Netfilter Erweiterungen HOWTO

Fabrice MARIE <fabrice@netfilter.org>, Mailing Liste netfilter-devel@lists.samba.org. Deutsche Übersetzung von Thomas Knop <tknop@maxrelax.de>
$Revision: 550 $

Dieses Dokument beschreibt wie die aktuellen iptables Erweiterungen installiert und benutzt werden.

Contents

1 Einleitung

Zunächst ein paar Worte zur deutschen Übersetzung dieses HOWTO. Es ist teilweise extrem schwierig englische Fachbergriffe wirklich sinnvoll zu übersetzen. Da ich davon ausgehe, dass der geneigte Leser wenigstens ein Grundverständniss von Netzwerken, Protokollen und natürlich Netfilter hat, habe ich Worte wie "conntrack" oder "match" nicht mit "Verbindungsverfolgung" oder "Übereinstimmung finden" übersetzt. Falls jemand Mängel in der Übersetzung feststellt oder Verbesserungen vorschlagen möchte, freue ich mich eine Email zu erhalten.

Hallo. Dies ist eine gute Möglichkeit mich bei all den Leuten zu bedanken, die sehr viel Zeit in die Entwicklung, das Testen, das Melden von Fehlern und das Benutzen von Netfilter investiert haben.

Also, danke euch allen !!

Dieses HOWTO setzt voraus, dass du Rusty's

Linux 2.4 Packet Filtering HOWTO http://www.netfilter.org/documentation/HOWTO/de/packet-filtering-HOWTO.html gelesen und verstanden hast. Es wird ebenfalls vorausgesetzt, dass du weiSSt wie man einen Linux Kernel richtig kompiliert und installiert.

Die `iptables` Distribution enthält Erweiterungen, die nicht von regulären Benutzern verwendet werden, die noch recht experimentell sind oder die zu guter Letzt noch nicht in den offiziellen Kernel eingeflossen sind.

Diese Erweiterungen sind im Normalfall nicht mit kompiliert, es sei denn du hast das veranlaSSt.

Du findest die aktuellste Version dieses Dokumenten auf der *Netfilter Dokumentation* http://www.netfilter.org/documentation/index.html#HOWTO Homepage.

Das Ziel dieses HOWTO soll es sein Leuten zu helfen, die mit den Netfilter Erweiterung gerade anfangen. Hierzu wird erklärt, wie man die Erweitrungen installiert und rudimentär wie man sie nutzt.

2 Patch-O-Matic

2.1 Was is Patch-O-Matic ?

Das `iptables` Makefile enthält eine Funktionalität die sich 'patch-o-matic' (oder 'p-o-m') nennt. p-o-m erleichtert dir die Patches auszuwählen, die du möchtes, und patcht dann automatisch den Kernel für dich.

Zunächst solltest du dir die aktuellen CVS Sourcen holen. Damit ist sicher gestellt, dass du die jüngsten Erweiterungen benutzt. Um das zu erledigen, führe folgende Schritte durch:

```
# cvs -d :pserver:cvs@pserver.netfilter.org:/cvspublic login

(Wenn du nach dem Passwort gefragt wirst, gebe 'cvs' an).

# cvs -d :pserver:cvs@pserver.netfilter.org:/cvspublic co netfilter/userspace netfilter/patch-o-matic
```

Es wird ein Verzeichnis 'netfilter/' erstellt und alle Dateien werden innerhalb dieses Verzeichnisses für dich ausgescheckt.

Stelle sicher, dass sich deine Kernelsourcen in '/usr/src/linux/' befinden. Falls - aus welchen Gründen auch immer - der Kernel, den du patchen willst, sich nicht in'/usr/src/linux/' befindet, setze die Variable KERNEL_DIR so das sie auf deine Kernelsourcen zeigt:

```
# export KERNEL_DIR=/das/verzeichis/linux
```

Stelle sicher, dass die Abhängigkeiten in den Kernelsourcen stimmen. Wenn du unsicher bist:

```
# cd /usr/src/linux/
# make dep
```

Nun kannst du zurück zum Netfilter Verzeichnis in das Unterverzeichnis ‘userspace/’ wechseln. Jetzt ist alles vorbereitet damit du p-o-m auszuführen kannst.

2.2 Patch-O-Matic ausführen

Wenn du im Unterverzeichnis ‘userspace/’ bist, starte p-o-m:

```
# make patch-o-matic

Welcome to Rusty's Patch-o-matic!

Each patch is a new feature: many have minimal impact, some do not.
Almost every one has bugs, so I don't recommend applying them all!
-------------------------------------------------------

Already applied: 2.4.1 2.4.4
Testing... name_of_the_patch NOT APPLIED ( 2 missing files)
The name_of_the_patch patch:
   Here usually is the help text describing what
   the patch is for, what you can expect from it,
   and what you should not expect from it.
Do you want to apply this patch [N/y/t/f/q/?]
```

Patch-o-matic wird dann alle Patche durchgehen. Wenn ein Patch bereits installiert ist, siehst du das an der ersten Zeile ‘Already applied:’. Ist ein Patch noch nicht installiert, wird dir p-o-m den Namen des Patches und einige Erklärungen anzeigen. p-o-m teilt dir den Zustand mit: ‘NOT APPLIED (n missing files)’ meint schlicht, dass der Patch noch nicht installiert ist, während ‘NOT APPLIED (n rejects out of n hunks)’ generell meint:

1. Entweder das Patchen konnte nicht sauber durchgeführt werden ...
2. ...Oder der Patch ist bereits in dem Kernel enthalten, den du versuchst zu patchen.

Zuletzt wird du immer aufgefordert zu entscheiden, ob der Patch installiert werden soll oder nicht.

- Drücke einfach Enter (Eingabetaste), wenn du den Patch nicht installieren willst.
- Gebe ‘y’ ein, wenn du möchtest, dass p-o-m den Patch testet und installiert. Wenn der Test nicht erfolgreich ist, wird es dir mitgeteilt und du wirst gefragt ob du den Patch dennoch installieren möchtest (conformation). Ist der Test erfolgreich, wird der Patch installiert und du wirst den Namen des Patches in der Zeile ‘Already applied’ sehen.
- Gebe ‘t’ ein, wenn du zunächst nur testen möchtest, ob sich der Patch normal installieren läSSt.
- Gebe ‘f’ ein, um p-o-m zu zwingen den Patch zu installieren.
- Gebe ‘q’ ein, um p-o-m zu beenden.

Als Daumenregel gilt die kurzen Erklärungen zu den Patchen aufmerksam zu lesen bevor du einen Patch installierst. Da es derzeit SEHR VIELE offizielle Patche für patch-o-matic gibt (und wahrscheinlich noch mehr inoffizielle), ist es nicht empfohlen alle zu installieren! Du solltest wirklich nur die Patches installieren, die du jetzt benötigst, auch wenn das bedeutet, dass du Netfilter neu kompilieren muSSt, wenn du einmal mehr Patches benötigst.

Anmerkung des Übersetzers: Du solltest einen Patch nur dann installieren, wenn du die Erklärung zum Patch verstanden hast und dir sicher bist, dass du dieses Feature unbedingt brauchst!

Es wurde eine neue Variante von patch-o-matic entwickelt, die dir nur solche Patche anzeigt, von denen bekannt ist, dass sie sich sauber installieren lassen oder aber mindestens von denen bekannt ist, dass sie keine anderen Patche negativ beinflussen. Um diese Installationsart auszuführen, führe folgendes aus:

```
# make most-of-pom
```

Es arbeitet genauso wie patch-o-matic was die Bedienung und das Patchen angeht, es überspring schlicht die Entwickler-Patches (‘developer-only’).

Patch-o-matic an sich ist einfach nur ein Shellscript im Unterverzeichnis ‘patch-o-matic/’ und heiSSt ‘runme’. Du kannst in das Verzeichnis wechseln und es von Hand aufrufen. Rufst du es ohne Parameter auf, wird es dir eine Hilfeseite ausgeben:

```
Usage: ./runme [--batch] [--reverse] [--exclude suite/patch-file ...] suite|suite/patch-file

  --batch       batch Modus, automatisches installieren der Patches
  --reverse     ausgewählte Patche deinstallieren
  --exclude     die benannten Patche auslassen
```

Die Patche befinden sich in den Verzeichnissen ‘patch-o-matic/pending/’, ‘patch-o-matic/base’, u.s.w... Dabei sind ‘pending’ und ‘base’ zwei Suitenamen. Führe ein ls im Verzeichnis ‘patch-o-matic’ aus, um alle Suites zu sehen. Beispiele für den Aufruf von ‘runme’:

```
./runme --batch pending
./runme --batch userspace/ipt_REJECT-fake-source.patch
```

Das erste Kommando installiert alle Patche der Suite pending (die Reihenfolge ist in der Datei ‘pending/SUITE’ festgelegt). Das zweite Kommando installiert nur den Patch ‘ipt_REJECT-fake-source.patch’ aus der userspace Suite.

2.3 Ok, und was als nächstes?

Nachdem du alle Patches installiert hast, die du installieren wolltest, ist der nächste Schritt das Neukompilieren und Installieren des Kernels. Diese HOWTO erklärt nicht wie das zu tun ist. Anstelle dessen kannst du das *Linux Kernel HOWTO* http://www.ibiblio.org/pub/Linux/docs/HOWTO/translations/de/aktuell/html/DE-Kernel-HOWTO.html.tar.gz lesen.

Beim Konfigurieren deines Kernels, werden dir die neuen Optionen unter “Networking Options -> Netfilter Configuration” angezeigt. Wähle die von dir benötigten Optionen aus, kompiliere & installiere deinen neuen Kernel.

Ist der Kernel erstmal installiert kannst du mit dem Kompilieren und Installieren des “iptables” Paketes aus dem Verzeichnis ‘userspace/’ wie folgt fortfahren:

```
# make && make install
```

Das war's! Deine neues iptables Packet ist installiert! Jetzt ist es an der Zeit die brandneuen Funktionalitäten zu benutzen.

3 Neue Netfilter Matches

In diesem Abschnitt werden wir versuchen die Benutzung der neuen Netfilter Matches zu erklären. Die Patche werden in alphabetischer Reihenfolge besprochen. Wir gehen hier nicht auf Patche ein, die andere Patche negativ beeinflussen; das kommt später mal.

Generell gilt für Matches, dass du die Hilfe für ein bestimmtes Modul wie folgt abfragen kannst:

```
# iptables -m match_den_du_willst --help
```

Das zeigt dir die normale Hilf von iptables an und zuzüglich am Ende die spezielle Hilfe zu "match_den_du_willst".

3.1 ah-esp Patch

Dieser Patch von Yon Uriarte <yon@astaro.de> fügt zwei neue Matches hinzu:

- "ah" : erlaubt das Matchen eines AH Packetes basierend auf dem Security Parameter Index (SPI).
- "esp" : erlaubt das Matchen eines ESP Paketes basierend auf dem SPI.

Dieser Patch kann recht brauchbar sein für Leute die IPSEC benutzen und die Verbindungen an Hand des SPI unterscheiden möchten.

Als Beispiel wollen wir alle AH Pakete verwerfen, die einen SPI von 500 haben:

```
# iptables -A INPUT -p 51 -m ah --ahspi 500 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       ipv6-auth--  anywhere             anywhere           ah spi:500
```

Unterstützte Optionen für den ah Match sind:

–ahspi [!] spi[:spi]

-> matcht SPI (Bereich)

Der esp Matche arbeitet ebenso:

```
# iptables -A INPUT -p 50 -m esp --espspi 500 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       ipv6-crypt--  anywhere             anywhere           esp spi:500
```

Unterstützte Optionen für den esp Match sind:

–espspi [!] spi[:spi]

-> matcht SPI (Bereich)

Es ist wichtig, dass du nicht vergisst das passende Protokoll mit “-p 50” or “-p 51” anzugeben (für jeweils ESP & AH), wenn du einen dieser Matche verwendest. Ansonsten bricht das Einfügen der Regel aus offensichtlichen Gründen ab.

3.2 conntrack Patch

Dieser Patch von Marc Boucher <marc+nf@mbsi.ca> fügt ein conntrack Match Modul hinzu (eine Obermenge des state Match), welches dir erlaubt weitere Conntrack Informationen zu matchen.

Wenn du Beispielsweise alle RELATED Verbindungen für das TCP Protokoll erlauben möchtest, kannst du wie folgt vorgehen:

```
# iptables -A FORWARD -m conntrack --ctstate RELATED --ctproto tcp -j ACCEPT

# iptables --list
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere           ctstate RELATED
```

Unterstützte Optionen für den conntrack Match sind:

[!] –ctstate [INVALID|ESTABLISHED|NEW|RELATED|SNAT|DNAT][,...]

-> Match auf Zustand/Zustände. Die neuen ‘SNAT’ and ‘DNAT’ Zustände sind virtuell die dann zutreffen, wenn die ursprüngliche Quelladresse sich vom Antwortziel oder die ursprüngliche Zieladresse sich von der Antwortquelle unterscheiden.

[!] –ctproto proto

-> Protokoll; als Nummber oder Name, z.B. ‘tcp’.

–ctorigsrc [!] address[/mask]

-> Spezifikation der ursprünglichen Quelladresse.

–ctorigdst [!] address[/mask]

-> Spezifikation der ursprünglichen Zieladresse.

–ctreplsrc [!] address[/mask]

-> Spezifikation der Antwortquelle.

–ctrepldst [!] address[/mask]

-> Spezifikation des Antwortzieles.

[!] –ctstatus [NONE|EXPECTED|SEEN_REPLY|ASSURED][,...]

-> match auf Status/Staten.

[!] –ctexpire time[:time]

-> Match der restlichen “Lebenszeit” (lifetime) in Sekunden gegen eine Wert oder einen Bereich (die Randwerte sind einbezogen).

3.3 iplimit Patch

Dieser Patch von Gerd Knorr <kraxel@bytesex.org> erlaubt es dir die Anzahl der parallelen TCP Verbindungen von einem bestimmten Host oder Netzwerk zu bergenzen.

Als Beispiel wollen wir die Anzahl der parallelen HTTP Verbindungen von einer einzelnen IP Adresse auf 4 begrenzen:

```
# iptables -A INPUT -p tcp --syn --dport http -m iplimit --iplimit-above 4 -j REJECT

# iptables --list
Chain INPUT (policy ACCEPT)
target   prot opt source    destination
REJECT   tcp  --  anywhere  anywhere     tcp dpt:http flags:SYN,RST,ACK/SYN #conn/32 > 4 reject-with icmp-port-unreachable
```

Oder du möchtest eventuell die Anzahl paralleler Verbindungen einer ganzen Gruppe von Host (Netzmaske 255.255.255.0, ehm. Class C Netz) begrenzen:

```
# iptables -A INPUT -p tcp --syn --dport http -m iplimit --iplimit-mask 24 --iplimit-above 4 -j REJECT

# iptables --list
Chain INPUT (policy ACCEPT)
target   prot opt source    destination
REJECT   tcp  --  anywhere  anywhere     tcp dpt:http flags:SYN,RST,ACK/SYN #conn/8 > 4 reject-with icmp-port-unreachable
```

Unterstützte Optionen für den iplimit Match sind:

[!] –iplimit-above n

-> matcht, wenn die Anzahl der existierenden TCP Verbindungen (nicht) gröSSer als n ist.

–iplimit-mask n

-> Guppierung von Hosts per Maske.

3.4 ipv4options Patch

Dieser Patch von Fabrice MARIE <fabrice@netfilter.org> erlaubt es dir Pakete an Hand der gesetzten IP Protokoll Optionen zu matchen.

Als Beispiel verwerfen wir alle Pakete, bei denen die IP Option Record-Route oder Timestamp gesetzt ist:

```
# iptables -A INPUT -m ipv4options --rr -j DROP
# iptables -A INPUT -m ipv4options --ts -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere            IPV4OPTS RR
DROP       all  --  anywhere             anywhere            IPV4OPTS TS
```

Unterstützte Optionen für den ipv4options Match sind:

–ssrr

-> Strict Source Routing Flag gesetzt.

–lsrr

-> Loose Source Routing Flag gesetzt.

–no-srr

-> Pakete ohne Source Routing.

[!] –rr

-> Record Route Flag gesetzt.

[!] –ts

-> Timestamp Flag gesetzt.

[!] –ra

-> Router-Alert Option aktiv.

[!] –any-opt

-> prüft ob mindestens eine IP Option aktiv ist (oder keine wenn ! vorangestellt wurde).

3.5 length Patch

Dieser Patch von James Morris <jmorris@intercode.com.au> erlaubt es Pakete auf ihre Länge zu prüfen.

Als Beispiel wollen wir alle Ping's verbieten, die eine Paketlänge von mehr als 85 Bytes aufweisen:

```
# iptables -A INPUT -p icmp --icmp-type echo-request -m length --length 86:65535 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere           icmp echo-request length 86:65535
```

Unterstützte Optionen für den length Match sind:

[!] –length length[:length]

-> prüft auf entsprechende Länge. Die Werte bei einem Bereich sind enthalten.

Wird bei einem Bereich ein Wert nicht angegeben, werden die implizierten Werte 0 (Minimum) und 65535 (Maximum) verwendet.

3.6 mport Patch

Dieser Patch von Andreas Ferber <af@devcon.net> ermöglicht es in einer Regel für das UDP oder TCP Protokoll beliebige Kombination aus einzelnen Ports und Portbereichen zu verwenden.

Möchtest du zum Beispiel ftp, ssh, telnet und http mit einer einzigen Regel verwerfen, kannst du die folgende verwenden:

```
# iptables -A INPUT -p tcp -m mport --ports 20:23,80 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere           mport ports ftp-data:telnet,http
```

Unterstützte Optionen für den mport Match sind:

–source-ports port[,port:port,port...]

-> Quellports.

–sports port[,port:port,port...]

-> Synonym für –source-ports.

–destination-ports port[,port:port,port...]

-> Zielports.

–dports port[,port:port,port...]

-> Synonym für –destination-ports.

–ports port[,port:port,port]

-> Match für Quell- und Zielports.

3.7 nth Patch

Dieser Patch von Fabrice MARIE <fabrice@netfilter.org> erlaubt es das n-te Paket, welches eine Regel erreicht zu erkennen.

Wenn du zum Beispiel jedes zweite Ping Paket verwerfen willst, kannst du folgende Regel verwenden:

```
# iptables -A INPUT -p icmp --icmp-type echo-request -m nth --every 2 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere           icmp echo-request every 2th
```

Erweiterungen von Richard Wagner <rwagner@cloudnet.com> ermöglichen es schnell und einfach eine Last-Verteilung zu implementieren.

Möchtest du zum Beispiel die Last auf die 3 Adressen 10.0.0.5, 10.0.0.6 and 10.0.0.7 verteilen, kanst du wie folgt vorgehen:

```
# iptables -t nat -A POSTROUTING -o eth0 -m nth --counter 7 --every 3 --packet 0 -j SNAT --to-source 10.0.0.5
# iptables -t nat -A POSTROUTING -o eth0 -m nth --counter 7 --every 3 --packet 1 -j SNAT --to-source 10.0.0.6
# iptables -t nat -A POSTROUTING -o eth0 -m nth --counter 7 --every 3 --packet 2 -j SNAT --to-source 10.0.0.7

# iptables -t nat --list
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
SNAT       all  --  anywhere             anywhere           every 3th packet #0 to:10.0.0.5
SNAT       all  --  anywhere             anywhere           every 3th packet #1 to:10.0.0.6
SNAT       all  --  anywhere             anywhere           every 3th packet #2 to:10.0.0.7
```

Unterstützte Optionen für den nth Match sind:

–every n

-> Regel trifft auf jedes n-te Packet zu.

[–counter] num

-> Benuze Zähler 0-15 (default:0).

[–start] num

-> Initialisiere den Zähler mit Wert 'num' anstelle von 0. Der Wert muSS zwischen 0 und (n-1) liegen.

[–packet] num

-> Trifft auf das 'num'-te Packet zu. Der Wert muSS zwischen 0 und (n-1) liegen. Wenn '–packet' für einen Zähler benutzt wird, so müssen n '–packet' Regeln existieren um alle Werte zwischen 0 und (n-1) abzudecken.

3.8 pkttype Patch

Dieser Patch von Michal Ludvig <michal@logix.cz> ermöglicht es Pakete anhand ihres Types host/broadcast/multicast zu erkennen.

Wenn du zum Beispiel alle broadcast Pakete stillschweigend verwerfen willst:

```
# iptables -A INPUT -m pkttype --pkt-type broadcast -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere           PKTTYPE = broadcast
```

Unterstützte Optionen für diesen Match sind:

–pkt-type [!] packettype

-> trifft auf den entsprechenden Pakettyp zu. Der Pakettyp kann einder der folgenden sein:

host

-> an uns

broadcast

-> an alle

multicast

-> an eine Gruppe

3.9 pool Patch

Patch von Patrick Schaaf <bof@bof.de>. Joakim Axelsson und Patrick sind gerade dabei diesen Match neu zu implementieren. Deswegen wird dieser Abschnitt mit einer aktuellen Erklärung dann aktualisiert, wenn die Neuimplementierung erfolgt ist.

3.10 psd Patch

Dieser Patch von Dennis Koslowski <dkoslowski@astaro.de> versucht Portscans zu erkennen.

In der einfachsten Version wird der psd Patch wie folgt verwendet:

```
# iptables -A INPUT -m psd -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target  prot opt source    destination
DROP    all  --  anywhere  anywhere    psd weight-threshold: 21 delay-threshold: 300 lo-ports-weight: 3 hi-ports-weight: 1
```

Unterstützte Optionen für psd Match sind:

[–psd-weight-threshold threshold]

-> Grenze ab der von einem Portscan ausgegangen wird und diese Regel zutrifft.

[–psd-delay-threshold delay]

-> Zeit innerhalb der die Portzugriffe erfolgen müssen.

[–psd-lo-ports-weight lo]

-> Gewichtung für priviligierte Ports.

[–psd-hi-ports-weight hi]

-> Gewichtung für unpreviligierte Ports.

3.11 quota Patch

Dieser Patch von Sam Johnston <samj@samj.net> gestattet es Quotas zu setzen. Wenn eine Quote erreicht wird, trifft die entsprechende Regel nicht mehr zu.

Wenn du zum Beispiel dein Limit von 50 Megabyte für den eingehenden http Traffic setzen möchtes, kannst du folgende Regeln verwenden:

```
# iptables -A INPUT -p tcp --dport 80 -m quota --quota 52428800 -j ACCEPT
# iptables -A INPUT -p tcp --dport 80 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere           tcp dpt:http quota: 52428800 bytes
DROP       tcp  --  anywhere             anywhere           tcp dpt:http
```

Unterstützte Optionen für den quota Match sind:

–quota quota

-> Das Limit, das du setzen möchtest.

3.12 random Patch

Dieser Patch von Fabrice MARIE <fabrice@netfilter.org> ermöglicht es Pakete zufällig mit eine bestimmten Wahrscheinlichkeit auszuwählen.

Wenn du zum Beispiel zufällig 50% aller Ping Anfragen verwerfen möchtest, kannst du das wie folgt tun:

```
# iptables -A INPUT -p icmp --icmp-type echo-request -m random --average 50 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source       destination
DROP       icmp --  anywhere     anywhere        icmp echo-request  random 50%
```

Unterstützte Optionen für den random Match sind:

[–average percent]

-> Die Wahrscheinlichkeit in Prozent, mit der die Regel zutreffen soll. Wenn diese Option nicht angegeben wird, wird eine Wahrscheinlichkeit von 50% gesetzt. Der Wert muss inerhalb von 1 <= percent <= 99 liegen.

3.13 realm Patch

Dieser Patch von Sampsa Ranta <sampsa@netsonic.fi> erlaubt es Pakete an Hand des "realm key" vom Routing auszuwählen; dies entspricht einer Klassifizierung von bestimmten Paketen.

Wenn du zum Beispiel alle ausgehenden Pakete loggen möchtest, die einen realm Wert von 10 haben, gehst du wie folgt vor:

```
# iptables -A OUTPUT -m realm --realm 10 -j LOG

# iptables --list
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
LOG        all  --  anywhere             anywhere           REALM match 0xa LOG level warning
```

Unterstützte Optionen für den realm Match sind:

–realm [!] value[/mask]

-> matcht den entsprechendeb realm Wert zu.

3.14 recent Patch

Dieser Patch von Stephen Frost <sfrost@snowman.net> erlaubt es dynamisch Listen mit IP Adressen zu erstellen und dann an Hand der Listen Pakete auf unterschiedliche Weise zu filtern.

Zum Beispiel kannst du eine Liste 'badgay' von Leuten erstellen, die versuchen einen Verbindung zum Port 139 (NETBIOS Session Service) deiner Firewall herzustellen und dann alle weiteren Pakete (also auch die normalerweise erlaubten) von diesen Leuten verwerfen ohne sie weiter zu betrachten.

```
# iptables -A FORWARD -m recent --name badguy --rcheck --seconds 60 -j DROP
# iptables -A FORWARD -p tcp -i eth0 --dport 139 -m recent --name badguy --set -j DROP

# iptables --list
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere           recent: CHECK seconds: 60
DROP       tcp  --  anywhere             anywhere           tcp dpt:netbios-ssn recent: SET
```

Unterstützte Optionen für den recent Match sind:

–name name

-> Spezifiziert den Namen der Liste, die für das Kommando verwendet werden soll. Wird kein Name angegeben, so wird 'DEFAULT' verwendet.

[!] –set

-> Fügt die Quelladresse des Paketes zu der Liste hinzu. Ist die Adresse bereits in der Liste, wird ein Update durchgeführt. Das Ergebnis dieser Regel ist immer true oder bei vorangestelltem ! false.

[!] –rcheck

-> Prüft ob die Quelladresse in der Liste ist und gibt in diesem Fall true, sonst false zurück. Das Ergebnis wir bei vorangestlltem ! umgekehrt.

[!] –update

-> Prüft ob die Quelladresse in der Liste ist. Ist die Adresse vorhanden, wird ein Update durchgeführt und true zurückgegeben. Ist die Adresse nicht in der Liste wird false zurückgegeben. Das Ergebnis wir bei vorangestlltem ! umgekehrt.

[!] –remove

-> Prüft ob die Quelladresse in der Liste ist. Ist die Adresse vorhanden, wird sie aus der Liste entfernt und true zurückgegeben. Ist die Adresse nicht in der Liste wird false zurückgegeben. Das Ergebnis wir bei vorangestlltem ! umgekehrt.

[!] –seconds seconds

-> Diese Option muss etweder mit der 'rcheck' oder 'update' Option zusammen angegeben werden. Sie bewirkt, dass diese Regel nur dann greift, wenn die Quelladresse innerhalb der letzten angegebenen Sekunden 'seconds' in der Liste gesehen worden ist. Das Ergebnis wir bei vorangestlltem ! umgekehrt.

[!] –hitcount hits

-> Diese Option muss etweder mit der 'rcheck' oder 'update' Option zusammen angegeben werden. Sie bewirkt, dass diese Regel nur dann greift, wenn die Quelladresse wenigstens 'hits' mal in der Liste gesehen worden ist. Die Option kann zusammen mit der '–seconds' Option verwendet werden, um eine Regel zu erstellen, die nur dann greift wenn eine Adresse mit einer bestimmten Häufigkeit innerhalb eines Zeitintervalles gesehen wird. Das Ergebnis wir bei vorangestelltem ! umgekehrt.

–rttl

-> Diese Option muss etweder mit der 'rcheck' oder 'update' Option zusammen angegeben werden. Sie bewirkt, dass diese Regel nur dann greift, wenn die Adresse in der Liste ist und die TTL des aktuellen Paketes mit der TTL des Paketes übereinstimmt, welches die –set Regel ausgelöst hat. Das kann sinvoll sein, wenn du Probleme mit Leuten hast die ihre Quelladresse fälschen und so einen DoS Angriff über diesen Modul auslösen in dem andere Adressen dadurch gesperrt werden, dass sie die gefälschte Pakete senden.

3.15 record-rpc Patch

Dieser Patch von Marcelo Barbosa Lima <marcelo.lima@dcc.unicamp.br> erlaubt es Pakete zu erkennen, wenn die Quelladdresse bereits vorher einen Port über den Portmapper angefordert hat oder diese eine neue GET Anfrage an den Portmapper ist. Das ermöglicht effektives RPC-Filtern.

Um RPC Verbindungen zu verfolgen benutze einfach folgende Regel:

```
# iptables -A INPUT -m record_rpc -j ACCEPT

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
```

Der record_rpc Match hat keine Optionen.

Wundere dich nicht, wenn keine Informationen zum Match ausgegeben werden. Das liegt daran, dass die print() Funktion dieses Matches nicht implementiert ist:

```
/* Prints out the union ipt_matchinfo. */
static void
print(const struct ipt_ip *ip,
      const struct ipt_entry_match *match,
      int numeric)
{
}
```

3.16 string Patch

Dieser Patch von Emmanuel Roger <winfield@freegates.be> erlaubt es dir nach einem String an einer beliebigen Stelle im Paket zu filtern.

Um zum Beispiel Pakete zu erkennen, die den String "cmd.exe" enthalten und diese an den Userspace weiterzuleiten kannst du folgende Regel benutzen:

```
# iptables -A INPUT -m string --string 'cmd.exe' -j QUEUE

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
QUEUE      all  --  anywhere             anywhere           STRING match cmd.exe
```

Bitte benutze diesen Match mit Vorsicht. Viele Leute wollen diesen Match benutzen um zusammen mit dem DROP Target Würmer abzuweheren. Dies ist ein grundlegender Fehler. Hierfür gibt es IDS Methoden, die wesentlich besser geeignet sind.

Auf ähnliche Weise versuchen viele Leute diesen Match dazu zu benutzen um bestimmte HTTP Funktionalitäten wie POST zu verbieten, indem sie alle Pakete verwerfen die diese String enthalten. Bitte verstehe, dass dieser Job wesentlich besser von einem filterndem Proxy übernommen werden kann. Darüber hinaus wird mit dieser Methode auch jeder HTML Inhalt verworfen der den String POST enthält. Dieser Match wurde nur entworfen um interessante Pakete an den Userspace weiterzureichen um sie dort besser zu analysieren. Pakete mit diesem Patch zu verwerfen hebelt jede andere IDS Methode aus.

Unterstützte Optionen für den string Match sind:

–string [!] string

-> String im Paket.

3.17 time Patch

Dieser Patch von Fabrice MARIE <fabrice@netfilter.org> ermölicht es Pakete anhand der Zeitmarke (timestamp) zu filtern.

Um zum Beispiel ankommende Pakete nur von 8:00 Uhr bis 18:00 Uhr von Montags bis Freitags zuzulassen kannst du folgende Regel verwenden:

```
# iptables -A INPUT -m time --timestart 8:00 --timestop 18:00 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source           destination
ACCEPT     all  --  anywhere         anywhere        TIME from 8:0 to 18:0 on Mon,Tue,Wed,Thu,Fri
```

Unterstützte Optionen für den time Match sind:

–timestart value

-> Minimum HH:MM

–timestop value

-> Maximum HH:MM

–days listofdays

-> Eine Liste von Tagen (GroSS-/Kleinschreibung beachten) aus:

- Mon
- Tue
- Wed
- Thu
- Fri
- Sat
- Sun

3.18 ttl Patch

Dieser Patch von Harald Welte <laforge@gnumonks.org> erlaubt es Pakete auf der Basis ihrer TTL zu filtern:

Um zum Beispiel alle Pakete zu loggen, die eine TTT kleiner als 5 aufweisen, kannst du folgende Regel verwenden:

```
# iptables -A INPUT -m ttl --ttl-lt 5 -j LOG

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
LOG        all  --  anywhere             anywhere           TTL match TTL < 5 LOG level warning
```

Optionen des ttl Match sind:

–ttl-eq value

-> trifft zu bei gegebener Time-to-Live.

–ttl-lt value

-> trifft zu bei TTL < value

–ttl-gt value

-> trifft zu bei TTL > value

4 Neue Netfilter Targets (Ziele)

In diesem Abschnit werden wir versuchen die Benutzung der neuen Netfilter Targets zu erklären. Die Patche werden in alphabetischer Reihenfolge besprochen. Wir werden hier nicht auf Patche eingehen, die andere Patche negativ beeinflussen; das kommt später mal.

Generell gilt für Targets, dass du die Hilfe für ein bestimmtes Modul wie folgt abfragen kannst:

```
# iptables -j TARGET_DAS_DU_WILLST --help
```

Dies wird dir die normale Hilf von iptables ausgeben zuzüglich der speziellen Hilfe zu "TARGET_DAS_DU_WILLST" am Ende.

4.1 ftos Patch

Dieser Patch von Matthew G. Marsh <mgm@paktronix.com> fügt ein neues Target hinzu, welches dir gestattet den TOS eines Paketes auf einen beliebigen Wert zu setzen.

Wenn du zum Beispiel möchtest dass der TOS für alle ausgehenden Pakete den Wert 15 hat, benutzt du folgende Regel:

```
# iptables -t mangle -A OUTPUT -j FTOS --set-ftos 15

# iptables -t mangle --list
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
FTOS       all  --  anywhere             anywhere           TOS set 0x0f
```

Unterstützte Optionen für das FTOS Target sind:

–set-ftos value

-> Setzt das TOS Feld im Paketheader auf den gegebenen Wert. Der Wert kann dezimal (z.B. `32`) oder hexadezimal (z.B. `0x20`) angegeben werden.

4.2 IPV4OPTSSTRIP Patch

Dieser Patch von Fabrice MARIE <fabrice@netfilter.org> fügt ein neues Target hinzu, welches dir gestattet alle IP Optionen von einem IPv4 Paket zu entfernen.

Es wird einfache auf die folgende Art aktiviert:

```
# iptables -t mangle -A PREROUTING -j IPV4OPTSSTRIP

# iptables -t mangle --list
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
IPV4OPTSSTRIP  all  --  anywhere             anywhere
```

Dieses Target hat keine Optionen.

4.3 NETLINK Patch

Dieser Patch von Gianni Tedesco <gianni@ecsc.co.uk> fügt ein neues Target hinzu, welches dir gestattet verworfene Pakete an den Userspace über den Netlink Socket zu senden.

Wenn du zum Beispiel alle Pings verwerfen und sie and den Userspace Netlik Socket weiterleiten möchtest, kannst du das wie folgt erreichen:

```
# iptables -A INPUT -p icmp --icmp-type echo-request -j NETLINK --nldrop

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
NETLINK    icmp --  anywhere             anywhere           icmp echo-request nldrop
```

Unterstützte Optionen für das NETLINK Target sind:

–nldrop

-> Weiterreichen und verwerfen.

–nlmark <number>

-> Mark setzen.

–nlsize <bytes>

-> PaketgröSSe limitieren.

Für weitere Informationen über Netlink Sockets, kannst du dir die

Netlink Sockets Tour http://www.skyfree.org/linux/kernel_network/netlink.html ansehen.

4.4 NETMAP Patch

Dieser Patch von Svenning Soerensen <svenning@post5.tele.dk> fügt ein neues Target hinzu, welches dir gestattet ein statisches 1:1 Mapping für Netzwerk Adressen vorzunehmen, wobei die Host Adressen erhalten bleiben.

Wenn du zum Beispiel möchtest das alle einkommenden Pakete von 1.2.3.0/24 nach 5.6.7.0/24 gemappt werden, kannst du das wie folgt erreichen:

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

# iptables -t nat --list
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
NETMAP     all  --  anywhere             1.2.3.0/24         5.6.7.0/24
```

Unterstützte Optionen für das NETMAP Target sind:

–to address[/mask]

-> Netwerk Address auf die gemappt werden soll.

4.5 SAME Patch

Dieser Patch von Martin Josefsson <gandalf@wlug.westbo.se> fügt ein neues Target hinzu, welches ähnlich dem SNAT ist, jedoch dem Client für jede Verbindung die gleiche Adresse zuweist.

Wenn du zum Beispiel Quelladressen auf den Bereich 1.2.3.4-1.2.3.7 abbilden willst, kannst du folgende Regel verwenden:

```
# iptables -t nat -A POSTROUTING -j SAME --to 1.2.3.4-1.2.3.7

# iptables -t nat --list
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
SAME       all  --  anywhere             anywhere           same:1.2.3.4-1.2.3.7
```

Optionen die vom SAME Target unterstützt werden sind:

–to <ipaddr>-<ipaddr>

-> Adressen auf die die Quelladressen gemappt werden. Kann auch mehrfach für mehrere Ranges spifiziert werden.

–nodst

-> Die Ziel IP nicht für die Quellauswahl verwenden.

4.6 tcp-MSS Patch

Dieser Patch von Marc Boucher <marc+nf@mbsi.ca> fügt ein neues Target hinzu, welches dir gestattet den MSS Wert von TCP SYN Paketen zu untersuchen und zu ändern um die maximale PaketgröSSe für die Verbindung festzulegen.

Wie Marc selbs sagt: DIES IST EIN HACK, der benutzt wird um shon kiminell gehirntote ISPs oder Server die ICMP Fragmentation blocken zu überwinden.

Die typische Anwendung ist wie folgt:

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu

# iptables --list
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
TCPMSS     tcp  --  anywhere             anywhere           tcp flags:SYN,RST/SYN TCPMSS clamp to PMTU
```

Optionen die vom tcp-MSS Target unterstützt werden (schlieSSen sich gegenseitig aus):

–set-mss value

explizites setzen der MSS Option auf “value”.

–clamp-mss-to-pmtu

automatisches Anpassen des MSS Wertes (Pfad_MTU - 40).

4.7 TTL Patch

Dieser Patch von Harald Welte <laforge@gnumonks.org> fügt ein Target hinzu, mit dem der Benutzer die TTL eines Paketes setzen oder erhöhen/vermindern kann.

Wenn du zum Beispiel möchtest das alle ausgehenden Verbindungen eine TTL von 126 erhalten, kannst du wie folgt vorgehen:

```
# iptables -t mangle -A OUTPUT -j TTL --ttl-set 126

# iptables -t mangle --list
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
TTL        all  --  anywhere             anywhere           TTL set to 126
```

Unterstützte Optionen für das TTL Target sind:

–ttl-set value

-> Setze TTL auf <value>.

–ttl-dec value

-> Vermindert die TTL um <value>.

–ttl-inc value

-> Erhöhe die TTL um <value>.

4.8 ulog Patch

Dieser Patch von Harald Welte <laforge@gnumonks.org> fügt ein neues Target hinzu, welches einen erweiterten Loggingmechanismus als das standard LOG Target zur Verfügung stellt. Das 'libipulog/' Verzeichnis enthält eine Library um ULOG Meldungen zu empfangen.

Harald betreibt eine *Website* http://www.gnumonks.org/projects/ulogd auf der die Dokumentation von ULOG bereit stellt. Es besteht also keinen Grund hier weiter darauf einzugehen..

4.9 ROUTE Patch

Dieser Patch von Cédric de Launois <delaunois@info.ucl.ac.be> für ein Target hinzu, das es dir gestattet Pakete direkt and ein bestimmtes Interface weiterzuleitet auch und gerade dann, wenn die IP Adresse des Paketes eine IP Adresse des Router selbst ist. Da der Kernel solche Pakete lokal ausliefern würde, können solche Pakete nicht mit dem stadard Routing zu anderen Computern weitergeleitet werden. Manchmal ist es sinnvoll genau dies zu tun ohne die Pakete zu verändern.

Optionen die vom ROUTE Target unterstützt werden sind:

–iface name

Sende das Paket direkt zu dem über den Namen dem spezifizerten Interface.

–ifindex index

Sende das Paket direkt zu dem über den Index spezifizierten Interface. Das ist sinnvoll, wenn du eine Regel entfernen möchtest die an ein Interface gebunden ist, welches nicht mehr existiert (z.B. ein Tunnel der abgebaut wurde).

Als Beispiel lass uns annehmen du möchtest einen Service auf einem Server in deinem internen Netz installieren; sagen wir ssh. Du möchtest aber auch, das der Service nach aussen so aussieht als wenn er auf denem Router laufen würde, was natürlich einschlieSSt, dass keine Pakete verändert werden. Diese Situation verbietet das Verwenden des standard Mechnismus der Portweiterleitung.

Die Lösung besteht dank dieses ROUTE Targets und einem IPIP Tunnep einfach darin, die Pakete mit Zielport 22 neu an den Rechner zu routen, der die gleiche IP Adresse hat wie dein Router und den ssh Servive betreibt.

Zeit für ein ASCII Bild:

```
              eth0  +------+ 192.168.0.1        192.168.0.2 +----+
    ----------------|router|--------------------------------|host|
    IP: 150.150.0.1 +------+                                +----+
                       | | tunl1             IP: 150.150.0.1 | |
                       | +----------------------------------+ |
                       +--------------------------------------+
                                      IPIP tunnel
```

Es ist nicht möglich solche Pakete mit dem standard Routing umzuleiten, einfach weil der Kernel davon ausgeht, dass diese Pakete lokal ausgeliefert werden müssen (sie haben die gleiche IP Address wie der Router selbst).

Das ROUTE Target stellt einen Weg zur Verfügung um das zu machen. Für das obige Beispiel würden wir folgende Regeln verwenden:

```
# iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 22 -j ROUTE --iface tunl1
# iptables -A PREROUTING -t nat -i tunl1 --j ROUTE --iface eth0

# iptables -L PREROUTING -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
ROUTE      tcp  --  anywhere             anywhere           tcp dpt:ssh ROUTE iface tunl1(7)
ROUTE      all  --  anywhere             anywhere           ROUTE iface eth0(3)
```

Die erste Regel routet Pakete aus dem Internet um und die zweite routet alle Pakete vom Host um. Die Werte in den Klammern hinter den Interfacenamen sind die Interfaceindexe. Es wird die nat Tabelle benutzt, da diese den PREROUTING Hook zur Verfügung stellt. Das heiSSt aber nicht, dass hier Pakete verändert werden.

5 Neue Connection Tracking Patches

In diesem Abschnitt stellen wir die verfügbaren Connection Tracking/NAT Patches vor. Um sie zu verwenden lade einfach das entsprechende Modul (wenn benötigt mit Optionen).

5.1 eggdrop-conntrack Patch

Dieser Patch von Magnus Sandin <magnus@sandin.cx> fügt Support für eggdrop bot Netwerke hinzu.

5.2 ftp-fxp Patch

Dieser Patch von Magnus Sandin <magnus@sandin.cx> fügt FXP Support für ftp Connection Tracking hinzu. FXP zu ge-NAT-teten ftp Servern funktioniert derzeit noch nicht. Um FXP Conntracking zu erlauben, gehe wie folgt vor:

```
# modprobe ip_conntrack_ftp.o fxp=1
```

Der Patch enthält eine Sicherheits-Warnung: WARNUNG, diesen Patch zu installieren und das Feature einzuschalten `WIRD` die Sicherheit des FTP Connection Tracking erheblich reduzieren. Benutze es sehr vorsichtig (und nur dann wenn du weiSSt was du machst).

5.3 irc-conntrack-nat Patch

Dieser Patch von Harald Welte <laforge@gnumonks.org> bewirkt das DCC mit NAT und Connection Tracking funktioniert. Per Default, trackt dieses Modul IRC Verbindungen auf Port 6667. Du kannst das aber durch die Option 'ports=xx' ändern.

5.4 pptp Patch

Diser Patch ermöglicht es pptp Verbindungen zu tracken und zu NAT-en.

5.5 record-rpc Patch

Dieser Patch von Marcelo Barbosa Lima <marcelo.lima@dcc.unicamp.br> gestattet es Netfilter portmapper Anfragen für UDP und TCP zu tacken.

5.6 snmp-nat Patch

Dieser Patch von James Morris <jmorris@intercode.com.au> gestattet es Netfilter grundlegend NAT für SNMP durchzuführen. Diese "grundlegende" Art ist SNMP-ALG, wie sie in *RFC 2962* `http://www.faqs.org/rfcs/rfc2962.html` beschrieben wird. Es funktioniert indem die IP Adresse innerhalb der SNMP Payloads an das IP-Mapping angepaSSt wird.

5.7 talk-conntrack-nat Patch

Dieser Patch von Jozsef Kadlecsik <kadlec@blackhole.kfki.hu> gestattet Netfilter talk Verbindungen zu tracken und NAT mit talk zu verwenden. Per Default werden otalk (UDP Port 517) und talk (UDP Port 518) unterstützt. Du kannst otalk/talk selektiv über die Parameter der Module ip_conntrack_talk und ip_nat_talk ein- und ausschalten. Die Optionen sind:

- otalk = 0 | 1
- talk = 0 | 1

Wobei '0' den Support für das entsprechende Protokoll ausschaltet und '1' einschaltet.

5.8 tcp-window-tracking Patch

Dieser Patch von Jozsef Kadlecsik <kadlec@blackhole.kfki.hu> gestattet Netfilter das Tracking von TCP Verbindungen entsprechend des Artikels *Real Stateful TCP Packet Filtering in IP Filter* http://www.iae.nl/users/guido/papers/tcp_filtering.ps.gz von Guido van Rooij. Es supportet window scaling und kann jetz auch bereits etablierte Verbundungen verarbeiten.

Dieser Patch setzt den "ftp-fixes" Patch voraus. Das sollte Bestandteil des standard Kernels sein. Dieser Tage...

5.9 tftp Patch

Dieser Patch von Magnus Boden <mb@ozaba.mine.nu> gestattet Netfilter das Tracking und NAT-en von tftp Verbindungen. Per Default trackt dieses Modul tftp Verbindungen auf Port 69. Dies kann mit dem Parameter 'ports=xx' auf einen anderen Port festgelegt werden.

6 Neue IPv6 Netfilter Matches

In dieser Sektion werden wir versuchen die Benutzung der neuen Netfilter Matches für IPv6 zu erklären. Die Patche werden in alphabetischer Reihenfolge besprochen. Wir werden hier nicht auf Patche eingehen, die andere Patche negativ beeinflussen; das kommt später mal.

Generell gilt für Matches, dass du die Hilfe für ein bestimmtes Modul wie folgt abfragen kannst:

```
# iptables -m match_den_du_willst --help
```

Dies wird dir die normale Hilf von ip6tables ausgeben zuzüglich der speziellen Hilfe zu "match_den_du_willst" am Ende.

6.1 agr Patch

Dieser Patch von Andras Kis-Szabo <kisza@sch.bme.hu> fügt einen neuen Match hinzu:

- "eui64": erlaubt es dir IPv6 Packete auf Basis ihrer addressierungs Parameter zu matchen.

Aktueller Status: In der Schwebe (2.4.19)

Dies Patch kann für Leute die das EUI-64 IPv6 Adressierungschema verwenden sinvoll sein, wenn sie die Pakete auf Basis der Auslieferungsadresse im LAN filtern wollen.

Als Beispiel wollen wir Pakete mit eine korrekten EUI-64 Adresse umleiten:

```
# ip6tables -N ipv6ok
# ip6tables -A INPUT -m eui64 -j ipv6ok
# ip6tables -A INPUT -s ! 3FFE:2F00:A0::/64 -j ipv6ok
# ip6tables -A INPUT -j LOG
# ip6tables -A ipv6ok -j ACCEPT

# ip6tables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ipv6ok     all      anywhere             anywhere           eui64
```

```
ipv6ok     all     !3ffe:2f00:a0::/64    anywhere
LOG        all      anywhere             anywhere           LOG level warning

Chain ipv6ok (2 references)
target     prot opt source               destination
ACCEPT     all      anywhere             anywhere
```

Dieser Match hat keine Optionen.

6.2 ahesp6 Patch

Dieser Patch von Andras Kis-Szabo <kisza@sch.bme.hu> fügt einen Match hinzu, der es dir erlaubt Pakete auf Basis des Inhalts der ah und esp Header zu filtern. Der Name des Matches:

- "ah" : gestattet das Matchen von IPv6 Packeten auf Basis ihres ah Headers.
- "esp" : gestattet das Matchen von IPv6 Packeten auf Basis ihres esp Headers.

Aktueller Status: Basic, funktioniert bei Fabrice MARIE (Author der original Version dieses Dokumentes)

Als Beispiel wollen wir alle AH Pakete verwerfen, die einen SPI Wert gleich 500 haben wobei der reservierte Bereich des Headers geprüft wird:

```
# ip6tables -A INPUT -m ah --ahspi 500 --ahres -j DROP

# ip6tables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           ah spi:500 reserved
```

Unterstützte Optionen für den ah Match sind:

–ahspi [!] spi[:spi]

-> matcht spi (Bereich).

–ahlen [!] length

-> matcht Länge des Headers.

–ahres

-> prüft reservierten Bereich des Headers.

Der esp Match arbeitet genauso wie bei IPv4:

```
# ip6tables -A INPUT -m esp --espspi 500 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           esp spi:500
```

Unterstützte Optionen für den esp Match sind:

–espspi [!] spi[:spi]

-> matcht spi (Bereich).

Bei IPv6 können diese Matche zusammen benutzt werden:

```
# ip6tables -A INPUT -m ah --ahspi 500 --ahres --ahlen ! 40 -m esp --espspi 500 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           ah spi:500 length:!40 reserved esp spi:500
```

6.3 frag6 Patch

Dieser Patch von Andras Kis-Szabo <kisza@sch.bme.hu> fügt einen Match hinzu, der es gestattet Pakete auf Basis des Inhaltes der Fragmentheader zu filtern.

Name des Matches:

- “frag” : gestattet Pakete auf Basis des Inhaltes der Fragmentheader zu filtern.

Aktueller Status: Basic, funktioniert bei Fabrice MARIE (Author der orginal Version dieses Dokumentes)

Als Beispiel wollen wir alle Pakete verwerfen, die eine ID zwischen 100 and 200 haben und das erste Fragent sind:

```
# ip6tables -A INPUT -m frag --fragid 100:200 --fragfirst -j DROP

# ip6tables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           frag ids:100:200 first
```

Unterstützte Optionen für den frag Match sind:

–fragid [!] id[:id]

-> matcht die id (Bereich) der Fragmentation.

–fraglen [!] length

-> matcht totale Länge des Headers.

–fragres

-> prüft den Inhalt des resavierten Bereiches.

–fragfirst

-> trifft für das erste Fragment zu.

–fragmore

-> trifft zu wenn keine weiteren Fragmente vorhanden sind.

–fraglast

-> trifft auf das letzte Fragment zu.

6.4 ipv6header Patch

Dieser Patch von Andras Kis-Szabo <kisza@sch.bme.hu> fügt einen Match hinzu der es dir gestattet Pakete auf Basis der erweiterten Header zu filtern.

Name des Match:

- "ipv6header" : gestattet Pakete auf Basis der Header zu filtern.

Aktueller Status: Basic, funktioniert bei Fabrice MARIE (Author der orginal Version dieses Dokumentes)

Als Beispiel wollen wir alle Pakete verwerfen die hop-by-hop, ipv6-route Header und einen Protokoll Zusatzt haben:

```
# ip6tables -A INPUT -m ipv6header --header hop-by-hop,ipv6-route,protocol -j DROP

# ip6tables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           ipv6header flags:hop-by-hop,ipv6-route,protocol
```

Und nun laSS uns Pakete verwerfen, die einen ipv6-route Erweiterungsheader haben:

```
# ip6tables -A INPUT -m ipv6header --header ipv6-route --soft -j DROP

# ip6ptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           ipv6header flags:ipv6-route soft
```

Unterstützte Optionen für den ipv6header Match sind:

[!] –header headers

-> Du kanns die für dich interessanten Header mit dieser Option spezifizieren. Akzeptierte Formate:

- hop,dst,route,frag,auth,esp,none,proto
- hop-by-hop,ipv6-opts,ipv6-route,ipv6-frag,ah,esp,ipv6-nonxt,protocol
- 0,60,43,44,51,50,59

–soft

-> Du kannst den soft-Modus spezifizieren: In diesem Modus prüft der Match nur auf die Existens der Header, nicht den vollständigen Match!

6.5 ipv6-ports Patch

Dieser Patch von Jan Rekorajski <baggins@pld.org.pl> fügt vier neue Matches hinzu:

- "limit" : gestattet dir das limitieren von parallelen TCP Verbindungen von einem bestimmten Host oder Netzwerk.
- "mac" : gestattet dir das erkennen von MAC Adressen.
- "multiport" : gestattet das Angeben einer gemischten Kombination aus Einzelports und Portbereichen für das UDP und TCP Protokoll.

- “owner” : gestattet das erkennen des Benutzer ID, des Prozesses der das Paket verschickt hat.

Aktueller Status: im Kernel

Diese Matche sind Portierung der IPv4 Versionen. Für eine genauere Beschreibung siehe dir die Dokument dort an.

6.6 length Patch

Dieser Patch von Imran Patel <ipatel@crosswinds.net> erlaubt es Pakete auf ihrer Länge zu prüfen (Dieser Patch ist eine schamlose Adaption des IPv4 Match der von James Morris <jmorris@intercode.com.au> geschrieben wurde).

Als Beispiel wollen wir alle Ping’s verbieten, die eine Paketlänge von mehr als 85 Bytes aufweisen:

```
# iptables -A INPUT -p icmp --icmp-type echo-request -m length --length 86:65535 -j DROP

# iptables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere           icmp echo-request length 86:65535
```

Unterstützte Optionen für den length Match sind:

[!] –length length[:length]

-> prüft auf entsprechende Länge(n). Die Werte bei einem Bereich sind inclusiv.

Wird bei einem Bereich ein Wert nicht angegeben, werden die implizierten Wert 0 (Minimum) und 65535 (Maximum) verwendet.

6.7 route6 Patch

Dieser Patch von Andras Kis-Szabo <kisza@sch.bme.hu> fügt einen Patch hinzu, der es gestattet Pakete auf Basis des Inhalts ihres Routingheaders zu filtern.

Name des Matches:

- “rt” : gestattet Pakete auf Basis des Inhalts ihres Routinheaders zu filtern.

Aktueller Status: Basic, funktioniert bei Fabrice MARIE (Author der orginal Version dieses Dokumentes)

Als Beispiel wollen wir alle Pakete verwerfen die den Routing Typ 0 haben und nahe am letzten Hop sind (max. 2 Hops entfernt), der Routing Pfad enthält ::1 und ::2 (aber nicht exakt):

```
# ip6tables -A INPUT -m rt --rt-type 0 --rt-segsleft :2 --rt-0-addrs ::1,::2 --rt-0-not-strict -j DROP

# ip6tables --list
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      anywhere             anywhere           rt type:0 segslefts:0:2 0-addrs ::1,::2 0-not-strict
```

Unterstützte Optionen für den rt Match sind:

–rt-type [!] type

-> trifft auf den Typ zu.

–rt-segsleft [!] num[:num]

-> prüft das Segments Left Feld (Bereich).

–rt-len [!] length

-> totale Länge des Headers.

–rt-0-res

-> prüft den Inhalt des reservierten Feldes.

–rt-0-addrs ADDR[,ADDR...]

-> Type=0 Adressen (Lise, max: 16).

–rt-0-not-strict

-> Liste von Type=0 Adressen, keine strikte Liste.

7 Neue IPv6 Netfilter Targets (Ziele)

In diesem Abschnit werden wir versuchen die Benutzung der neuen IPv6 Netfilter Targets zu erklären. Die Patche werden in alphabetischer Reihenfolge besprochen. Wir werden hier nicht auf Patche eingehen, die andere Patche negativ beeinflussen; das kommt später mal.

Generell gilt für Targets, dass du dir die Hilfe für ein bestimmtes Modul wie folgt ansehen kannst:

```
# ip6tables -j TARGET_DAS_DU_WILLST --help
```

Das wird die allgemeine Hilfe zu ip6tables anzeigen und zusätzlich die spezielle Hilfe zu "TARGET_DAS_DU_WILLST" am Ende.

7.1 LOG Patch

Dieser Patch von Jan Rekorajski <baggins@pld.org.pl> fügt ein neues Target hinzu, welches dir gestattet IPv6 Pakete ebenso zu loggen, wie du es von der IPv4 version her gewohnt bist.

Aktueller Status: Im Kernel.

Die Beispiele sind die gleichen wie die für iptables. Siehe dir also bitte diese Hilfen an.

7.2 REJECT Patch

Dieser Patch von Harald Welte <laforge@gnumonks.org> fügt ein neues Target hinzu, welches dir gestattet IPv6 Pakete ebenso abzuweisen, wie du es von der IPv4 version her gewohnt bist.

Aktueller Status: Basic, wird getestet.

Die Beispiele sind die gleichen wie die für iptables. Siehe dir also bitte diese Hilfen an.

8 Neue IPv6 Connection Tracking Patches

Das Connection Tracking wird für IPv6 derzeit noch nicht unterstützt.

9 Mitwirken

9.1 Mitwirken an neuen Erweiterung

Das Netfilter Core-Team nimmt jederzeit und gerne neue Erweierungen oder Fehlerbereinigungen an. In diesem Abschnitt werden wir uns nicht damit beschäftigen wie du eine neue Erweiterung selbst in das Patch-o-matic Paket einbinden kannst. Aber wir wollen dieses in einer zukünftigen Version dieses HOWTO's aufnehmen.

Als erstest solltest du dich mit dem *Netfilter Hacking HOWTO* http://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO.html vertraut machen..

Rusty hat bereist eine Kurzanleitung geschrieben, wie du neue Patches für Netfilter entwickeln kannst. Du findest diese in

```
/path/to/netfiltercvs/netfilter/Patch-o-matic/NEWPATCHES
```

Oder lese die aktuellste Version online unter:

NEWPATCHES http://cvs.netfilter.org/cgi-bin/cvsweb/netfilter/Patch-o-matic/NEWPATCHES .

Zu guter Letzt ist es sicher ein guter Einfall sich auf der netfilter-devel Mailingliste einzuschreiben. Mehr Informationen hierzu findest du auf der *Netfilter Homepage* http://www.netfilter.org/contact.html#devlist .

9.2 Mitwirken dieses HOWTO

Du bist herzlich dazu aufgefordert dieses HOWTO zu erweitern und zu verbessern. Wenn du das tun möchtest ist es der beste Weg das SGML Master dieses Dokumentes an die netfilter-devel Mailingliste zu schicken.

Danke für eure Hilfe! Danke an die vielen Entwickler die an jenen Teilen des Netfilter-Extension-HOWTO mitarbeiten, die zu ihren Patches gehöhren.