Paketfilterung mit Linux

by Vincent Renardias
<vincent(at)renardias.com>

About the author:

GNU/Linux−Benutzer seit 1993, ist Vincent Renardias seit 1996 in dessen Entwicklung involviert: Debian−Entwickler, französischer Übersetzer von GIMP und GNOME, Gründer der Linux−Anwendergruppe in Marseille (PLUG) ... Nun Manager für Forschung und Entwicklung bei der Firma EFB2, trägt er weiterhin zu GNU/Linux bei.

Translated to English by:
Georges Tarbouriech
<gt(at)linuxfocus.org>

Abstract:

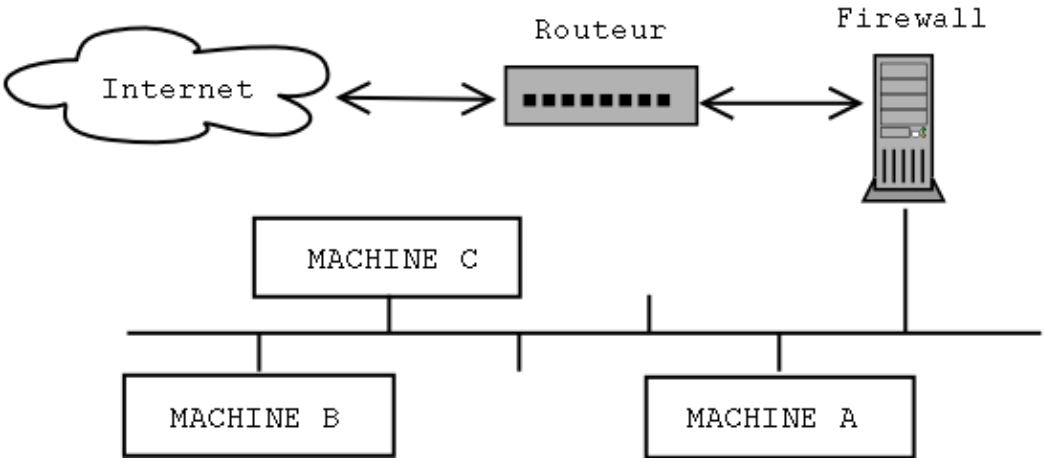
Dieser Artikel wurde zuerst in einer Spezial−Ausgabe des französischen Linux−Magazins veröffentlich, die sich auf Sicherheit konzentrierte. Der Editor, die Autoren und die Übersetzer haben LinuxFocus freundlicherweise erlaubt, jeden Artikel aus dieser Spezialausgabe zu veröffentlichen. Entsprechend wird LinuxFocus diese Artikel veröffentlichen, sobald diese ins Englische übersetzt sind. Dank an alle Personen, die an dieser Arbeit beteiligt sind. Dieser Abstrakt wird für jeden Artikel reproduziert, der den gleichen Ursprung hat.

Eine der guten Möglichkeiten, Einbruchsversuche zu verhindern, ist es zu filtern, was in einem Netzwerk nutzlos ist. Diese Aufgabe wird normalerweise einer Maschine übertragen, die als Firewall genutzt wird. In diesem Artikel präsentieren wir die erforderlichen Grundlagen für die Implementierung und Konfiguration eines solchen Systems.

Gateway, Proxy−Arp oder Ethernet−Brücke ?

Der Filtermechanismus kann als ein Netz betrachtet werden, das einige unerwünschte Pakete stoppt. Das Wichtigste ist es, die richtige Größe der Maschen zu bestimmen und den rechten Ort für die Installation.

Firewall−Anordnung im Netzwerk



Um Pakete ordnungsgemäß filtern zu können, muss der Filtermechanismus physikalisch zwischen dem zu schützenden Netzwerk und dem "Rest der Welt" positioniert sein. Dies wird durch eine Maschine mit zwei Netzwerk−Karten (normalerweise Ethernet) erreicht, eine mit dem internen Netzwerk verbunden und die andere mit dem Router, der den Zugang nach aussen ermöglicht. Auf diese Art muss alle Kommunikation durch die Firewall gehen, die diese Kommunikation in Abhängigkeit von ihrem Inhalt blockt oder passieren lässt.
Der Rechner mit dem Filtermechanismus kann auf drei verschiedene Arten konfiguriert werden:

− "einfaches" Gateway: dies ist die häufigste Konfiguration. Der Rechner wird als Gateway zwischen zwei Netzwerken oder Subnetzwerken benutzt. Die Computer im lokalen Netzwerk werden so konfiguriert, dass sie die Firewall anstelle des Routers für ihre Default−Route benutzen.

− "Proxy−ARP"−Gateway: die vorstehende Konfiguration impliziert die Aufteilung des Netzwerkes in zwei Subnetz−Bereiche, wobei die Hälfte der verfügbaren IP−Adressen verloren geht. Dies ist etwas ärgerlich. Beim Beispiel eines Subnetzwerkes mit 16 Adressen (und einer 28−Bit−Netzmaske) gehen wir von 14 auf nur 6 verfügbare Adressen, da die Netzwerk− und Broadcast−Adresse benutzt werden. Durch Hinzufügen eines Bits zur Subnetzmaske verringern wir von 14 auf 6 verfügbare Adressen (8 IPs abzüglich der für Netzwerk und Broadcast). Wenn Sie sich es nicht leisten können, die Hälfte der verfügbaren IP−Adressen zu verlieren, können Sie die Technik nutzen, die etwas später im Artikel erläutert wird. Weiterhin erfordert diese Technik keine Änderung in der Netzwerk−Konfiguration der vorhandenen Maschinen, weder am Router noch an den geschützten Computern.

− Ethernet−Brücke: Installation eines Ethernet−Gateways (nicht eines IP−Gateways) macht den Filtermechanismus für andere Rechner unsichtbar. Eine solche Konfiguration kann vorgenommen werden, ohne den Ethernet−Schnittstellen IP−Adressen zuzuweisen. Die Maschine ist dann mittels ping, traceroute usw. nicht zu entdecken. Wir müssen beachten, dass eine Paketfilter−Implementation in einer solchen Konfiguration einen 2.2.x−Kernel erfordert, da die Portierung dieser Eigenschaft auf die 2.4.x−Kernel noch nicht fertig ist.

Grundregeln des Filterns

Da wir nun wissen, wo wir unseren Filter installieren, müssen wir definieren, was er blockieren und was akzeptieren soll.
Es gibt zwei Möglichkeiten, einen Filter zu konfigurieren:

– Die gute: Es sind nur Pakete zulässig, die von einer Regel akzeptiert werden.
– Die schlechte: (leider oft benutzt) nur ausdrücklich verbotene Pakete werden blockiert, alle anderen werden akzeptiert.

Dies ist einfach zu erklären: im ersten Fall führt das Vergessen einer Regel dazu, dass ein Dienst nicht ordentlich oder gar nicht funktioniert. Dies fällt normalerweise recht schnell auf und dann reicht es, die passende Regel zu aktivieren, damit wieder alles funktioniert.
Im zweiten Fall bedeutet das Vergessen einer Regel eine mögliche Verletzbarkeit, die vielleicht schwer zu finden ist ... wenn wir sie überhaupt finden.

Netfilter

Die am häufigsten verwendete Paketfilter–Software in Linux 2.4 ist Netfilter; es ersetzt das in Linux–Kernel 2.2 genutzte 'ipchains' recht gut. Netfilter besteht aus zwei Teilen: der Kernel–Unterstützung, die in Ihrem Kernel einkompiliert sein muss und dem `'iptables'`–Befehl, der in Ihrem System verfügbar sein sollte.

Beispiel für eine Einrichtung

Ein kommentiertes Beispiel ist besser als eine lange Rede, daher beschreiben wir als nächstes, wie man einen Filtermechanismus installiert und einrichtet. Zunächst wird die Maschine als Gateway unter Benutzung von Proxy–ARP konfiguriert, um die Anzahl der IP–Adressen zu begrenzen und wir werden das Filtersystem einrichten.

Der Autor hat eine Vorliebe für die Debian–Distribution, um solch ein System einzurichten, aber jede andere Distribution ist genauso gut.

Testen Sie zuerst, ob Ihr Kernel über Netfilter–Unterstützung verfügt. Wenn dies der Fall ist, finden Sie in den Boot–Nachrichten folgendes:

```
ip_conntrack (4095 buckets, 32760 max)
ip_tables: (c)2000 Netfilter core team
```

Andernfalls müssen Sie den Kernel neukompilieren, nachdem Sie die Netzfilter–Unterstützung aktiviert haben. Die entsprechenden Optionen finden sich im Untermenü `"Netzwerk-Paketfilterung"` des Menüs `"Netzwerk-Optionen"`. Aus dem Abschnitt `"Netfilter-Konfiguration"` wählen Sie die von Ihnen benötigten Optionen. Im Zweifel können Sie alle aktivieren. Desweiteren ist es besser, Netfilter in den Kernel zu legen und keine Module zu benutzen. Wenn aus irgendeinem Grund eines der Netfilter–Module fehlt oder nicht geladen wird, funktioniert die Filterung nicht und wir reden besser nicht über die Risiken, die das impliziert.

Sie sollten auch das Paket 'iproute2' installieren (dies ist nicht obligatorisch, aber unser Beispiel benutzt es, da es uns ermöglicht, das Konfigurations–Skript einfacher zu gestalten). Mit Debian reicht der Befehl 'apt–get install iproute'.
Bei anderen Distributionen müssen Sie das entsprechende Paket installieren. Der übliche Weg ist es, die Software aus den Quellen zu installieren, die Sie von der folgenden Adresse abrufen können:
`ftp://ftp.inr.ac.ru/ip-routing/`

Nun müssen die 2 Ethernet−Karten konfiguriert werden. Wir müssen beachten, dass der Linux−Kernel bei der Auto−Erkennung der Hardware mit der Suche nach Netzwerk−Karten aufhört, sobald eine gefunden wurde. Dementsprechend wird nur die erste erkannt.
Eine einfache Lösung für dieses Problem besteht darin, der `lilo.conf`−Datei folgende Zeile hinzuzufügen:
`append="ether=0,0,eth1"`

Nun müssen wir die Ethernet−Schnittstellen konfigurieren. Die Methode, die wir benutzen, erlaubt es uns, die gleiche IP−Adresse für beide Karten zu verwenden und damit eine Adresse zu sparen:
Wir nehmen an, dass wir ein `10.1.2.96/28`−Subnetzwerk haben, d.h. Adressen von 10.1.2.96 bis 10.1.2.111 einschließlich. Der Router erhält die Adresse 10.1.2.97 und unser Filter−Rechner die 10.1.2.98.
Die `eth0`−Schnittstelle wird mit dem Router über ein RJ45−Cross−Kabel verbunden, wenn beide ohne einen Hub/Switch direkt verbunden sind; die `eth1`−Schnittstelle wird mit dem Hub/Switch verbunden und von dort mit den Rechnern im lokalen Netzwerk.

Entsprechend werden beide Schinttstellen mit den folgenden Parametern konfiguriert:

```
Adresse  : 10.1.2.98
Netzmaske  : 255.255.255.240
Netzwerk  : 10.1.2.96
Broadcast: 10.1.2.111
Gateway  : 10.1.2.97
```

Als nächstes benutzen wir das folgende Skript, das nach der Initial−Konfiguration der Netzwerk−Karten gestartet werden muss, um die Einrichtung zu beenden.

```
net.vars: Konfigurations-Variablen

PREFIX=10.1.2
DMZ_ADDR=$PREFIX.96/28
# Schnittstellen-Definitionen
BAD_IFACE=eth0
DMZ_IFACE=eth1
ROUTER=$PREFIX.97


net-config.sh: Netzwerk-Konfigurationsskript

#!/bin/sh
# Aktivieren Sie die nächste Zeile, um die Befehle während der Ausführung anzuzeigen
# set -x
# Wir lesen die definierten Variablen aus der vorherigen Datei
source /etc/init.d/net.vars
# Wir entfernen die aktuellen Routen aus dem lokalen Netzwerk
ip route del $PREFIX.96/28 dev $BAD_IFACE
ip route del $PREFIX.96/28 dev $DMZ_IFACE
# Wir definieren, dass das lokale Netzwerk über eth1 erreicht wird
# und der Router über eth0.
ip route add $ROUTER dev $BAD_IFACE
ip route add $PREFIX.96/28 dev $DMZ_IFACE
# Wir aktivieren Proxy-ARP für beide Schnittstellen
echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
# Wir aktivieren IP-Forwarding, damit die Pakete von einer Karte zur anderen
# geleitet werden können.
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Wir kehren nun zum Proxy−ARP−Mechanismus zurück, den wir für unsere Konfiguration benötigen.
Wenn ein Rechner mit einem anderen im gleichen Netzwerk Verbindung aufnehmen möchte, benötigt er

dessen Ethernet−Adresse (oder MAC− oder Hardware−Adresse), die der IP−Adresse des Zielrechners entspricht. Der Quellrechner sendet dann die Frage "Wie lautet die MAC−Adresse der Schnittstelle, die die IP−Adresse 1.2.3.4 besitzt?", und der Zielrechner muss antworten.

Hier ist ein Beispiel eines solchen "Gesprächs", mittels tcpdump aufgezeichnet:
− die Anforderung: der Rechner 172.16.6.72 fragt nach der MAC−Adresse zur IP−Adresse 172.16.6.10.
```
19:46:15.702516 arp who-has 172.16.6.10 tell 172.16.6.72
```
− die Antwort: der Rechner 172.16.6.10 teilt seine Karten−Nummer mit:
```
19:46:15.702747 arp reply 172.16.6.10 is-at 0:a0:4b:7:43:71
```

Dies bringt uns zum Ende dieser kurzen Erklärung: Die ARP−Anforderungen werden mittels Broadcast (Rundspruch) vorgenommen und sind daher auf ein physikalisches Netzwerk begrenzt. Entsprechend sollte die Anforderung von einer geschützten Maschine, die MAC−Adresse des Routers herauszufinden, von dem Filter−Rechner geblockt werden. Die Aktivierung der Proxy−ARP−Eigenschaft erlaubt uns die Lösung dieses Problems, da die ARP Anfragen und Antworten, die zu der Karte kommen, dann weitergeleitet werden.

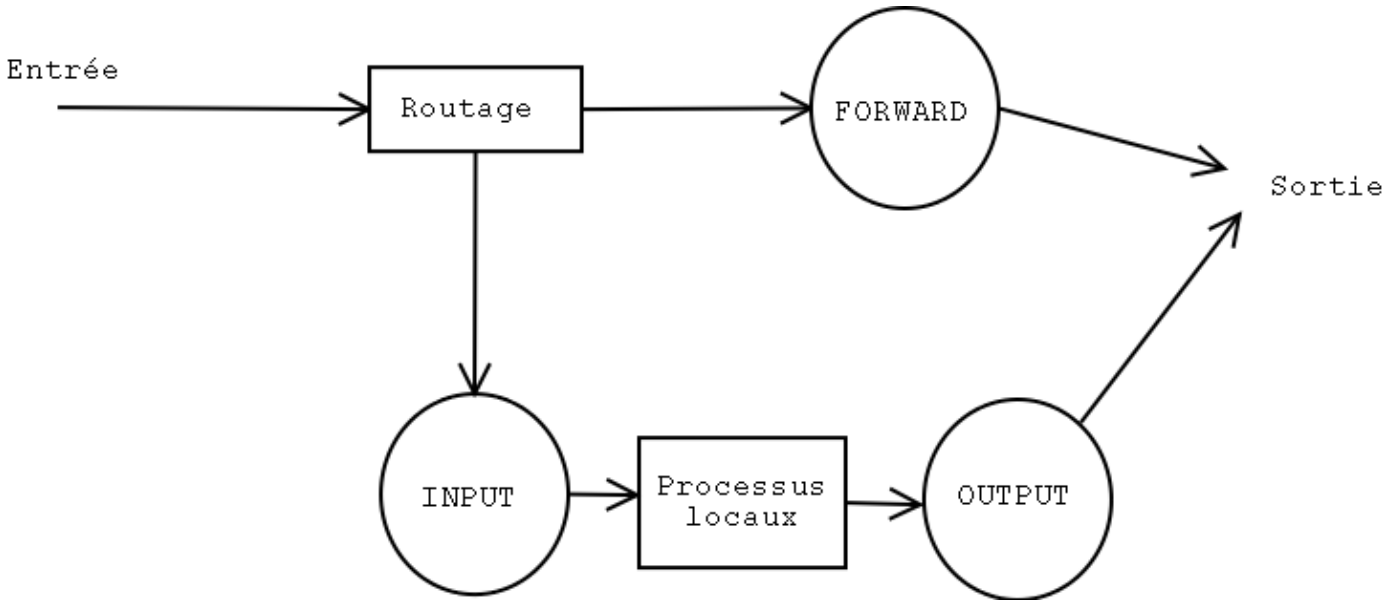
Zu diesem Zeitpunkt sollten Sie über ein funktionierendes Netzwerk verfügen mit einem Rechner, der den gesamten Verkehr zwischen dem lokalen Netzwerk und der Außenwelt verwaltet.

Nun müssen wir das Filtersystem mittels Netfilter einrichten.
Netfilter erlaubt es, direkt auf den Paketfluss zu reagieren. In der Basiskonfiguration werden die Pakete über 3 Regel−Ketten verwaltet:
− INPUT: für die über eine Schnittstelle hereinkommenden Pakete,
− FORWARD: für alle Pakete, die von einer zur anderen Schnittstelle weitergeleitet werden,
− OUTPUT: für die über eine Schnittstelle herausgehenden Pakete.

Der 'iptables'−Befehl erlaubt es, Regeln in jeder dieser Ketten hinzuzufügen, zu ändern oder zu entfernen, um das Filter−Verhalten zu modifizieren.
Weiterhin hat jede Kette eine Standard−Regelung, d.h. sie weiss, was zu tun ist, wenn keine Regel in der Kette auf ein Paket zutrifft.

Die vier häufigsten Optionen sind:
− ACCEPT: das Paket kann passieren,
− REJECT: das Paket wird zurückgewiesen und das entsprechende Fehlerpaket wird gesendet (ICMP Port Unreachable, TCP RESET, je nach Fall),
− LOG: schreibt einen Paket−Hinweis in syslog,
− DROP: das Paket wird ignoriert und keine Antwort gesendet.

Paketfluß durch die Standard−Regelketten



Hier sind die Hauptoptionen von iptables, die die Manipulation ganzer Ketten ermöglichen. Wir werden sie später noch erläutern:

−N: erstellt eine neue Kette.
−X: entfernt eine leere Kette.
−P: ändert die Standardregel einer Kette.
−L: listet die Regeln einer Kette auf.
−F: entfernt alle Regeln in einer Kette.
−Z: löscht die Byte− und Paketzähler, die die Kette durchlaufen haben.

Zum Ändern einer Kette gibt es die folgenden Befehle:
−A: fügt eine Regel am Ende einer Kette an.
−I: fügt eine neue Regel in einer bestimmten Position in einer Kette ein.
−R: ersetzt eine bestimmte Regel in einer Kette.
−D: löscht eine Regel in einer Kette, entweder über deren Nummer oder die Beschreibung der Regel.

Ein kleines Beispiel: wir werden die PING−Antworten (das ist das ICMP−Paket 'echo−reply') blockieren, die von einer bestimmten Maschine kommen.
Zuerst testen wir, ob wir den Rechner mittels "ping" erreichen können:

```
# ping -c 1 172.16.6.74
PING 172.16.6.74 (172.16.6.74): 56 data bytes
64 bytes from 172.16.6.74: icmp_seq=0 ttl=255 time=0.6 ms

--- 172.16.6.74 ping statistics ---

1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
```

Nun fügen wir der INPUT−Kette eine Regel hinzu, die die ICMP−Reply−Pakete ('−p icmp −−icmp−type echo−reply') abfängt, die vom Rechner 172.16.6.74 ('−s 172.16.6.74') kommen. Diese Pakete werden ignoriert ('−j DROP').

```
# iptables -A INPUT -s 172.16.6.74 -p icmp --icmp-type echo-reply -j DROP
```

Nun PINGeln wir diesen Rechner erneut an:

```
# ping -c 3 172.16.6.74
PING 172.16.6.74 (172.16.6.74): 56 data bytes

--- 172.16.6.74 ping statistics ---

3 packets transmitted, 0 packets received, 100% packet loss
```

Wie wir erwarten konnten, wurden diese Antworten nicht durchgelassen. Wir können überprüfen, dass die 3 Antworten geblockt wurden (3 Pakete mit insgesamt 252 Byte):

```
# iptables -L INPUT -v
Chain INPUT (policy ACCEPT 604K packets, 482M bytes)
 pkts bytes target     prot opt in     out     source       destination
   3   252   DROP       icmp --  any    any     172.16.6.74    anywhere
```

Um zur Ausgangssituation zurückzukehren, müssen wir nur die erste Regel aus der INPUT−Kette entfernen:

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

Nun sollte PING wieder funktionieren:

```
# ping -c 1 172.16.6.74
PING 172.16.6.74 (172.16.6.74): 56 data bytes
64 bytes from 172.16.6.74: icmp_seq=0 ttl=255 time=0.6 ms

--- 172.16.6.74 ping statistics ---

1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.6/0.6/0.6 ms
#
```

Es klappt!

Sie können weitere Regelketten zu den 3 vordefinierten hinzufügen (die Sie auf keinen Fall entfernen können) und Datenverkehr erzeugen, der diese Ketten durchläuft. Dies kann z.B. nützlich sein, um doppelte Regeln in verschiedenen Ketten zu vermeiden.

Nun richten wir die erforderlichen Regeln für eine minimale Firewall ein. Diese wird die Dienste ssh, domain (DNS), http und smtp erlauben und sonst nichts.
Zur Vereinfachung schreiben wir die Einrichtungs−Befehle in ein Shell−Skript, um die Konfiguration zu erleichtern. Das Skript beginnt damit, die aktuelle Konfiguration zu löschen, bevor die neue eingerichtet wird. Dieser kleine Trick erlaubt es, das Skript zu starten, wenn die Konfiguration aktiv ist, ohne doppelte Regeln zu riskieren.

rc.firewall

```
#!/bin/sh

# Verwerfen der bisherigen Regeln
iptables -F
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD


# Die Regelkette wird entsprechend der Richtung erstellt.
# bad = eth0 (outside)
# dmz = eth1 (inside)
```

```
iptables -X bad-dmz
iptables -N bad-dmz
iptables -X dmz-bad
iptables -N dmz-bad
iptables -X icmp-acc
iptables -N icmp-acc
iptables -X log-and-drop
iptables -N log-and-drop

# Spezielle Regelkette zum Aufzeichnen von Paketen, bevor sie geblockt
# werden
iptables -A log-and-drop -j LOG --log-prefix "drop "
iptables -A log-and-drop -j DROP

# Pakete mit aktivierten TCP-Flags werden verworfen, ebenso diejenigen
# ohne irgendwelche Flags (oft mit Nmap-Scans genutzt)
iptables -A FORWARD -p tcp --tcp-flags ALL ALL -j log-and-drop
iptables -A FORWARD -p tcp --tcp-flags ALL NONE -j log-and-drop

# Pakete von reservierten Adressen werden verworfen, ebenso multicast
iptables -A FORWARD -i eth+ -s 224.0.0.0/4 -j log-and-drop
iptables -A FORWARD -i eth+ -s 192.168.0.0/16 -j log-and-drop
iptables -A FORWARD -i eth+ -s 172.16.0.0/12 -j log-and-drop
iptables -A FORWARD -i eth+ -s 10.0.0.0/8 -j log-and-drop

# Zu einer bestehenden Verbindung gehörende Pakete werden akzeptiert
iptables -A FORWARD -m state --state INVALID -j log-and-drop
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
#  Die entsprechende Kette wird entsprechend der Paketquelle weitergeleitet
iptables -A FORWARD -s $DMZ_ADDR -i $DMZ_IFACE -o $BAD_IFACE -j dmz-bad
iptables -A FORWARD -o $DMZ_IFACE -j bad-dmz
# Der gesamte Rest wird ignoriert
iptables -A FORWARD -j log-and-drop

# Akzeptierte ICMPs
iptables -A icmp-acc -p icmp --icmp-type destination-unreachable -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type source-quench -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type time-exceeded -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type echo-request -j ACCEPT
iptables -A icmp-acc -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A icmp-acc -j log-and-drop

# Aussen -> Innen-Regelkette
# mail, DNS, http(s) und SSH werden akzeptiert
iptables -A bad-dmz -p tcp --dport smtp -j ACCEPT
iptables -A bad-dmz -p udp --dport domain -j ACCEPT
iptables -A bad-dmz -p tcp --dport domain -j ACCEPT
iptables -A bad-dmz -p tcp --dport www -j ACCEPT
iptables -A bad-dmz -p tcp --dport https -j ACCEPT
iptables -A bad-dmz -p tcp --dport ssh -j ACCEPT
iptables -A bad-dmz -p icmp -j icmp-acc
iptables -A bad-dmz -j log-and-drop

# Innen -> Aussen-Regelkette
# mail, DNS, http(s) und telnet werden akzeptiert
iptables -A dmz-bad -p tcp --dport smtp -j ACCEPT
iptables -A dmz-bad -p tcp --sport smtp -j ACCEPT
iptables -A dmz-bad -p udp --dport domain -j ACCEPT
iptables -A dmz-bad -p tcp --dport domain -j ACCEPT
iptables -A dmz-bad -p tcp --dport www -j ACCEPT
iptables -A dmz-bad -p tcp --dport https -j ACCEPT
iptables -A dmz-bad -p tcp --dport telnet -j ACCEPT
iptables -A dmz-bad -p icmp -j icmp-acc
iptables -A dmz-bad -j log-and-drop
```

```
# Regelketten für den Rechner selbst
iptables -N bad-if
iptables -N dmz-if
iptables -A INPUT -i $BAD_IFACE -j bad-if
iptables -A INPUT -i $DMZ_IFACE -j dmz-if

# Externe Schnittstelle
# auf diesem Rechner wird nur SSH akzeptiert
iptables -A bad-if -p icmp -j icmp-acc
iptables -A bad-if -p tcp --dport ssh -j ACCEPT
iptables -A bad-if -p tcp --sport ssh -j ACCEPT
ipchains -A bad-if -j log-and-drop

# Interne Schnittstelle
iptables -A dmz-if -p icmp -j icmp-acc
iptables -A dmz-if -j ACCEPT
```

Einige Worte zur Dienst–Qualität ("quality of service"). Linux kann das ToS–("Type of Service")–Feld modifizieren und dessen Wert ändern, um dem Paket eine andere Priorität zu geben. Z.B. ändert der folgende Befehl die ausgehenden SSH–Pakete, um die Verbindungs–Antwortzeit zu verbessern.

```
iptables -A OUTPUT -t mangle -p tcp --dport ssh -j TOS --set-tos
Minimize-Delay
```

In gleicher Weise könnten Sie für FTP–Verbindungen die Option `'--set-tos Maximize-Throughput'` nutzen, um die Übertragungsrate zum Nachteil der Sitzungs–Interaktivität zu ändern.

Das wars. Nun kennen Sie die Grundlagen, um ein effektives Paketfilter–System aufzusetzen. Denken Sie jedoch daran, dass eine Firewall kein Allheilmittel ist, wenn es um Sicherheit geht. Er ist nur eine weitere Vorsichtsmaßnahme. Die Einrichtung einer Firewall entbindet Sie nicht davon, starke Passwörter zu nutzen, die aktuellen Sicherheits–Patches, Einbruchs–Entdeckungssysteme usw.

Referenzen

- Proxy–ARP Mini–HOWTO: http://www.linuxdoc.org/HOWTO/mini/Proxy–ARP–Subnet/index.html
- Netfilter: http://netfilter.samba.org/

Webpages maintained by the LinuxFocus Editor team	Translation information:
© Vincent Renardias	fr –––> –– : Vincent Renardias <vincent(at)renardias.com>
"some rights reserved" see linuxfocus.org/license/	fr –––> en: Georges Tarbouriech <gt(at)linuxfocus.org>
http://www.LinuxFocus.org	en –––> de: Hermann–Josef Beckers <beckerst/at/lst–oneline.de>

2005–01–11, generated by lfparser_pdf version 2.51