

# **Datenkommunikation Prof. Dr. Marke**

SS 2006

## **Seminar-Thema : IPTABLE Aufbau und Funktion**

Name : Kulyk Nazar  
Matr.Nr: 258360030100  
Sem. : 8W  
Stud.Richtung: Informatik

URL : <http://myeburg.net/dako/index.html>

Datum : 17.05.2006

# 1. Übersicht

---

Netfilter ist eine Software innerhalb des Linux-Kernels, die es erlaubt, Netzwerkpakete abzufangen und zu manipulieren. Es bildet damit das Herzstück einer Firewall auf Basis von Linux.

Iptables ist das dazugehörige Dienstprogramm zur Konfiguration von Netfilter.

Die beiden Begriffe Netfilter und Iptables werden oft austauschbar für die Summe aus den Kernel- und den Userspace- Bestandteilen verwendet.

Das Netfilter/Iptables - Gespann kann die wesentlichen Protokolle des Internets aus der Vermittlungsschicht, der Transportschicht und teilweise auch aus der Anwendungsschicht verarbeiten (siehe Internet-Protokoll-Familie). Für die darunterliegende Netzzugangsschicht gibt es dagegen Programme wie ebtables (Ethernet-Bridge Tables).

Zu den Funktionen gehören:

- Paketfilterung einschließlich Stateful Inspection bzw. connection tracking. Netfilter kann auch mit FTP und anderen Protokollen umgehen.
- Network Address Translation (NAT) einschließlich Masquerading und Portweiterleitung

Der Kernel-Anteil, also Netfilter in engerem Sinne, ist in eine Vielzahl von Kernel-Modulen aufgeteilt, so dass sich einzelne Bausteine und Netfilter insgesamt leicht ein- und ausschalten lassen.

Netfilter und Iptables gehören zum Lieferumfang aller neueren Linux-Distributionen und sind der Standard für Firewalls unter Linux. Es wird vom netfilter.org - Projekt unterstützt und weiterentwickelt. Die Software unterliegt, wie der Linux-Kernel insgesamt auch, der GNU General Public License, ist also freie Software.

## 2. Geschichte

---

Vor dem Kernel 2.0 gab es bereits zahlreiche "Progrämmchen", die allerdings meist als Patch in den Kernel implantiert werden mußten, um diverse Effekte zu erreichen. Diese Programme waren zwar vom Code her bemerkenswert, doch bei weitem nicht so elegant wie die heute eingesetzte Architektur.

Mit dem Kernel 2.0 tauchte "ipfwadm" auf. Es war ein durchaus leistungsfähiges System, doch es gab nur die Möglichkeit, eingehende, geroutete oder ausgehende Pakete zu filtern.

Mit dem 2.2er Kernel erblickte "ipchains" die Linux-Welt. Hier gab es bereits mehrere "chains", die aus jeweils einzelnen Regeln bestanden, die der Reihe nach getestet wurden. Die erste erfolgreiche Regel verließ das Regelwerk und bestimmte den Werdegang des gerade getesteten Paketes. Darüber hinaus konnte man in "ipchains" erstmals eigene Ketten definieren, die den Charakter von Unterprogrammen hatten.

Der Kernel 2.4 brachte ebenfalls wieder zahlreiche Neuerungen. Die "Netfilter"-Architektur bringt ein neues Tool namens "iptables" mit. Verbesserungen gab es bei:

- konsistenterer Namensgebung
- drei Tabellen, die aus mehreren "Chains" bestehen
- stateful inspection jetzt möglich
- Port forwarding mit dem selben Tool
- snat, dnat, masquerading
- DoS Limiting
- Ethernet MAC Adress-Filtering
- variables Logging
- rejects mit einstellbarem Verhalten
- bessere Routing-Kontrolle
- flexiblere Architektur

### 3. Aufbau

---

#### Tabellen (tables)

Die iptables-Architektur gruppiert die Regeln für die Verarbeitung von Netzwerk-Paketen gemäß ihrer Funktion in drei Tabellen:

- **filter** für Paketfilter. Die Standard-Tabelle, die immer dann verwendet wird, wenn keine Tabelle explizit angegeben wird. Diese Tabelle besteht aus den chains INPUT, FORWARD und OUTPUT. Eventuell lassen sich in dieser Tabelle weitere benutzerdefinierte Chains unterbringen.
- **nat** für Network Address Translation. Diese Tabelle ist für alle Arten von Adress-Umsetzungen oder Port-Forwarding verantwortlich und besteht aus den Chains PREROUTING, OUTPUT und POSTROUTING. Die in dieser Tabelle befindlichen Chains werden für jedes erste Paket einer neuen Verbindung aufgerufen und führen entsprechende Änderungen an den Port- oder IP-Nummern der Pakete durch.
- **mangle** für Paketmanipulationen. In dieser Tabelle existieren die Chains PREROUTING und OUTPUT und hier werden spezielle Änderungen an Paketen vorgenommen.

Diese Tabellen enthalten Ketten (engl. *chains*) von Verarbeitungsregeln, bestehend aus Mustern (engl. *patterns*), die bestimmen auf welche Pakete die Regel angewendet wird und Ziele (engl. *targets*), die festlegen, was mit den Paketen passiert. Anders gesagt bestimmen Chains also *wo* geprüft wird, Pattern *welche* Pakete betroffen sind und Targets *was* mit ihnen geschieht. Die Filterregeln werden dabei außer bei Ausnahmen stets sequentiell bis zum ersten Treffer abgearbeitet.

## Ketten (chains)

Iptables hat fünf fest vorgegebene Ketten (built-in chains) im Kernel eingebaut:

- **PREROUTING**: Unmittelbar, bevor eine Routing-Entscheidung getroffen wird, müssen die Pakete hier durch (nur *nat* und *mangle*)
- **INPUT**: Hier landen alle Pakete, die an einen lokalen Prozeß gerichtet sind (nur *filter* und *mangle*)
- **OUTPUT**: Hier laufen alle Pakete durch, die von einem lokalen Prozeß stammen
- **FORWARD**: für alle zu routenden Pakete, also Pakete die für andere Rechner bestimmt sind (nur *filter* und *mangle*)
- **POSTROUTING**: Alle Pakete, lokale und solche die geroutet werden, laufen hier durch (nur *nat* und *mangle*)

Zusätzlich lassen sich noch eigene Ketten definieren.

## Ziele (targets)

Jede Kette kann Regeln enthalten. Jede Regel besteht dabei aus einer Filterspezifikation und aus einem Ziel (target). Das Ziel gibt letztendlich an, was mit einem Paket passiert. Ein Ziel kann eine benutzerdefinierte Kette, ein Standardziel oder ein erweitertes Ziel sein. Für die fest vorgegebenen Ketten kann man eine **Policy** definieren, die angewandt wird, wenn keine der Regeln greift. Eine Policy ist immer ein Standardziel. Default ist ACCEPT.

Folgende Standardziele gibt es:

- **ACCEPT**: Das Paket wird akzeptiert.
- **DROP**: Das Paket wird ohne Rückmeldung an den Sender verworfen.
- **QUEUE**: Das Paket wird in eine Queue im Userspace geschickt, sodass es von einem Benutzerprogramm bearbeitet werden kann. Wird die Queue von keinem Programm gelesen, hat dies denselben Effekt wie DROP.
- **RETURN**: In benutzerdefinierten Ketten angewendet wird die Abarbeitung dieser Kette abgebrochen und mit der nächsten Regel der vorhergehenden (aufrufenden) Kette fortgeführt. Bei den fest vorgegebenen Ketten greift die Policy der Kette.

Daneben gibt es zahlreiche erweiterte Ziele. Die wichtigsten sind:

- **LOG**: Das Paket wird mit ausgewählten Informationen im Syslog aufgezeichnet und anschließend weiter durch die Kette geleitet.

- **REJECT**: Das Paket wird verworfen und der Sender darüber informiert (mittels ICMP-Nachricht oder RST-Packet bei TCP-Verbindungen)

Folgende Ketten sind nur in der nat-Tabelle gültig:

- **DNAT** und **SNAT** (nur PREROUTING und OUTPUT bzw. POSTROUTING) stehen für Destination NAT bzw. Source NAT. Dabei wird die Ziel- bzw. die Quell-Adresse des Paketes durch eine angegebene Adresse ersetzt.
- **MASQUERADE** (nur POSTROUTING) ist eine Spezialform von Source NAT. Die Quell-Adresse des Paketes wird dabei durch die IP-Adresse der Schnittstelle ersetzt, auf welcher es den Rechner verlassen wird.
- **REDIRECT** (nur PREROUTING und OUTPUT) leitet das Paket zum lokalen Rechner um; wird beispielsweise für transparente Proxyserver benötigt.

Bei DNAT, SNAT und MASQUERADE *merkt* sich Netfilter die Adressübersetzung - daher *connection tracking* - und wendet sie auf alle nachfolgenden Pakete derselben Verbindung (in beide Richtungen) ebenfalls an.

## Muster (patterns)

Muster machen einen Großteil der Regeln aus, da sie die Bedingungen enthalten, auf die Pakete hin geprüft werden. Diese können sowohl auf OSI-Schicht 3 stattfinden, als auch darunter (z.B. das Filtern von MAC-Adressen per `--mac-source`) oder darüber (z.B. bestimmte Protokolle herausfiltern mit `--tcp`, `--udp` oder `--icmp`). Neben generischen Mustern enthält iptables viele spezialisierte Matches, die über dynamisch geladene Erweiterungen zur Verfügung stehen und mit dem Schalter `-m` oder `--match` geladen werden.

## 4. Erweiterungen

---

Die iptables-Organisation ist zusätzlich für die Projekte *Patch-o-matic* (kurz POM) und *Patch-o-matic-ng* (ng = "next generation") verantwortlich, welche experimentelle Funktionen erproben und dessen Erkenntnisse evtl. in zukünftige iptables-Versionen miteinfließen. Die Erweiterungen fügen neue Module hinzu, indem mit einem Skript der Quellcode von iptables der netfilter-Teil des Kernels verändert wird. Damit die Änderungen übernommen werden, müssen beide Komponenten nach dem Patchen neu kompiliert werden und der Rechner (oder die virtuelle Maschine) neugestartet werden.

*Siehe auch:* <http://www.netfilter.org/patch-o-matic/index.html> und <http://www.netfilter.org/patch-o-matic/pom-base.html>.

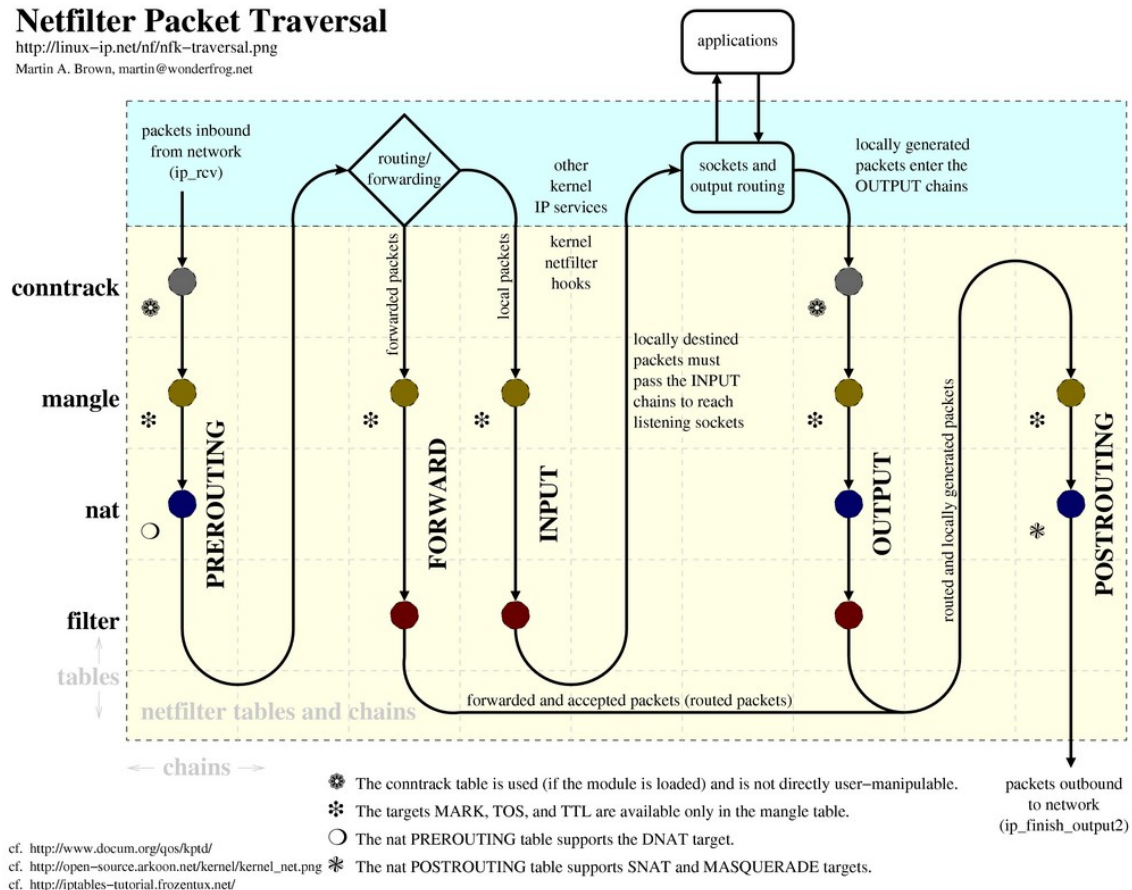
---

## 5. Grundsätzliche Funktionsweise

### Netfilter Packet Traversal

<http://linux-ip.net/nf/nfk-traversal.png>

Martin A. Brown, [martin@wonderfrog.net](mailto:martin@wonderfrog.net)



Um die Funktion besser zu verstehen, gilt es, sich zunächst den Weg, den ein Datenpaket durch den Kernel macht, vorzustellen. Gerade hier haben sich die gravierendsten Änderungen zur bisherigen Architektur gezeigt.

Die "chains" haben also mit der Lage der Datenpakete bei ihrem Weg durch den Kernel zu tun. Um bestimmte Zwecke zu erreichen, bzw. bestimmte Schutzvorrichtungen aufzubauen, sind entsprechende Regeln in die jeweiligen "chains" einzutragen.

- INPUT Hier landen alle Pakete, die an einen lokalen Prozeß gerichtet sind.
- OUTPUT Hier laufen alle Pakete durch, die von einem lokalen Prozeß stammen.
- PREROUTING Unmittelbar, bevor eine Routing-Entscheidung getroffen wird, müssen die Pakete hier durch.
- FORWARD für alle zu routenden Pakete
- POSTROUTING Alle Pakete, die geroutet werden, laufen nach dem Routing hier durch.

## 6. Aufruf-Konventionen von iptables

### Grundsätzlich gilt:

- Kommandos bestehen aus einem GROSS-Buchstaben, z.B. -L oder einer Lang-Form z.B. --list
- Targets (d.h. Ziele oder Aktionen) bestehen aus einem Wort in GROSS, z.B. DROP
- Chains bestehen aus einem Wort in GROSS, z.B. PREROUTING
- Tabellen sind Worte in Kleinbuchstaben, z.B. filter
- Optionen bestehen aus Kleinbuchstaben, z.B. -t oder --source-port

### Stateless Firewalling - Filterregeln

Diese Funktionen stellen das Grundgerüst eines Paketfilters dar. Fast alle Funktionen dieser Art gab es bei den Vorgängern bereits. Nur eben die Syntax ist ein bißchen anders als gewohnt.

Alle Stateless-Funktionen werden in der filter-Tabelle in den Chains INPUT, OUTPUT und FORWARD vorgenommen.

INPUT Hier stehen Regeln zum Schutz des lokalen Rechners

OUTPUT evtl. fehlskonfigurierte Programme "taub" machen

FORWARD zum Schutz von Rechnern, zu denen geroutet werden muß

Mit diesen Regeln lassen sich einfache Schutzmechanismen aufbauen - allerdings nur wirklich einfache Mechanismen. Denn eine Prüfung kann nur für jedes Paket einzeln durchgeführt werden und kennt keine Zustände. So sind z.B. Befehle wie "ping" generell verboten und Ergebnisse von Name-Server-Abfragen generell erlaubt, was Angreifern leicht die Möglichkeit eines Eindringens bieten kann und gleichzeitig starke Einschränkungen darstellt.

### Stateful Firewalling - erweiterte Filterregeln

Wie wir gerade gesehen haben, reicht eine Prüfung auf Paket-Ebene nicht aus, um ausreichenden Schutz zu bieten. Besseren Schutz bietet ein Mechanismus, der den Zustand aller Verbindungen, die durch die Firewall aktiv bestehen, mitverfolgt und dementsprechend reagiert. Diese Funktion führt das Modul "ipt\_state" aus, welches mit `-m state` aktiviert wird und über die Datei `/proc/net/ip_conntrack` mit der Außenwelt kommuniziert. Ein gelegentlicher Blick in diese Datei verrät einiges über das, was dieses Modul tut...

Aufrufkonventionen:

`-m` Aktivieren des Moduls "ip\_state".

`state`

`[!] --state` Prüfen, ob Paket in einem gewählten  
`<status>` Verbindungszustand ist.

mögliche Zustände:

INVALID	Ungültiger Zustand, dieses Paket eröffnet weder eine neue Verbindung, noch gehört es zu einer bestehenden.
NEW	Dieses Paket eröffnet eine neue Verbindung.
ESTABLISHED	Dieses Paket gehört zu einer bereits bestehenden Verbindung
RELATED	Dieses Paket hat etwas mit einer bestehenden Verbindung zu tun.

## Port-Forwarding und Maskierung

Hinter diesen Schlagworten verstecken sich eigentlich ganz einfach zu verstehende Mechanismen. Pakete werden so geändert, daß sie

- an andere Maschinen gerichtet sind (Destination NAT). Hierbei wird einfach nur eine andere Ziel-IP-Nummer in das Paket eingetragen. Dies muß in der PREROUTING Chain der Tabelle "nat" erfolgen.
- von einer anderen Maschine zu kommen scheinen (Source NAT). Hierbei wird die Quell-IP-Adresse des Paketes ausgetauscht. Dieser Schritt kann nur in der POSTROUTING Chain der "nat" Tabelle durchgeführt werden.
- an einen anderen Port gerichtet werden.
- von einem anderen Port zu kommen scheinen.

Typische Anwendungen dieser Methoden sind das Verstecken von Rechnern hinter der Firewall, das Verschleiern von IP-Adressen oder das gemeinsame Nutzen einer (z.B. vom Provider) vorgegebenen IP-Adresse durch mehrere Nutzer. Alle nachfolgend aufgelisteten Targets werden nur für das erste Paket einer Verbindung spezifiziert. Alle nachfolgend gesandten/empfangenen Pakete werden automatisch korrekt modifiziert.

## Protokollieren

Nicht nur zum Test, auch zur Überwachung von "schrägen" Mustern von IP-Paketen ist es sinnvoll, über diverse Dinge Protokoll zu führen. Dies kann mit dem LOG-Target sehr einfach geschehen. Alles Protokollierte wird im Syslog (`/var/log/messages`) mitgeschrieben.

## Wichtige Informationen

Einige der bereits im Text genannten Dateien des `/proc` Dateisystems seien an dieser Stelle nochmals genannt. Sie ermöglichen in beschränktem Umfang, einen Blick in die Arbeitsweise der Netfilter-Architektur zu werfen:

`/proc/net/ip_conntrack` zeigt alle derzeit aktiven Verbindungen (selbst bei verbindungslosen Protokollen wie UDP oder ICMP) an.

`/proc/net/ip_tables_names` Enthält die Liste aller zur Zeit geladenen Tables.  
`/proc/net/ip_queue` Wird von den UserSpace-Queue-Routinen zur Kommunikation genutzt.

## 7. Mögliche Probleme

Probleme beim Aufbau einer Firewall bereiten vor allem ältere Protokolle, wie

- FTP
- Portmap/NFS/YP
- IRC

Während sich die Problematik mit FTP seit der Einführung von iptables gelegt hat, machen die Portmap-basierten Dienste wie NFS oder YP nach wie vor die Konfiguration einer sicheren Firewall problematisch.

Für die Probleme, die aus dem 2-Port Protokoll von FTP resultieren, schafft ein Kernel-Modul Abhilfe. Dieses observiert den Verbindungsaufbau einer FTP-Verbindung auf Protokoll-Ebene, und erkennt den Rückkanal automatisch.

```
/sbin/modprobe ip_conntrack_ftp
```

## 8. Verwendete Quellen

---

Literatur:

- Gregor N. Purdy: *Linux iptables - kurz & gut*. O'Reilly, Dezember 2004, ISBN 3897215063
- Gregor N. Purdy: *Linux iptables Pocket Reference*. O'Reilly Media, August 2004, englisch, ISBN 0596005695
- Ralf Spenneberg: *Linux-Firewalls mit iptables & Co., m. CD-ROM*. Addison-Wesley, München, September 2005, ISBN 3827321360

WebLinks:

- <http://www.netfilter.org> - Homepage des Netfilter-Teams
- <http://iptables-tutorial.frozentux.net/> - Umfangreichstes Tutorial für iptables (Englisch)
- [http://www.pl-forum.de/t\\_netzwerk/iptables.html](http://www.pl-forum.de/t_netzwerk/iptables.html) - Umfangreicher deutschsprachiger Workshop
- <http://www.linuxguruz.com/iptables/> - Englische Linksammlung zum Thema
- <http://www.egocrew.de/fw> - Online Iptables-Generator
- <http://www.linux-magazin.de/Artikel/ausgabe/2000/06/IPTables/iptables.html> - Linux Magazin - IPTables
- <http://www.linux-user.de/ausgabe/2002/05/030-firewall/firewall-4.html> - Linux User - Firewall
- <http://17-filter.sourceforge.net/> - Application Layer Packet Filtering

## 9. Fragen

1. Aus welchen zwei Teilen besteht ein Iptables Basiertes Packet Filter?  
Netfilter – Kernelspace Bestandteil und Iptables Userspace Bestandteil
2. Wie kann man eine Packet Überwachung im Firewall durch Syslog aufbauen?  
Mit Hilfe von einer LOG Zeil werden Paketed mit ausgewählten Informationen im Syslog aufgezeichnet und anschließend weiter durch die Kette geleitet.
3. Welches Mechanism bezeichnet man mit Destination NAT und Source NAT?  
Beide Mechanismen werde beim Maskierung und Port-Forwarding verwendet. Hierbei wurden einfach nur eine andere Ziel-IP, oder Quel-IP in das Paket eingetragen.