

Firewall mit Netfilter/iptables

Proseminar Linux SS 2002

Jürgen Lehle

Inhaltsverzeichnis

<u>EINLEITUNG</u>	3
WAS IST EINE FIREWALL?	3
WARUM SOLLTE MAN EINEN PAKETFILTER VERWENDEN?	3
WIE WERDEN PAKETE UNTER LINUX GEFILTERT?	4
<u>WICHTIGE VERÄNDERUNGEN GEGENÜBER IPCHAINS</u>	4
<u>NETFILTER / IPTABLES</u>	4
AUFBAU VON NETFILTER / WIE PAKETE DIE FILTER PASSIEREN	5
TABELLEN	5
REGELN UND KETTEN	6
OPERATIONEN AUF EINER EINZELNEN REGEL	6
BEISPIEL EINER FILTERREGEL	6
ZIELE UND AKTIONEN	7
DEFAULT POLICY	8
ADRESSEN, PORTS UND PROTOKOLLE	8
LIMITS	9
ZUSTÄNDE	9
NAT (NETWORK ADDRESS TRANSLATION)	10
TRANSPARENT PROXY	10
<u>ZUSAMMENFASSUNG</u>	10
<u>BEISPIEL EINES EINFACHEN REGELSATZES</u>	11
<u>QUELLEN</u>	13

Einleitung

Wer sich heute im Internet bewegt, sollte sich stets bewusst sein, dass er dabei nicht zu unvorsichtig sein sollte. Leicht gibt man dabei mehr von sich, seinem Rechner und seinem ganzen Netzwerk preis als man möchte. In extremen Fällen können dabei Daten auf dem Rechner des Anwenders manipuliert oder gar gelöscht werden. Ein Netzwerk ist grundsätzlich nicht sicher, gegen solche unerwünschten Zugriffe von aussen, kann aber eine Firewall abhilfe schaffen. Diese kontrolliert alle Datenpakete, die das System empfängt oder versendet. Trotz aller Sicherheitsvorkehrungen sollte der User aber auch möglichst wenig in seinen Aktivitäten eingeschränkt werden.

Was ist eine Firewall?

Allgemein versteht man unter einer Firewall einen Rechner oder eine spezielle Hardwarelösung, die als einziger Verbindungspunkt zwischen zwei Netzwerken, z.B. einem geschützten Firmen-LAN und dem Internet, dient und den Datenverkehr nach genau definierten Regeln überwacht und unerwünschte Daten abblockt.

Es existieren verschiedene Arten von Firewalls. Die klassischen Packet Filtering Firewalls schauen sich die Header von Paketen an und entscheiden darüber was mit dem Paket geschehen soll. Der Paketfilter kann ein eintreffendes Paket verwerfen, als wäre es nie angekommen, es akzeptieren oder etwas Komplizierteres. Die einfachen „stateless“ Paketfilter analysieren jedes Paket ohne Rücksicht auf dazugehörige weitere Pakete. „Stateful Packetfilters“ dagegen betrachten Pakete nicht einzeln, sondern erkennen z.B., ob ein Paket eine neue Anfrage ist, oder zu einem früheren Paket gehört.

Als dritte Gruppe existieren sogenannte Application Level Firewalls. Diese lassen keine IP-Pakete direkt passieren, sondern nehmen die eingehenden Daten anstelle des eigentlichen Clients an, überprüfen die Daten auf ihren Inhalt und leiten erst dann die Daten an den eigentlichen Client weiter. Dadurch kann z.B. jeglicher JavaScript Code aus dem http-Verkehr ausgefiltert werden.

Bei Linux ist eine Packet-Filtering-Firewall bereits im Kernel enthalten, als Modul oder direkt eingebaut. Benutzer anderer Betriebssysteme wie z.B. Windows müssen dies in Form eine „Personal Firewall“ erst nachrüsten, die allerdings auch keinen wirklichen Schutz bietet, da oft ohne jegliches Konzept einfach ein paar Programme gesperrt werden.

Warum sollte man einen Paketfilter verwenden?

Kontrolle: Wenn man einen Linuxrechner dazu benutzt, um sich aus einem internen Netzwerk z.B. mit dem Internet zu verbinden, dann kann man bestimmte Arten von Traffic erlauben und andere verbieten. Der Header von Netzwerkpaketen enthält z.B. die Zieladresse des Paketes. So kann man also verhindern, dass Daten an bestimmte Adressen im Internet gesendet werden.

Sicherheit: Wenn ein Linuxrechner als Router zwischen dem Internet und einem lokalen Netzwerk eingesetzt wird kann man einschränken was durch die Tür des Routers unser Netz betreten oder verlassen darf. Zum Beispiel kann man alles erlauben was aus dem lokalen Netz rausgeht, einen Ping auf einen lokalen Rechner aber verhindern.

Wachsamkeit: Manchmal könnte ein schlecht konfigurierter Rechner im lokalen Netz Pakete regelrecht ins Internet zu spucken. Da ist es sehr sinnvoll, wenn man dem Paketfilter sagen kann, dass er ungewöhnliche Aktivitäten an den Administrator melden soll, vielleicht kann dieser dann ja was ändern.

Wie werden Pakete unter Linux gefiltert?

Seit der Version 1.1 gehören Paketfilter zum Linuxkernel. Die erste Version, die auf „ipfw“ von BSD basiert, wurde Ende 1994 eingeführt.

Dies wurde für Linux 2.0 weiterentwickelt. Das Tool „ipfwadm“ war jetzt für die Kontrolle der Filterregeln zuständig.

Mitte 1998 wurde der Kernel für Linux 2.2 sehr stark überarbeitet und das Tool „ipchains“ eingeführt. Seit Mitte 1999 gibt es eine komplette Überarbeitung des Kernels und damit auch ein neues Tool für den Paketfilter, „iptables“ auf das sich auch dieser Vortrag konzentriert.

Um die Fähigkeiten von „iptables“ zu nutzen benötigt man einen Kernel mit der Netfilter-Infrastruktur, d.h. Kernel ab 2.3.15 können dafür verwendet werden. Dazu muss bei der Kernelkonfiguration CONFIG_NETFILTER aktiviert werden. Netfilter ist ein Kernelmodul, auf dem andere Dinge (wie die iptables-Module) aufbauen können.

Wichtige Veränderungen gegenüber ipchains

Gegenüber ipchains gab es bei iptables einige entscheidende Änderungen. Iptables ist modularer aufgebaut, was eine spätere Erweiterung erleichtern soll. Pakete, die vom Rechner weitergeleitet werden sollen, durchqueren nur noch die FORWARD Chain. Dies vereinfacht die kontrollierte Weiterleitung von IP-Paketen. Iptables beinhaltet schon eine Netwerkadressumsetzung (NAT). DoS-Angriffen können durch Limits erschwert werden. Auch die Logging-Möglichkeiten wurden erweitert.

Netfilter / iptables

Die Entwickler von IPChains, Russel, Boucher und Morris waren sich der Schwächen im Konzept von IPChains bewusst, weshalb Sie in nur einem Jahr Netfilter entwickelten. Entwickler bekommen damit ein Framework mit den Basisfunktionen eines Paketfilters an die Hand, ohne sich in die Kernelprogrammierung einarbeiten zu müssen.

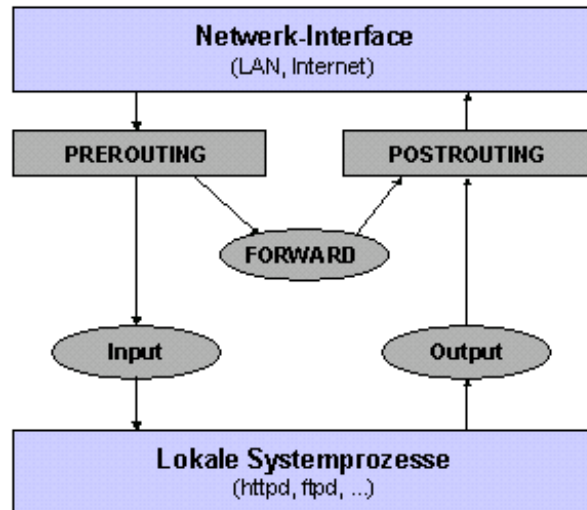
Netfilter ist protokollunabhängig und kann damit beliebig erweitert werden.

Das Tools iptables dient dazu, Regeln in die Filtertabellen des Kernels einzufügen und andere zu löschen. Iptables „spricht“ mit dem Kernel und sagt ihm welche Pakete er filtern soll.

Das heißt auch, dass Regeln, wie immer sie auch aufgesetzt werden, bei einem Neustart des Rechners verloren sind. Deshalb sollte für die Filterregeln ein Script erstellt werden, welches die Regeln bei jedem Systemstart wieder neu an den Kernel übermittelt.

Aufbau von Netfilter / Wie Pakete die Filter passieren

Der Kernel hat drei Listen von Regeln, auch Firewall-Ketten oder nur Ketten genannt. Diese heißen INPUT, OUTPUT und FORWARD.



Wenn ein Paket eines dieser drei Ketten erreicht, dann entscheidet diese, was mit dem Paket geschehen soll. Wenn in dieser Kette steht, dass das Paket zu DROPPEN ist, dann wird es an dieser Stelle verworfen. Wenn die Kette jedoch entscheidet, dass das Paket in Ordnung ist (ACCEPT), dann kann es weiter seinen Weg durch das System suchen. Jede Kette ist wie eine Checkliste mit Regeln. Diese Liste wird für jedes Paket von oben her abgearbeitet. Zuerst wie die allererste Regel befragt, ob sie auf das aktuelle Paket zutrifft, wenn nicht, dann wird das nächste Paket befragt. Wenn keine Regel auf das Paket zutrifft, dann fragt der Kernel die Policy der Kette, wie mit dem Paket zu verfahren ist. Normalerweise sagt diese Policy dem Kernel dann, dass das Paket zu verwerfen (DROP) ist.

Bei eingehenden Paketen, z.B. über die Netzwerkkarte, sieht sich der Kernel zuerst die Zieladresse des Paketes an. Ist das Paket für diesen Rechner bestimmt, dann landet es an der INPUT-Kette. Ist diese Kette passiert, erhält das Paket der Prozess, der auf dieses Paket wartet. Wenn dies nicht zutrifft, im Kernel forwarding aktiviert ist und das Paket für eine andere Netzwerkschnittstelle bestimmt ist, dann geht das Paket direkt an die FORWARD-Kette. Wenn es dort akzeptiert wird, wandert es weiter zur Schnittstelle für die es bestimmt ist. Ist Forwarding im Kerner deaktiviert, wird das Paket verworfen. Werden Netzwerkpakete von einem Programm, das auf dem Rechner läuft verschickt, gehen diese Pakete zur OUTPUT-Kette. Wenn es dort akzeptiert wird, wandert es weiter zur Schnittstelle, für die es bestimmt ist.

Tabellen

Es existieren mehrere Tabellen, um die vielfältigen Funktionen des Netfilter zu steuern. Diese „IP Tables“ werden mit iptables -t [TABELLE] angesprochen. Zur Zeit existieren drei verschiedene Tabellen, „nat“ zur Network Address Translation, „mangle“ für die Analyse und Manipulation des Netzwerkverkehrs und „filter“ für das Filtern und Protokollieren des Netzwerkverkehrs. filter ist die Standardtabelle, die nicht eigens angegeben werden muss.

Regeln und Ketten

Die verschiedenen Regeln werden bei iptables in Form von Regelketten („Chains“) organisiert. Die wichtigsten vordefinierten Regeln sind INPUT, OUTPUT und FORWARD. INPUT ist zuständig für alle Pakete, die auf dem System eingehen. Die Kette OUTPUT definiert die Regeln für Pakete welche vom eigenen System verschickt werden. Die dritte Kette, FORWARD, behandelt alle Pakete die am System eingehen, aber nicht für dieses bestimmt sind, sondern an andere Rechner weitergeleitet werden sollen.

Für jede dieser einzelnen Ketten kann ein eigener Regelsatz erstellt werden. Um die ganzen Regeln besser zu strukturieren können auch eigene, benutzerdefinierte Ketten erstellt werden.

Der Regelsatz einer Chain wird vom System von der ersten bis zur letzten Regel abgearbeitet. Trifft eine Regel mit all Ihren Bedingungen auf ein Paket zu, so wird diese Regel auf das Paket angewendet. Weitere Regeln werden danach nicht mehr beachtet. Falls keine der Regeln auf das Paket zutrifft, so bestimmt die Default-Policy der Kette was mit dem Paket passieren soll.

Die Manipulation der Filterregeln ist die Hauptaufgabe beim Einrichten einer Firewall. Jede einzelne Filterregel hat mehrere Bedingungen, die überprüft werden wenn ein Paket für diese Regel eintrifft, und das Ziel, was mit dem Paket nach erfolgreicher Überprüfung geschehen soll.

Operationen auf einer einzelnen Regel

```
iptables [Befehl][Kette][Schnittstellen][Optionen] -j [Ziel]
```

[Befehl]: -A Neue Regel an Kette anhängen
 -I neue Regel einfügen
 -R Bestehende Regel ersetzen
 -D Regel in der Kette löschen

[Kette]: INPUT | OUTPUT | FORWARD | PREROUTING | POSTROUTING | eigene

Beispiel einer Filterregel

Als einfaches Anwendungsbeispiel kann man z.B. alle ICMP-Pakete, die von einer bestimmten Adresse (127.0.0.1) kommen, verwerfen. In dem Beispiel haben wir als Bedingungen zum einen das Protokoll ICMP („Internet Control Message Protocol“) und zum anderen die Quelladresse des Paketes, 127.0.0.1. Als Ziel für dieses Paket definieren wir DROP.

```
linux:~ # ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=465 usec

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% loss, time 0ms
rtt min/avg/max/mdev = 0.465/0.465/0.465/0.000 ms
```

Hier sehen wir, dass der erste Ping ankommt.

Die Ausgabe von tcpdump verdeutlicht dies noch:

```
14:48:14.073416 localhost > localhost: icmp: echo request (DF)
14:48:14.073581 localhost > localhost: icmp: echo reply
14:48:15.073370 localhost > localhost: icmp: echo request (DF)
14:48:15.073470 localhost > localhost: icmp: echo reply
...
```

```
iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
```

Jetzt hängen wir an unsere INPUT-Kette diese Regel an, die den Kernel anweist, Pakete welche von 127.0.0.1 (-s 127.0.0.1) kommen und das Protokoll ICMP (-p icmp) verwenden, verworfen werden müssen. (-j DROP).

```
linux:~ # ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 received, 100% loss, time 0ms
```

Dann testen wir nochmal den Ping auf 127.0.0.1 und sehen, dass wir keine Bestätigung für das Paket erhalten.

Die Ausgabe von tcpdump zeigt uns auch, dass zwar die Anfrage noch ankommt, aber keine Antwort zurückgeschickt wird.

```
14:48:24.972304 localhost > localhost: icmp: echo request (DF)
14:48:25.983286 localhost > localhost: icmp: echo request (DF)
14:48:26.983297 localhost > localhost: icmp: echo request (DF)
...
```

Zum Schluß können wir die neu angelegt Regel wieder löschen. Dafür gibt es zwei Möglichkeiten:

Zum einen können wir einfach das -A durch -D ersetzen

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

Eine zweite Möglichkeit, die man wohl hauptsächlich bei einfachen Regelsätzen mit wenigen Regeln verwenden wird ist das Löschen nach der Nummer der Regel. In unserem Fall möchten wir die erste Regel der INPUT-Kette löschen.

```
iptables -D INPUT 1
```

Ziele und Aktionen

Was der Paketfilter mit einem Paket anfängt, wird durch das Ziel einer Regel festgelegt. Als Ziel kann eine der vordefinierten Ziele (z.B. ACCEPT, DROP, REJECT, LOG) oder eine benutzerdefinierte Kette sein. ACCEPT lässt ein Paket den Filter passieren, DROP unterbindet die Weiterleitung des Pakets. REJECT funktioniert ähnlich wie DROP, informiert den Absender aber mit einem Fehlerpaket. Wenn ein Paket ACCEPT, DROP oder REJECT erreicht hat, ist die Abarbeitung der Filterregeln beendet, alle folgenden Regeln werden ignoriert.

Dies erreicht man z.B. mit der Option

```
iptables FORWARD -j DROP
```

In der Regel geht einem Angriff auf einen Server ein ausgiebiges, mehr oder weniger vorsichtiges, Auskundschaften der Schwachstellen des Servers voraus. Das bietet erfahrenen Administratoren durch Logging-Mechanismen die Möglichkeit, schon vor dem eigentlichen Angriff zu erfahren auf was es die Angreifer abgesehen haben.

Neben der Quell- und Zieladresse lassen sich z.B. auch die im TCP-, bzw. IP-Header enthalten Optionen protokollieren. Damit nicht durch gezielte DoS-Attacken das Logging selbst zur Überlastung des Servers führt, kann die Anzahl der Einträge innerhalb eines Zeitintervalls frei gewählt werden. Durch Angabe des Zieles LOG können Regeln erstellt werden, um bestimmte Pakete im Syslog zu protokollieren. LOG ist das einzige Ziel, nach dessen Erreichen die Regelkette nicht verlassen wird.

Das folgende Beispiel eine Regelkette zum Loggen aller ICMP-Pakete:

```
iptables -N logchain
iptables -A logchain -p icmp -j LOG
iptables -A INPUT -j logchain
```

Default Policy

Es gibt zwei grundlegende Arten, wie ein Paketfilter arbeitet:

1. Alles erlauben und unsichere Pakete verwerfen
2. Alles verbieten und benötigte Pakete erlauben

Die Policy legt fest, was der Filter mit Paketen anfängt, für die keine Regel zutrifft. Voreingestellt ist für INPUT und OUTPUT das Ziel ACCEPT, für FORWARD ist die Default-Policy DROP.

Gesetzt wird die Default-Policy z.B. mit

```
iptables -P FORWARD DROP
```

Adressen, Ports und Protokolle

Nur Aktion und Ziel sind zwingend erforderlich, alle anderen Argumente einer Regel sind optional und lassen sich beliebig miteinander kombinieren.

Am häufigsten wird man wohl den Absender und den Empfänger eines Datenpakets angeben. Quell- („-s“, „--source“ oder „-src“) und Ziel- („-d“, „--Destination“ oder „-dst“) IP-Adresse können auf vier verschiedenen Arten bestimmt werden. Meistens wird dabei der vollständige Name (z.B. „localhost“ oder „www.unim.de“) verwendet. Man kann aber natürlich auch die IP-Adresse (z.B. „127.0.0.1“) angeben.

Die beiden anderen Möglichkeiten werden verwendet um Gruppen von IP-Adressen („134.60.246.0/24“ oder „134.60.246.0/255.255.255.0“) zu bestimmen. Beide bestimmen die IP-Adressen von 134.60.246.0 bis 134.60.246.255. Die Zahl hinter dem / gibt an welcher Teil der IP-Adresse von Bedeutung ist. Wenn man gar keine IP-Adresse angeben will, kann man /0 verwenden. Das wird allerdings nur sehr selten benötigt.

Ein Beispiel zum Filtern nach der Quelladresse:

```
iptables -A FORWARD -s 192.168.17.10 -j REJECT
```

Damit werden alle Pakete verworfen, die von der IP-Adresse 192.168.17.10 verworfen und der Absender benachrichtigt, dass sein Paket nicht angekommen ist.

Bei vielen Optionen kann auch die Inversion verwendet werden. Beispielsweise bedeutet „-s ! localhost“ alle Pakete, die nicht von localhost stammen.

Die verschiedenen IP-Protokolle lassen sich über -p tcp, -p udp, oder -p icmp ansprechen, oder man kann die Protokolle auch über ihren numerischen Wert (z.B. 6 für TCP) angeben.

TCP- bzw. UDP-Ports werden mit den Optionen `--sport` („source port“) und `--dport` („destination port“) angegeben. So lassen sich z.B. alle Pakete abblocken, die den Zielport 23 haben:

```
iptables -A INPUT --dport 23 -j DROP
```

Beim Protokoll ICMP kennt iptables die Option `--icmp-type` über die sich verschiedene ICMP-Typen wie „echo-request“ oder „echo-reply“ ansprechen lassen.

Mit der folgenden Regel werden alle ICMP-Pakete, welche vom Rechner mit der IP-Adresse 192.168.0.1 kommen, verworfen:

```
iptables -A INPUT -s 192.168.0.1 -p icmp -j DROP
```

Die verschiedenen Netzwerkschnittstellen lassen sich mit den Optionen `„-i“` (oder `„--in-interface“`) und `„-o“` (oder `„--out-interface“`) angeben. Eine Schnittstelle kann z.B. eine Netzwerkkarte sein, an der Pakete eingehen (`„-i“`) oder ausgehen (`„-o“`). Pakete, welche die INPUT-Kette durchwandern, haben keine Output-Schnittstelle, daher wird keine Regel, für die eine `„-o“` angegeben ist zutreffen. Genauso haben OUTPUT-Pakete keine Input-Schnittstelle. Pakete, die die FORWARD-Kette durchlaufen, können sowohl eine Input- als auch Output-Schnittstelle haben.

Man kann bei der Definition einer Filterregel auch Schnittstellen angeben, die zum Zeitpunkt der Initialisierung des Filter-Scriptes gar nicht existieren. Dies ist z.B. nützlich um Pakete über Wählverbindungen, etwa über die Schnittstelle `„ppp0“` zu filtern.

Limits

Eine weitere interessante Filteroption ist `„limit“`. Damit kann man herausfinden, ob und wie viele Pakete mit gleichen Kriterien innerhalb einer bestimmten Zeit eintreffen. Der Einsatz von `„limit“` ist recht einfach. Die Option `--limit [treffer/zeit]` gibt an, wie oft eine Regel angewendet werden darf, z.B. `10/minute` oder `20/hour`. Sobald dieses Limit überschritten wird, so wird die Regel außer Kraft gesetzt, bis die Anzahl der Treffer das Limit wieder unterschreitet. Mit Limit kann man beispielsweise die Anzahl der Log-Einträge für bestimmte Ereignisse begrenzen.

Im folgenden Beispiel werden bis zu 30 Pings/Minute protokolliert:

```
iptables -A INPUT -p icmp --icmp-type ping -m
        limit --limit 30/minute -j LOG
```

Eine weitere Option ist `„limit-burst [treffer]“`. Damit gibt man an, wie viele Treffer maximal in der internen Warteliste bearbeitet werden sollen. Es ist damit möglich, die Bandbreite für bestimmte Services oder Protokolle zu begrenzen.

Zustände

Mit dem Modul `„state“` ist ein sogenanntes `„Stateful Filtering“` möglich. Damit werden nicht einzelne Pakete untersucht, sondern die Pakete werden mit ihren zugehörigen Paketen betrachtet. So können beispielsweise alle neuen eingehenden Pakete geblockt werden, eingehenden Paketen die als Antwort auf ein ausgehendes Paket ankommen, kann dagegen der Eintritt gewährt werden. Die Benutzung dieser recht komplexen Funktion gestaltet sich relativ unkompliziert. Es gibt nur eine einzige Option `--state [state]`. Hier sind verschiedene Angaben möglich, die auch kombiniert angegeben werden können: `--state INVALID` erfasst Pakete, die weder zu einer bestehenden, noch zu einer neuen Verbindung gehören, `--state NEW` trifft auf alle neuen Pakete zu, `--state ESTABLISHED` bezeichnet Pakete, die zu einer bestehenden Verbindung gehören und `--state RELATED` umfasst alles, was zu einer Verbindung gehört, aber ein echter Teil dieser Verbindung ist. Dazu gehören z.B. ICMP Fehlermeldung einer nicht zustande gekommenen Verbindung.

NAT (Network Address Translation)

Mit „Network Address Translation“ kann die Quell- bzw. Zieladresse eines Paketes manipuliert werden. Damit können beispielsweise Anfragen an eine bestimmte Adresse an einen anderen Rechner weitergeleitet werden:

```
iptables -A INPUT --dport 21 -j DNAT --to 192.168.0.5
```

Um mehrere Rechner über nur einen Internetzugang mit einer einzigen IP-Adresse ans Internet anzubinden wird auch eine Form des NAT, nämlich Masquerading benötigt. Dazu wird die Quelladresse eines Datenpaketes von der ursprünglichen Adresse im lokalen Netzwerk auf die im Internet eindeutige IP-Adresse des Rechners umgestellt, der die Pakete ans Internet weiterleitet.

Dieses Masquerading kann mit iptables recht einfach eingerichtet werden. Dazu ist lediglich folgende Regel einzutragen:

```
iptables -t nat -A POSTROUTING -o ppp0 -s 192.168.17.0/24  
-j MASQUERADE
```

Damit werden alle Pakete geändert, die aus dem lokalen Netz (192.168.17.xxx) stammen. Beim Masquerading muss die Quelladresse nicht explizit angegeben werden. Es wird automatisch die Adresse der Schnittstelle genommen, über die das Paket ausgeht.

Dies ist eine einfache Möglichkeit das eigene Netzwerk weiter abzusichern, da so keine direkten Verbindungen von außen auf die lokalen Rechner möglich sind.

Transparent Proxy

Eine weitere Einsatzmöglichkeit von NAT sind sogenannte „transparente Proxies“. Damit können bestimmte Arten von Paketen an einen Proxy weitergeleitet werden, z.B. lassen sich http-Anfragen, ohne Einstellungen am Browser vornehmen zu müssen, an den Linux-Proxy Squid weiterleiten.

Dieser kann auf Applikationsebene die http-Pakete genauer untersuchen und gegebenenfalls den Inhalt der Pakete noch verändern.

Mit folgendem einfachen Kommando werden alle TCP-Pakete, welche an eth0 ankommen und Port 80 als Ziel haben, an den Port des Proxy-Servers umgeleitet.

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

Zusammenfassung

Zusammengefasst lässt sich feststellen, dass das was Linux mit Netfilter/iptables anbietet mehr ist als nur eine einfach zu bedienende Firewall. Paketfilter, NAT und Connection Tracking ergänzen sich optimal und liefern dadurch eine flexible und kostengünstige Lösung um ein Netzwerk vor ungewolltem Datenverkehr zu schützen.

Beispiel eines einfachen Regelsatzes

```
#!/bin/bash
#
# /etc/ppp/ip-up.local
#

# Benötigte Variablen füllen
EXTIF=ppp0
INTIF=eth0
IPTABLES=/usr/sbin/iptables

# Benötigte Module laden
modprobe ip_tables &> /dev/null
modprobe ip_conntrack &> /dev/null
modprobe ip_conntrack_ftp &> /dev/null
modprobe ipt_state &> /dev/null
modprobe iptable_nat &> /dev/null
modprobe ipt_REJECT &> /dev/null
modprobe ipt_MASQUERADE &> /dev/null

# Alle alten Regeln löschen, anschließend die Default-Policy setzen
$IPTABLES -F
$IPTABLES -X
$IPTABLES -F -t filter
$IPTABLES -F -t nat
$IPTABLES -F -t mangle
$IPTABLES -t filter -X
$IPTABLES -t nat -X
$IPTABLES -t mangle -X

# Wenn keine andere Regel greift, alles sperren
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT DROP

# In der NAT-Tabelle (-t nat) eine Regel fuer alle ueber das Internet-
# Device (-o) ausgehenden Pakete, die maskiert werden sollen, hinter dem
# Routing (POSTROUTING) anhaengen (-A).
$IPTABLES -t nat -A POSTROUTING -o $EXTIF -j MASQUERADE

# Forwarding für VNC aktivieren
iptables -t nat -A PREROUTING -p tcp --dport 5800 -i $EXTIF \
-j DNAT --to 192.168.17.50:5800
iptables -t nat -A PREROUTING -p tcp --dport 5900 -i $EXTIF \
-j DNAT --to 192.168.17.50:5900

# Rechner10 für's Internet sperren
$IPTABLES -A FORWARD -d 192.168.17.10 -s ! 192.168.17.0/24 -j REJECT
$IPTABLES -A FORWARD -s 192.168.17.10 -d ! 192.168.17.0/24 -j REJECT

# Alles von intern erlauben
$IPTABLES -t nat -A PREROUTING -i $INTIF -s 192.168.17.0/24 -j ACCEPT
```

```
# Alle ICMP-Pakete von extern erlauben
$IPTABLES -A INPUT -i $EXTIF -p icmp -j ACCEPT

# Pakete, die zu bestehenden Verbindungen gehören, erlauben,
# neue Pakete von extern blocken
$IPTABLES -t nat -A PREROUTING -m state --state ESTABLISHED,RELATED \
-j ACCEPT
$IPTABLES -t nat -A PREROUTING -i $EXTIF -m state --state NEW -j DROP

# Alles Loggen, was bis jetzt durchgegangen ist.
# Sollte man sich genauer anschauen, wenn das im Log auftaucht
$IPTABLES -A INPUT -m limit -j LOG --log-prefix "FINAL IN "
$IPTABLES -A OUTPUT -m limit -j LOG --log-prefix "FINAL OUT "
$IPTABLES -A FORWARD -m limit -j LOG --log-prefix "FINAL FOR "

# Und dann alles abblocken, was bis hier durchgegangen ist
$IPTABLES -A INPUT -j REJECT
$IPTABLES -A OUTPUT -j REJECT
$IPTABLES -A FORWARD -j REJECT
```

Quellen

c't Ausgabe 26/2001, Ausgabe 04/2002

LinuxEnterprise 6/2001

<http://archiv.tu-chemnitz.de/pub/2000/0051/data/netfilter.html>

<http://netfilter.samba.org/unreliable-guides/packet-filtering-HOWTO/>

<http://www.iks-jena.de/mitarb/lutz/usenet/Firewall.html>

<http://www.adsl4linux.de/howtos/lan/chapter5.php>

<http://www.tldp.org/HOWTO/mini/TransparentProxy.html>