da es die Markierung direkt aus der IP-Adresse erzeugen kann. Statt 50 Regeln für 50 Clients genügt dann häufig eine einzige Regel.

Das `IPMARK`-Target unterstützt in der Mangle-Tabelle noch die folgenden Optionen:

- `--addr src|dst`. Diese Option definiert, ob die Quell- oder die Ziel-IP-Adresse für die Erzeugung der Markierung genutzt wird.
- `--and-mask <mask>`. Vor Erzeugung der Markierung kann die IP-Adresse mit dieser Maske Und-verknüpft werden.
- `--or-mask <mask>`. Zusätzlich kann die IP-Adresse mit dieser Maske Oder-verknüpft werden. Es wird immer zuerst die Und- und dann die Oder-Verknüpfung durchgeführt.

Wenn Sie zum Beispiel ein Client-Netzwerk mit den IP-Adressen `192.168.0.0/24` verwenden und für jeden Client eine eigene Markierung zwischen 0 und 255 setzen möchten, können Sie die folgende Zeile verwenden:

```
iptables -t mangle -A PREROUTING -i eth1 -j IPMARK --addr=src  
--and-mask=0xff
```

18.4.3 ROUTE

Auch bei diesem Patch handelt es sich um einen Patch, der umständliche Konfigurationen, die über eine Firewall-Markierung eine andere Routing-Tabelle auswählen, überflüssig macht. Sie können mit diesem Patch direkt in der Mangle-Tabelle die Route eines Pakets ändern.

Dieser Patch unterstützt die folgenden Optionen:

- `-oif <ethX>`. Das Paket verlässt das System über diese Netzwerkkarte.
- `--iif <ethX>`. Das Paket kommt scheinbar über diese Netzwerkkarte an der Firewall an.
- `--gw <ip-address>`. Das Paket wird über diesen Router weitergeleitet.
- `--continue`. Normalerweise werden die weiteren Regeln nach dem `ROUTE`-Target nicht mehr betrachtet. Hiermit wird dieses Verhalten umgedreht und werden weitere Regeln ausgewertet.
- `--tee`. In diesem Fall wird eine Kopie des Pakets entsprechend der Regel geroutet. Das originale Paket wird aber den Regeln unverändert unterworfen.



Achtung

Die drei Optionen `--tee`, `--continue` und `--iif` können nicht gleichzeitig in derselben Regel auftauchen!

Um zum Beispiel den gesamten ausgehenden SMTP-Verkehr über ein anderes Gateway zu routen, können Sie den folgenden Befehl verwenden:

```
iptables -t mangle -A POSTROUTING -p tcp --dport 25 -j ROUTE --gw 1.2.3.4
```

18.4.4 TARPIT

Dieser Patch erzeugt ein neues Target `TARPIT`, das ähnlich wie LaBreas Tarpit eingesetzt werden kann. Dies ist zum Beispiel sinnvoll, um Würmer in ihrer Ausbreitung zu verlangsamen oder einen Port-Scanner zu verwirren. Dieses Target akzeptiert TCP-Verbindungen, indem es einen kompletten TCP-Handshake durchführt. Anschließend wird von Tarpit für die Verbindung ein TCP-Window von null übermittelt. Das weist den Client an, die Datenübertragung einzustellen. Der Client wird anschließend in regelmäßigen Abständen fragen, ob er fortfahren kann. Tarpit wird dies und auch einen Verbindungsabbau ablehnen. Der Client muss auf den Timeout der Verbindung warten.

**Tipp**

Wenn Sie `TARPIT` verwenden, sollten Sie alle Verbindungen, die Sie später mit `TARPIT` verzögern möchten, bereits in der `raw`-Table aus dem Connection Tracking entfernen. Ansonsten speichert der Kernel unnötigerweise in seiner Zustandstabelle die Verbindung ab und überwacht sie. Die Manpage gibt hier ein gutes Beispiel.

18.4.5 TRACE

Für die Fehlersuche in Ihren Regelsätzen ist das `TRACE`-Target eine interessante Funktion. Hiermit können Sie in der `Raw`-Tabelle ein Paket für den Trace auswählen. Anschließend protokolliert das System bei jeder auf dieses Paket zutreffenden Regel die Tabelle, die Kette und die Regelnummer. Die `raw`-Tabelle muss zur Verfügung stehen, um dieses Target zu verwenden.

18.4.6 XOR

Mit diesem Patch wird ein neues Target `XOR` für die Mangle-Tabelle eingeführt, mit dem Sie eine einfache XOR-»Verschlüsselung« durchführen können.

Hierzu akzeptiert dieses Target zwei Optionen:

- `--key <string>`
- `--block-size <blockgröße>`

18.4.7 account

Dieser Patch ähnelt dem `ACCOUNT`-Patch. Während Letzterer ein Target hinzufügt, implementiert dieser Patch einen Paketttest. Die Erweiterung `account` erlaubt die folgenden zusätzlichen Optionen:

- `--aname <name>`. Name des Zählers.
- `--aaddr <net/mask>`. Adresse, für die das Accounting durchgeführt werden soll.
- `--ashort`. Das Accounting implementiert getrennte Zähler für jedes Protokoll (TCP, UDP, ICMP und Other). Mit diesem Schalter werden alle Protokolle gemeinsam in einem Zähler geführt.

Die Zähler können anschließend über das Sysctl-Interface unter `/proc/net/ipt_account/<name>` ausgelesen und auch mit Startwerten gesetzt werden.

Weitere Informationen über den Patch erhalten Sie auf http://www.barbara.eu/~quaker/ipt_account/.

18.4.8 condition

Mit diesem Patch für den 2.4.x-Kernel können Sie selbst Variablen in dem Verzeichnis `/proc/net/iptables` erzeugen und diese in Regeln auswerten:

```
echo 1 > /proc/net/iptables/web_ok
```

```
iptables -A INPUT -p tcp -m condition --condition web_ok --dport 80 -j ACCEPT
```

18.4.9 connrate

Mit diesem Patch fügen Sie zu Ihrem Kernel einen Test hinzu, der die Datenrate der Verbindungen überwacht und es Ihnen erlaubt, Verbindungen anhand dieser Datenrate zu identifizieren. Dazu definieren Sie eine Unter- und eine Obergrenze, und dieser Test wird auf alle Verbindungen innerhalb dieses Bereichs zutreffen. Sie können die Raten nach Anwendung dieses Patches auch in der Datei `/proc/net/ip_conntrack` nachvollziehen.

Eine sinnvolle Anwendung ist zum Beispiel die Reklassifizierung von Verbindungen, die besonders viele Daten übertragen.

18.4.10 geoip

Mit diesem Patch können Sie IP-Adressen in Abhängigkeit des Ortes prüfen. Hierfür benötigen Sie die GeoIP-Datenbank. Diese Datenbank enthält Informationen, wo auf der Welt welche IP-Adressen verwendet werden. Diese Informationen können Sie nutzen, um zu testen, woher und wohin ein Paket transportiert wird:

```
iptables -A FORWARD -m geoip --source-country DE,US -j ACCEPT
```

Weitere Informationen über die Anwendung erhalten Sie auf <http://people.netfilter.org/peejix/geoip/howto/geoip-HOWTO.html>.

18.4.11 goto

Ein häufiges Problem bei der Anwendung von benutzerdefinierten Ketten ist die Tatsache, dass nach deren Abarbeitung ein Rücksprung in die aufrufende Kette erfolgt. Wenn Sie das nicht möchten, müssen Sie explizit an das Ende der benutzerdefinierten Kette eine Catch-All-Regel anfügen. Mit diesem Patch können Sie statt eines Jumps nun ein Goto durchführen. Auf diesen erfolgt kein Rücksprung! Wenn Sie eine benutzerdefinierte Kette abarbeiten, wird am Ende kein Rücksprung durchgeführt!

18.4.12 h323-contrack-nat

Wenn Sie die H.323-Protokolle verwenden (z.B. Microsoft Netmeeting oder Gnomemeeting), dann wollen Sie wahrscheinlich auf Ihrer Firewall auch diesen Patch verwenden. Hiermit können Sie die sehr komplizierten Protokolle, bei denen

viele dynamische Ports ausgehandelt werden, recht einfach filtern und natten. Allerdings unterstützt dieser Patch kein H.245-Tunneling und kein H.225-RAS. Für eine minimale Netmeeting-Funktion genügt üblicherweise nach dem Laden der neuen Module die Berücksichtigung des Ports 1720/tcp.

18.4.13 ip_queue_vwmark

Mit diesem Patch können Sie die Firewall-Markierung eines Pakets in Userspace modifizieren. Hierzu wird das Paket über das QUEUE-Interface an einen Prozess im Userspace gegeben, der das Paket prüfen und markieren kann. Diese Funktionalität wird von der NuFW (<http://www.nufw.org>) genutzt, die spezielle Paketfilterregeln für einzelne Benutzer an- und abschalten kann.

18.4.14 ipp2p

Hiermit können Sie Peer-to-Peer-Verkehr testen. Die Funktion ist in Abschnitt 32.7 erläutert.

18.4.15 policy und IPsec-Patches

Es gibt eine Reihe von Patches in Patch-O-Matic, die den IPsec-Verkehr betreffen. Diese werden in dem entsprechenden Kapitel (siehe Kapitel 33) besprochen.

18.4.16 mms-contrack-nat

Dieser Patch fügt Helfermodule hinzu, die das crosft-Streaming-Media-Protokoll analysieren und neu ausgehandelte Verbindungen der Verbindungstabelle als Expectation mit dem Zustand RELATED hinzufügen.

Nach dem Laden der entsprechenden Module ist es ausreichend, den Port 1755 sowohl für TCP als auch für UDP zu öffnen.

18.4.17 mport

Für alte Kernel < 2.6.11 fügt dieser Patch die Möglichkeit hinzu, bei der Benutzung von `-m multiport` nicht nur einzelne Ports, sondern auch Port-Bereiche zu verwenden:

```
iptables -A FORWARD -p tcp -m mport --ports 21:23,80 -j ACCEPT
```

Ab dem Kernel 2.6.11 ist diese Funktion in `multiport` enthalten.

18.4.18 owner-socketlookup

Mit diesem Patch können Sie den `owner`-Match auch in der INPUT-Kette verwenden. Sie können prüfen, welcher Benutzer das Paket entgegennehmen wird.

18.4.19 ptp-contrack-nat

Mit diesem Patch ist es möglich, Point-to-Point-Protokoll-Verkehr sicher zu filtern und zu natten. Das PPTP-Protokoll verwendet ähnlich dem FTP-Protokoll ausgehandelte Verbindungen. Die Steuerungsverbindung verwendet immer den Port 1723/tcp. Über diese Verbindung handeln Client und Server einen GRE-Tunnel aus. Mit diesem Patch können Sie den GRE-Tunnel automatisch mit dem Zustand RELATED akzeptieren. Netfilter erkennt dann selbstständig über ein Helfermodul den richtigen GRE-Tunnel. Außerdem können Sie auch das GRE-Protokoll natten. Dies ist schwierig, da GRE nicht über Ports verfügt und dadurch die Zuordnung mehrerer GRE-Tunnel zu verschiedenen Clients problematisch ist.

Nach Anwendung sind die folgenden zusätzlichen Module verfügbar, die Sie für die Funktion laden müssen:

```
ip_contrack_proto_gre
ip_contrack_ptp
ip_nat_proto_gre
ip_nat_ptp
```

Dann können Sie mit den folgenden Regeln PPTP zulassen:

```
iptables -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -p tcp --dport 1723 -m state --state NEW -j ACCEPT
```

18.4.20 quake3-contrack-nat

Dieser Patch funktioniert ähnlich dem PPTP-Patch. Auch dieser Patch fügt zwei neue Helfermodule (für Contrack und NAT) hinzu und erlaubt dann automatisch die Akzeptanz der ausgehandelten Quake3-Verbindungen. Für die Funktion müssen Sie anschließend nur den Zugriff auf den Port 27950/udp erlauben.

18.4.21 rpc

Dieser Patch fügt sowohl Helfermodule als auch Tests hinzu, mit denen Sie Unix-RPC-Verbindungen (UDP und TCP) sicher filtern können. Allerdings sollten Sie grundsätzlich kein RPC über das Internet einsetzen. Der Patch kann dennoch Sinn machen, wenn Sie in Ihrem lokalen Netz NFS oder NIS einsetzen und Ihre Server mit einer Firewall schützen möchten.

18.4.22 rsh

Dieser Patch fügt Helfermodule für das RSH-Protokoll hinzu. Die Remote Shell erlaubt die Arbeit auf einem anderen System ähnlich Telnet und der Secure Shell. Allerdings ist RSH nicht verschlüsselt. Einige Anwendungen benötigen es jedoch (z.B. Legato NetWorker Backup).

18.4.23 sip

Dieser Patch fügt Helfermodule für das SIP-Protokoll hinzu (siehe Abschnitt 32.22).

18.4.24 talk-contrack-nat

Dieser Patch fügt Helfermodule für das Talk- (517/udp) und NTalk/NTalk2-Protokoll hinzu. Damit können Talk-Verbindungen in beiden Richtungen auch durch Firewalls und NAT aufgebaut werden.

18.4.25 tproxy

Dieser Patch implementiert eine sehr interessante Funktion: einen transparenten Proxy. Ein echter transparenter Proxy ist ein Proxy, der sowohl für den Client als auch den Server unsichtbar ist. Viele Implementierungen bezeichnen bereits eine Firewall als transparenten Proxy, wenn sie lediglich für den Client unsichtbar ist. Dies genügt jedoch manchmal nicht, da die Adresse des echten Clients für den Server verloren geht. Stellen Sie sich eine Firewall vor, die einen Webserver mittels eines transparenten Proxys schützen möchte. Da der Proxy nur für den Client und nicht für den Webserver unsichtbar ist, werden Sie bei der Auswertung der Protokolle des Webserver nur Zugriffe Ihres Proxys erkennen. Bei einem echt transparenten Proxy, wie Sie ihn mit diesem Patch implementieren können, ist das nicht der Fall. Die Verbindung des Proxys mit dem Server wird mit der originalen IP-Adresse des Clients wieder aufgebaut. Die Informationen für die NAT-Zuordnungen werden in einer eigenen Tabelle `tproxy` abgespeichert. Sie wählen die Verbindungen, die der transparente Proxy übernehmen soll, mit dem `TPROXY`-Target aus.

18.4.26 unclean

Der `unclean`-Test prüft viele verschiedene Eigenschaften eines IP-Pakets und entscheidet, ob dieses Paket den Regeln und Standards entspricht oder irgendwelche Regeln missachtet. Da in der Vergangenheit beim 2.4-Kernel größere Probleme auftraten, bei denen korrekte Pakete verworfen wurden, ist dieser Test nun nicht mehr Bestandteil des Standard-Kernels, sondern Teil von Patch-O-Matic. Wenn Sie den Patch angewendet haben, können Sie fehlerhafte Pakete ganz einfach verwerfen:

```
iptables -t FORWARD -m unclean -j DROP
```

Im Einzelnen führt dieses Modul die folgenden Tests durch:

- ICMP-spezifisch:
 - Ist der Header komplett?
 - Enthält das ICMP-Fehler-Paket das Paket, das den Fehler auslöste?
 - Verwendet das ICMP-Paket einen ungültigen ICMP-Code?
 - Ist das ICMP-Paket zu groß?
 - Bezieht sich die Parameter-Problem-Fehlermeldung tatsächlich auf den IP-Header des auslösenden Fehlerpakets?

- Wurde die Source-Quench- oder Time-Exceeded-Fehlermeldung von einem Router gesendet? Router verwenden diese Fehlermeldungen nicht mehr!

UDP-spezifisch:

- - Stimmt die Länge des Pakets?
 - Zielport 0 ist nicht erlaubt.
- TCP-spezifisch:
 - Ist das Paket kleiner als die Mindestgröße des TCP-Headers?
 - Wenn es sich um einen in einer ICMP-Fehlermeldung eingebetteten TCP-Header handelt, müssen die Ports vorhanden sein.
 - Port 0 ist nicht erlaubt.
 - Sind die reservierten Bits im TCP-Header tatsächlich ungenutzt? Dieser Test kennt bereits die ECN-Bits und berücksichtigt sie.
 - Werden nur erlaubte TCP-Flag-Kombinationen verwendet? Die erlaubten Kombinationen sind: SYN, SYN/ACK, RST, RST/ACK, RST/ACK/PSH, FIN/ACK, ACK, ACK/PSH, ACK/URG, ACK/URG/PSH, FIN/ACK/PSH, FIN/ACK/URG und FIN/ACK/URG/PSH.
 - Werden die TCP-Optionen in der richtigen Reihenfolge und Länge angegeben?

Allgemeine Prüfungen:

- - Stimmt die Länge des IP-Pakets?
 - Weisen die Fragmente die richtige Länge auf, und bestehen sie aus Vielfachen von 8 Byte?
 - Ist die Länge der zusammengesetzten Fragmente länger als 64 kByte (Ping of Death)?
 - Werden die Fragment-Flags in ungültiger Kombination genutzt, oder ist die Länge eines Fragments null?
 - Ist das erste Fragment mindestens 128 Bytes lang (AX25)?
 - Ist das Next-Protocol-Feld gesetzt?

Alle diese Tests manuell (zum Beispiel mit dem `u32`-Test) zu implementieren, ist sehr aufwendig und verbraucht auch extrem viel Prozessorzeit. Hier ist `unclean` wesentlich einfacher in der Anwendung. Sie sollten die Pakete, die von diesem Test erkannt werden, mindestens protokollieren, vielleicht auch verwerfen.