Firewall mit iptables Rules

Grundsätzliches

Mithilfe des Userspace-Programms iptables (bzw. ip6tables) lassen sich Ketten und Regeln in Form einer Tabelle erstellen, welche dann von der im Linux-Kernel enthaltenen Firewall abgearbeitet werden.

Das Konfigurieren mittels iptables kann als eine Sicherheitsmaßnahme für sein System verwendet werden, z.B. durch Kontrolle und Einschränkung des Netzverkehrs, Zugriffssperren von Diensten, o.ä., aber auch um Netzwerkverkehr zu manipulieren (z.B. durch Weiter-/und Umleitungen, usw.).

Iptables arbeitet auf der ISO/OSI Transport- / und Vermittlungsschicht. Das bedeutet, es können nicht alle PDU's manipuliert werden: Lediglich UDP-Datagramme, TCP-Segmente und IP-Pakete. Es gibt allerdings auch Möglichkeiten, auf Data-Link Ebene MAC-Adressen Manipulation durchzuführen. Diese Fähigkeit ist für ein Routing ohnehin unabdingbar. Iptables kann eine Penetration des Systems auf Applikationsebene daher nicht verhindern: hierfür müssen andere Maßnahmen getroffen werden (Proxy/Application-Level-Gateway, ...).

Aufbau und Funktion

Funktionsweise - Chains/Ketten

Egal, ob ein Paket vom eigenen Rechner ins Netzwerk geschickt werden soll, oder von außerhalb zum Rechner gelangt, oder aber der Rechner das Paket an der einen Stelle annehmen und einfach zur nächsten weiterleiten soll, so durchläuft dieses Paket bei iptables immer mindestens eine Chain (dt. Kette„). Davon gibt es fünf vordefinierte, wobei nicht jede Tabelle wirklich alle Chains verwendet:

- INPUT - Der Name der Kette zeigt auch schon die Funtkionsweise auf: Ein Paket wird lokal zugestellt.
- OUTPUT - Ein Paket, welches vom eigenen Rechner erzeugt wurde und weggeschickt werden soll, wird mindestens diese Chain durchlaufen.
- FORWARD - Diese Kette dient allen Paketen, welche geroutet aber nicht lokal zugestellt werden.
- PREROUTING - Pakete durchlaufen diese Chain, noch vor dem Routing.
- POSTROUTING - Pakete durchlaufen diese Kette, nachdem sie geroutet wurden, aber noch bevor sie weitergeleitet werden.

Funktionsweise - Policy

Mithilfe von iptables kann man sehr fein definieren, was mit einzelnen Paketen, welche die unterschiedlichen Ketten durchlaufen, passieren soll. So durchläuft ein Paket nacheinander alle Regeln der Kette, bis eine passende gefunden ist.

Erreicht ein Paket aber das Ende der Kette, ohne dass eine der definierten Regeln Einfluss auf das Paket genommen hat, so greift die Policy. Diese Entscheidet dann, was mit all' den Paketen geschehen soll, welche das Ende der Tabelle erreichen. Hierzu dienen meist ACCEPT (Pakete dürfen Filterkette passieren), DROP (Paket wird als ungültig verworfen) und REJECT (Paket wird verworfen und ein ICMP Paket mit einer Meldung wie „port unreachable“ wird zurückgesendet).

Wenn man also z.B. alle eingehenden Pakete, auf welche keine Regel zutrifft, verwerfen möchte, so würde das wie folgt aussehen:

```
# iptables -P INPUT DROP
```

Funktionsweise - Regeln

Regeln werden in jeder Chain aufgestellt und können Pakete unterschiedlicher Protokolle, Herkunft, Ziele, usw. betreffen.

Ein Paket durchläuft von der 1. Regel, bis zur leztzten Regel solange die Kette, bis eine Regel auf das Paket zutrifft, oder das Ende der Kette erreicht ist. In letzterem Fall greift dann die oben beschriebene Policy.

```
# iptables -A INPUT -s 192.168.178.5 -p icmp -j ACCEPT
# iptables -A INPUT -s 192.168.178.61 -p icmp -j DROP
```

in diesem Beispiel, würde ein ICMP-Paket, welche von dem Host mit der IP-Adresse 192.168.178.5 stammt, akzeptiert werden. Das Paket würde an dieser Stelle die Kette verlassen und dürfte passieren. Ein ICMP-Paket, welches von der Adresse 192.168.178.61 stammen würde, würde die Kette bis zur 2. Regel durchlaufen (da die erste nicht auf das Paket zutrifft) und dann laut Regel verworfen werden.

Funktionsweise - Tabellen

Regeln und Ketten werden bei iptables in verschiedenen Tabellen zusammengefasst. Diese Tabellen dienen grundlegenden Aufgaben:

Tabelle	**Aufgabe**
Filter	Die Standardtabelle von iptables. Dient der reinen Paketfilterung. Beinhaltet die Ketten INPUT, OUTPUT, FORWARD.
NAT	Diese Tabelle wird für Adressumsetzung und Routing benötigt. Beinhaltet die Ketten PRE-/POSTROUTING und FORWARD.
MANGLE	Dient der Paketmodifikation. Beinhaltet PRE-/POSTROUTING, FORWARD, INPUT und OUTPUT.

Tabelle	Aufgabe
RAW	Paket durchläuft Ketten dieser Tabelle als Erstes. Dient dazu um Ausnahmen vom Connection Tracking zu definieren oder Pakete mittels TRACE-Kette zu verfolgen. Beinhaltet die Ketten TRACE, PREROUTING, OUTPUT.

Grundlegende Parameter

Hier einmal ein paar grundlegende Parameter, um iptables zu konfigurieren:

Parameter	Beschreibung
-N	Legt eine neue Kette an
-X	Löscht eine leere, selbsterstellte Kette
-P	Ändert die Policy einer Kette
-F	Löscht alle Regeln aus einer Kette
-Z	Paket- und Bytezähler aller Regeln einer Kette = 0
-L	Auflisten aller Regeln
-A	Eine neue Regel in einer Kette erstellen
-I	Eine neue Regel in einer bestimmten Position einfügen
-R	Eine Regel an eine bestimmte Position in der Kette ersetzen
-D <Nummer>	Eine Regel an einer bestimmten Position in einer Kette löschen
-D	Die erste passende Regel in einer Kette löschen

Ein paar Beispiele zur Verdeutlichung einer Möglichen Anwendung:

„Ping of Death“

Mittels des Befehls „ping <IP-Adresse>“ sendet man ein ICMP-Paket zur Zieladresse. Erreicht das Paket den Host, erhält man ein „Reply“ als Antwort zurück. Das kostet (wenn auch nur sehr sehr wenig) Systemressourcen.

So kann ein Angreifer, einfach mit mehreren Rechnern ein „ping 192.168.178.5 -i 0.00000001“ durchführen, wobei dann jeder Rechner alle 0.00000001 Sekunden ein ICMP Paket an die Zieladresse 192.168.178.5 sendet - die Folge ist eine eingehende Flut von ICMP-Paketen beim Zielhost. Um solche „Flutattacken“ abwehren zu können, kann man entsprechende Regeln definieren. Ein Beispiel wäre:

```
# iptables -A INPUT -d 192.168.178.5 -p icmp -j DROP
```

Damit werden alle eingehenden ICMP-Pakete, welche die Zieladresse 192.168.178.5 im Header haben, verworfen.

Mit

```
# iptables -L
```

wird jetzt auch die neue Regel angezeigt.

Ein

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

löscht die angelegte Regel in der Kette INPUT wieder.

Bei einem „Ping of Death“ kann es auch sinnvoll sein, ein REJECT durchzuführen, da oftmals automatisierte Systeme hinter solchen Angriffen stecken, welche bei entsprechenden Fehlermeldungen ihre Angriffe einstellen:

```
# iptables -A INPUT ! -s 192.168.178.7 -d 192.168.178.5 -p icmp -j REJECT --
reject-with icmp-port-unreachable
```

Mit diesem Befehl werden alle ICMP-Pakete, welche nicht (dafür steht das Ausrufezeichen) von der Adresse 192.168.178.7 stammen und das Ziel 192.168.178.5 haben, mit der Nachricht „port unreachable“ zurückgewiesen.

Man kann auch ganz einfach „Kindersicherungen“ anlegen. Möchte man z.B. die politisch umstrittene Internetpräsenz www.wikipedia.de sperren, so reicht ein

```
# iptables -A OUTPUT -d www.wikipedia.de -j DROP
```

vollkommen aus. Es kann allerdings passieren, dass (je nach DNS-Konfiguration/Netzwerk, usw.) auch Subdomains mit gesperrt werden.

Zusätzliche Paket-Manipulation

Neben DROP, REJECT und ACCEPT existieren allerdings noch zusätzliche Optionen zur Paket-Manipulation.

Folgende Optionen stehen unter anderem zur Network Address Translation:

SNAT (Source NAT) - manipuliert die Herkunftsadresse eines Paketes.

DNAT - Destination NAT - manipuliert die Ziel-Adresse eines Paketes.

MASQERADE - „maskiert“ die Herkunftsadresse eines Paketes. Dient vor allem in Netzen, wo IP-Adressen dynamisch zugewiesen werden.

REDIRECT - Diese Option ändert die Ziel-adresse eines Paketes in die des lokalen Rechners.

Explizite Erweiterungen

Neben der schon recht umfangreichen Liste an Optionen, bieten sich noch explizite Erweiterungen an. Diese lassen sich mit dem Parameter -m aufrufen:

-m mac → Hiermit kann man MAC-Adressen in Frames manipulieren.

-m limit → Mit dieser Option lassen sich Regeln mit „Limits“ erweitern. So kann man z.B. festlegen, dass nur 1 ICMP-Paket / Sekunde akzeptiert wird.

-m owner → manipuliert Pakete auf UID/GID/PID hin.

-m state → Hiermit können Segmente auf Verbindungszustände hin gefiltert werden. Mögliche Verbindungszustände sind wie folgt:

Status	**Beschreibung**
NEW	Ein Segment, welches eine neue Verbindung aufbaut.
ESTABLISHED	Pakete und Segmente, welcher zu einer bereits aufgebauten Verbindung gehören.
RELATED	Betrifft „verwandte“ Segmente und Pakete. Z.B. wären FTP Daten auf Port 20 und 21 verwandt.
INVALID	Pakete, welche nicht identifiziert werden können.

Protokollierung

Nachdem nun die ersten Schritte mit „iptables“ beschrieben wurden, widmet dieser Abschnitt sich einem der wichtigsten, wenn nicht sogar dem wichtigsten Thema: Logging.

Es ist möglich, ebenfalls mittels „Regeln“, auch wieder sehr fein eingestellt, alles mögliche mitzuprotokollieren. Dazu gibt ein Ziel: LOG.

Es werde ausgehenden DNS-Anfragen protokolliert:

```
# iptables -A OUTPUT -p udp --dport 53 -j LOG
#iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
```

Normalerweise durchläuft ein Paket die Tabelle nicht mehr, sobald DROP, ACCEPT oder REJECT über das Schicksal des Paketes entscheiden. Bei LOG wird das Paket allerdings nur „geloggt“, muss aber dann die Kette weiterdurchlaufen. Daher sollte man darauf achten, dass man ein LOG macht, bevor mal ein Paket verwirft, wenn weitere Regeln würden nicht erfolgen!

Man kann hinter -j LOG noch einige weitere Optionen dranhängen, welche das Log-Verhalten beeinflussen:

Optionen/Präfix	**Beschreibung**
–log-level <Level>	Passt die Priorität der Meldung(en) an. Damit ist es sogar möglich, über Konfiguration von /etc/syslog.conf in unterschiedliche Dateien zu protokollieren.
–log-uid	Protokolliert bei einem lokal erzeugten Paket (also ein Paket, welches die OUTPUT-Kette passiert) die UID des Benutzers mit, welcher das Paket erzeugt hat.
–log-tcp-sequence	Protokolliert die Sequenznummern der einzelnen TCP-Verbindungen mit.
–log-tcp-options	Protokolliert mögliche Optionen des TCP-Headers mit.
–log-ip-options	Protokolliert mögliche Optionen des IP-Headers mit.
–log-prefix <prefix>	

Dadurch wird eine bessere Lesbarkeit möglich, außerdem lassen sich so leichter bestimmte Protokollgruppen rausfiltern (z.B. mittels grep)|

Von den Log-Leveln gibt es 7 Stück:

Level	Bedeutung
1	alert
2	critical
3	error
4	warning
5	notice
6	info
7	debug

Eine solche Log-Ausgabe sieht im ersten Moment etwas kryptisch aus, enthält auf den zweiten Blick allerdings alle notwendigen Informationen:

```
# Nov  8 16:54:04 Uranus kernel: [15061.388110] IN= OUT=wlan0
SRC=192.168.178.23 DST=192.168.178.1 LEN=60 TOS=0x00 PREC=0x00 TTL=64
ID=20331 DF PROTO=UDP SPT=46779 DPT=53 LEN=40
```

In diesem Fall sieht man nochmal deutlich; das ausgehende UDP-Datagram auf Port 53 (und einige Zusatzinformationen), welches von der internen IP-Adresse 192.168.178.23 an das Default-Gateway gesendet wird, aber keinen Inhalt der Applikationsebene. Falls es sich um einen Log-Eintrag eines TCP-Segmentes handelt, so werden auch Window-Size, reservierte Bits im TCP-Header und Flags (z.B. SYN,ACK) mitprotokolliert.

Was ebenfalls auffällt: „Uranus“. Das ist der Name des Rechners, welcher den Log-Eintrag getätigt hat. Hierbei sei die Möglichkeit erwähnt, den BSD-Syslogd auch über das Netzwerk nutzen zu können und so einen externen Protokoll-Server anzulegen. Außerdem: Das Jahr wird nicht mitprotokolliert. Falls man dieses benötigt, so muss man dies händisch einfügen.

Aber wo finde ich jetzt diesen Log-Eintrag? In /var/log/syslog. Aber damit sich die iptables-Logs nicht mit den anderen aus syslog mischen, sollte man diese umlenken. Dazu wird die Datei /etc/rsyslog.conf (als root) erweitert. Idealerweise, kann man an dieser Stelle dann die –log-level Option ausnutzen:

```
kern.alert                      /var/log/kern.alert.log #enthält Level 1
Logs
kern.critical                   /var/log/kern.critical.log #enthält Level 2
Logs
kern.error                      /var/log/kern.error.log #enthält Level 3
Logs
kern.warning                    /var/log/kern.warning.log #enthält Level 4
Logs
kern.notice                     /var/log/kern.notice.log #enthält Level 5
Logs
kern.info                       /var/log/kern.info.log #enthält Level 6 Logs
kern.debug                      /var/log/kern.debug.log #enthält Level 7
Logs
```

Hierbei sollte man allerdings beachten, dass auch alle anderen Kernel-Meldungen nach in die entsprechenden Log-Dateien geschrieben werden. Es ist sinnvoll hierfür die –log-prefix Option zu

nutzen, um daraus später vernünftige Protokolle basteln zu können.

Falls man einen anderen System-Logger als den BSD-Syslogd verwendet (z.B. syslog-ng) so müssen andere Konfigurationsdateien geändert werden, damit die Ausgaben in entsprechende Log-Files (oder wohin auch immer) geschrieben werden.

Bei der Protokollierung sollte einem bewusst sein, dass je nach Nutzung des Netzwerks, durchaus Logfiles von >500MByte/Tag entstehen können. Hier ist also Vorsicht geboten.

Type of Service

Im IP-Header kann das 8Bit große Type of Service Feld manipuliert und so die Priorisierung des Paketes gesetzt werden (Quality of Service).

Betrachte man folgendes Beispiel:

```
# iptables -t mangle -A PREROUTING -p tcp --dport 22 -j TOS --set-tos 0x10
```

Hierbei werden die IP-Pakete bereits in der MANGLE-Tabelle, schon vor dem eigentlichen Routing manipuliert. Die oben beschriebene Regel trifft auf das SSH-Protokoll zu. mit –set-tos gefolgt von einem Hexadezimalwert, kann man nun den Type of Service in einem IP-Paket beeinflussen. Das bedeutet, man kann den Transport von Paketen, anderen vorziehen. Folgende Werte beeinflusst man mit –set-tos:

Value	**Effekt**
0x10	Minimale Verzögerung
0x08	Maximaler Durchsatz
0x04	Maximale Zuverlässigkeit
0x02	Minimale Kosten
0x00 (default)	Normaler Service

iptables Regeln dauerhaft speichern

Dieser Abschnitt zeigt verschiedene Möglichkeiten, wie iptables Rules unter Linux dauerhaft gespeichert werden können.

```
# iptables-save
```

Die eigentlichen iptables Rules werden auf der Kommandozeile mit dem Kommando iptables für IPv4 und ip6tables für IPv6 erstellt und angepasst. Dies geschieht über das Firewall Script /etc/init.d/firewall In eine Datei können diese mit dem Kommando iptables-save für IPv4 gespeichert werden.

Debian/Ubuntu:

```
# iptables-save > /etc/iptables/rules.v4
```

Diese Datei kann danach wieder mit dem Kommando iptables-restore für IPv4 geladen werden.

Debian/Ubuntu:

```
# iptables-restore < /etc/iptables/rules.v4
```

Wenn auch IPv6 Regeln verwenden möchten, können diese ebenso in eine eigene Datei gespeichert werden.

Debian/Ubuntu:

```
# iptables-save > /etc/iptables/rules.v6
```

Das automatische Laden der konfigurierten iptables Rules kann mit folgenden Methoden bewerkstelligt werden:

iptables-persistent für Debian/Ubuntu

Seit Ubuntu 10.04 LTS (Lucid) und Debian 6.0 (Squeeze) gibt es ein Paket namens „iptables-persistent“ welches das automatische Laden der gespeicherten iptables Rules übernimmt. Dafür müsssen die Rules in der Datei /etc/iptables/rules.v4 für IPv4 und in /etc/iptables/rules.v6 für IPv6 gespeichert werden.

Für die Verwendung muss lediglich das Paket installiert werden.

```
# apt-get install iptables-persistent
```

Ältere iptables-persistent Versionen (z.b. jene bei Debian Squeeze) unterstützen noch keine IPv6 Rules. Dort gibt es nur eine Datei namens /etc/iptables/rules für IPv4. Wichtig!!! Prüfe nach erfolgter Konfiguration unbedingt ob die Rules wie gewünscht nach einem Reboot geladen werden.