# Linux-PCs sichern mit iptables



## Einordnung und Abgrenzung

- Clients: Desktop-PCs, Notebooks unter Linux
- Server: z.B. Web-Sever, o.dgl.

```
# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
#
```

- "Personal Firewall"
  - eine von viele Schutzmaßnahmen
  - Bestandteil der *host based security*
  - Wirksamkeit umstritten (BSI empfiehlt den Einsatz)
  - filtert ein- und ausgehenden Datenverkehr eines PCs
  - Ziel: ungewollten/unbeabsichtigten Datenverkehr unterbinden
  - Voraussetzung: genaue und detaillierte Kenntnisse über Anwendungsprotokolle
- hier: Basismechanismen und -werkzeuge
- nicht:
  - grafische Frontends für Regel-Definitionen
  - Linux als Firewall-System zum Schutz von Netzen

## Was ist netfilter/iptables?

- Paket-Filter im Linux-Kernel (ab Version 2.4)
  - inspiziert IP-Pakete
  - entscheidet auf Grund von Regeln, wie mit einem IP-Paket zu verfahren ist

- konfiguriert durch das Kommando **`iptables`**
- realisiert als Kernel-Moduln

```
# ls /lib/modules/$(uname -r)/kernel/net/ipv4/netfilter
arptable_filter.ko          ipt_addrtype.ko    ipt_NETMAP.ko
arp_tables.ko               ipt_ah.ko          ipt_NOTRACK.ko
arpt_mangle.ko              ipt_CLASSIFY.ko    ipt_owner.ko
ip_conntrack_amanda.ko      ipt_comment.ko     ipt_physdev.ko
ip_conntrack_ftp.ko         ipt_conntrack.ko   ipt_pkttype.ko
ip_conntrack_irc.ko         ipt_dscp.ko        ipt_realm.ko
ip_conntrack.ko             ipt_DSCP.ko        ipt_recent.ko
ip_conntrack_proto_sctp.ko  ipt_ecn.ko         ipt_REDIRECT.ko
ip_conntrack_tftp.ko        ipt_ECN.ko         ipt_REJECT.ko
ip_nat_amanda.ko            ipt_esp.ko         ipt_SAME.ko
ip_nat_ftp.ko               ipt_helper.ko      ipt_sctp.ko
ip_nat_irc.ko               ipt_iprange.ko     ipt_state.ko
ip_nat_snmp_basic.ko        ipt_length.ko      ipt_tcpmss.ko
ip_nat_tftp.ko              ipt_limit.ko       ipt_TCPMSS.ko
ip_queue.ko                 ipt_LOG.ko         ipt_tos.ko
iptable_filter.ko           ipt_mac.ko         ipt_TOS.ko
iptable_mangle.ko           ipt_mark.ko        ipt_ttl.ko
iptable_nat.ko              ipt_MARK.ko        ipt_ULOG.ko
iptable_raw.ko              ipt_MASQUERADE.ko
ip_tables.ko                ipt_multiport.ko
#
```

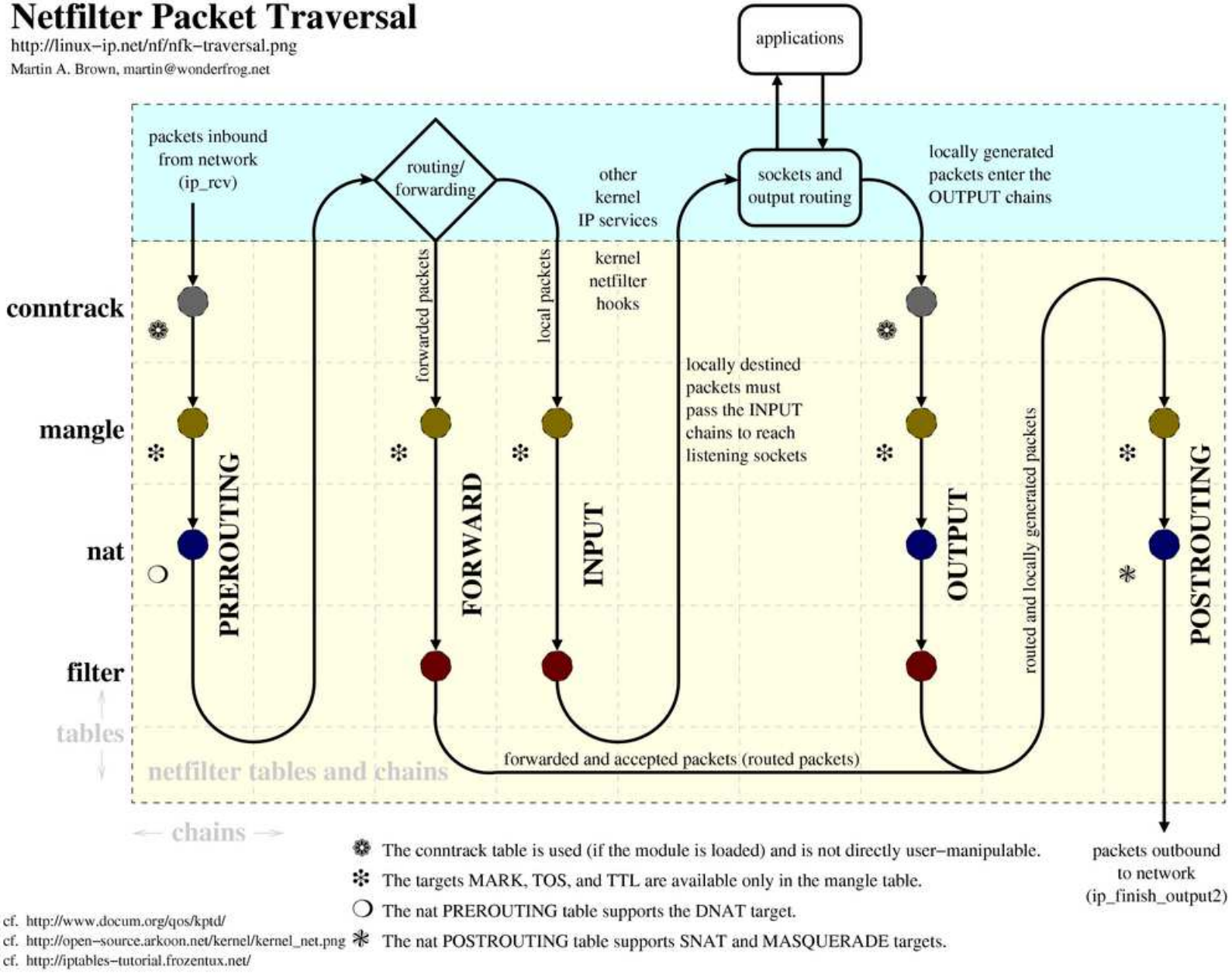
- es werden nur die Moduln geladen, für die Regeln definiert sind (bzw. zugehörige Helper-Moduln)

```
# lsmod|grep ip
ipv6                  240097  100
ipt_REJECT             10561  3
ipt_LOG                10049  1
ipt_multiport           5953  3
ipt_state               5825  4
ip_conntrack           45573  1 ipt_state
iptable_filter          6721  1
ip_tables              21441  5 ipt_REJECT,ipt_LOG,ipt_multiport,ipt_state,iptable_filter
#
```

## Funktionsprinzipien von iptables

- ein IP-Paket durchläuft mehrere **chains** / **tables**
- **chains** enthalten Regeln
- eine Regel definiert ein Muster (Bedingungen, denen ein Paket entsprechen soll, z.B. bestimmte Ziel-Adresse, bestimmte Flags im TCP-Header, o.dgl.) und ein Ziel (**target**)
- ein **target** ist eine weitere **chain** oder ein default-target (z.B. **`DROP`**, **`ACCEPT`**, ...)
- ein Paket wird an das target weitergeleitet, wenn es dem in der Regel definierten Muster entspricht (**match**)
- d.h.: sobald ein **match** erreicht ist, werden nachfolgende Regeln in dieser Tabelle nicht mehr angewendet
- wird kein **match** erreicht, bestimmt letztendlich die **policy** das Verhalten
- **policy** definiert eines der default-target

### Paketverarbeitung: das ganze Bild




## Tabellen (tables)

- regeln unterschiedliche Arten der Paketverarbeitung

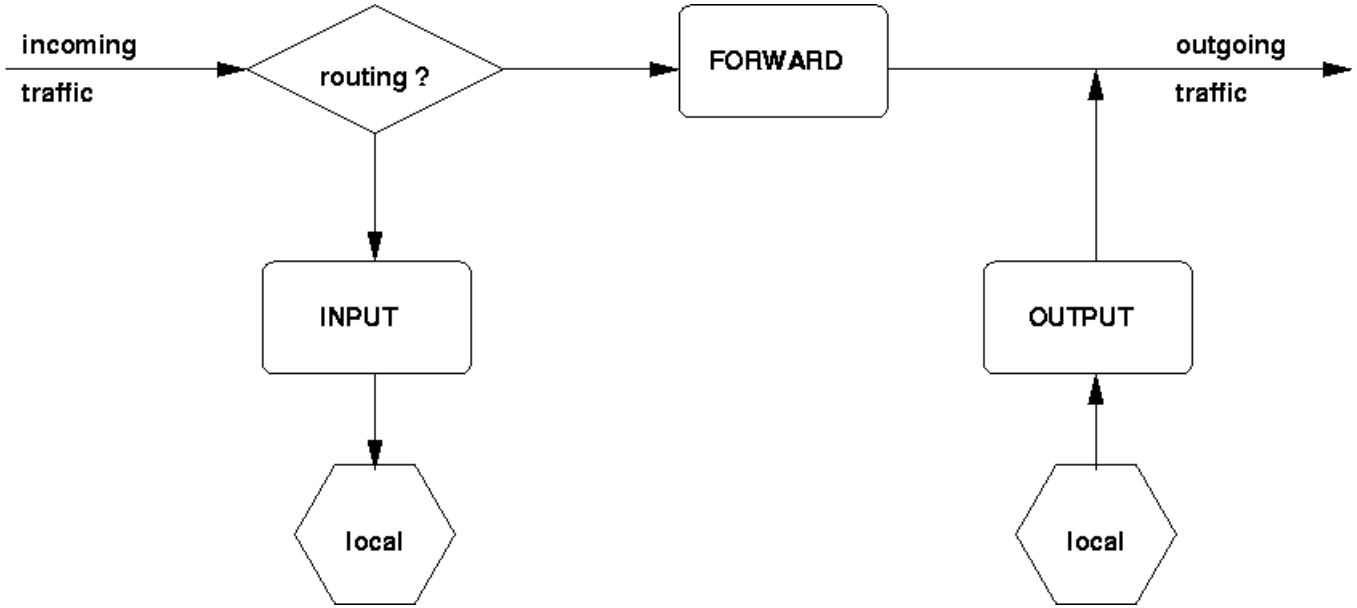
- grundlegende Gruppierung der Pakete
- jede Tabelle enthält vordefinierte und ggf. nutzerdefinierte **chains**
- **conntrack**
  - Connection Tracking
  - erkennen von zusammengehörigen Paketen (z.B. TCP-Session, UDP-Stream, Analyse von Anwendungsprotokollen)
  - realisiert sog. **statefull packet filter**
  - nicht veränderbar: es können keine Regeln angegeben werden
- **mangle**
  - bestimmte Änderungen an Paketen, z.B. TTL (time-to-live), TOS (Type of Service)
  - Pakete markieren
  - chains: alle
- **nat**
  - Network Address Translation
  - Änderung von IP-Adressen/Port-Nummern
  - SNAT (Soure-NAT):
    - Ändern der Quell-Adresse
    - chains: **`POSTROUTING`**
    - auch: Masquerading (Umgang mit dynamischen IP-Adressen, setzen der Quell-IP-Adresse auf IP-Adresse des outgoing Interfaces)
  - DNAT (Destination-NAT):
    - Ändern der Ziel-Adresse
    - chains: **`PREROUTING`** und **`OUTPUT`**
    - Redirection: Ändern des Ziel-Ports
- **filter**
  - Paket-Filter
  - chains: **`INPUT`**, **`OUTPUT`** und **`FORWARD`**
  - oftmals benutzerdefinierte chains
- wenn für eine Table keine Regeln definiert sind, entfällt die komplette Table (der Kernel-Modul wird nicht geladen)
- Maschinen, die kein Routing vornehmen, benutzen i.d.R. nur die filter-Table

## Regelketten (chains)

IP-Pakete passieren die einzelnen chains unter den folgenden Bedingungen (die einzelnen Tables enthalten ggf. nicht alle chains)

- **PREROUTING**
  - unmittelbar nach dem Eintreffen an der Maschine
  - vor der routing-Entscheidung
- **INPUT**
  - Ziel: lokaler Prozess
- **FORWARD**
  - Ziel: nicht lokal, Paket hat externen Empfänger
- **OUTPUT**
  - Quelle: lokaler Prozess
- **POSTROUTING**
  - vor dem Verlassen der Maschine

## Paketverarbeitung (nur filter)




## Das Kommando iptables

- Einrichten, Manipulieren und Anzeigen der iptables-Regeln im Kernel

### Aufrufkonventionen, Syntax, etc.

- siehe Punkt 7 in Wolfgang Kinkeldei: iptables - Die Firewall des Kernels 2.4

## "Personal Firewall"

- viele Linux-Distributionen enthalten vordefinierte Regeln
- diese Regeln implementieren ein "Personal Firewall"
- Aktivieren/Deaktivieren durch rc-Scripte

### Beispiel

- aus Red Hat Systemen (hier: Fedora Core 4)
- eingestellt durch **`/usr/bin/system-config-securitylevel`**
- aktiviert/deaktiviert durch **`/etc/rc.d/init.d/iptables`** vor bzw. nach Starten/Stoppen der Netzwerk-Konfiguration

```
# grep -v "^#" /etc/sysconfig/iptables
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
```

```
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -p tcp -m tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
#
```

- nur Tabelle **`filter`**
- default-Policy für alle chains: **`ACCEPT`**
- chains **`INPUT`** und **`FORWARD`** werden umgelenkt in chain **`RH-Firewall-1-INPUT`**
  - benutzerdefinierte chains werden oft in der Art eines Unterprogramms eingesetzt
- chain **`OUTPUT`** ist leer (nur default-policy)
  - Unterstellung: wir versenden keine "bösen" Pakete
- chain **`RH-Firewall-1-INPUT`** akzeptiert folgende Pakete:
  - alle Pakete, die über Interface **`lo`** hereinkommen
  - alle Pakete der Protokolle **`icmp`**, **`ipv6-crypt`** (50), **`ipv6-auth`** (51)
  - UDP-Pakete mit dem Ziel-Port 5353 und der Ziel-Adresse 224.0.0.251 (Multicast-DNS, mDNS - ZeroConf)
  - UDP-Pakete mit dem Ziel-Port 631 (Internet Printing Protocol, ipp - CUPS)
  - alle Pakete, die zu einer Verbindung (connection) im Zustand **`ESTABLISHED`** oder **`RELATED`** gehören, z.B.:
    - alle Pakete von korrekt aufgebauten, abgehenden TCP-Sessions
    - alle Pakete von selbst initierten UDP-Streams
  - alle Pakete, die eine neue TCP-Verbindung mit dem Zielport 22 (ssh) initiieren
- alle anderen Pakete werden zurückgewiesen (**`REJECT`**) mit einem icmp-Paket vom Typ **`icmp-host-prohibited`**
- somit ist erreicht:
  - eintreffende Pakete werden zurückgewiesen, wenn es keine "Antwort-Pakete" auf vorher selbst initiierte Verbindungen sind
  - Regeln sind unabhängig von IP- und Netzwerk-Konfiguration

### Erweiterung: zurückgewiesene Pakete protokollieren

```
# iptables -I RH-Firewall-1-INPUT 9 -j LOG --log-level info --log-prefix "iptables REJECT:"
```

- Pakete werden protokolliert über **`syslogd`** (Log-Level **`kern.info`**)
- Log-Meldungen erhalten den Prefix "iptables REJECT:"

```
# grep "iptables REJECT" /var/log/messages
...
Apr 10 09:51:40 muethos kernel: iptables REJECT:IN=eth0 OUT= MAC=ff:ff:ff:ff:ff:ff:00:04:ac:33:7b:26:08:00 SRC=0.0.0.0 DST=255.255.255.255 LEN=356 TOS=0x00 PREC
...
```

- Beachte: Protokollieren kann zu dramatischem Anstieg der Filegröße von Logfiles führen!
- Regelset kontrollieren, ggf. Protkollierung wieder ausschalten

```
# iptables -D RH-Firewall-1-INPUT 9
```

## Einzelne Dienste einrichten

- "Personal Firewall"-Einstellungen realisieren i.d.R. diese Strategie
  - alle selbst iniitierten Verbindungen (inkl. dazugehörigem Datenverkehr von angesprochenen Servern) soll erlaubt sein
  - Maschine soll auf unverlangt empfangene Pakete nicht reagieren, d.h. die Maschine betreibt selbst keine Server
  - oftmals eine Ausnahme: SSH-Server
- viele Dienste kann man relativ einfach erlauben, falls benötigt

    - siehe Beispiel `sshd`
- Dienste mit komplexeren Protokollen oder spezifische Ansprüche sind aufwändiger zu realisieren

## AFS-Client

- basiert auf UDP
- Datenverkehr wird durch Client initiiert, "Personal Firewall" enthält alle erforderlichen Regeln
- jedoch erforderlich: Erhöhen des Timeout-Wertes für UDP-Streams

```
# sysctl -w net.ipv4.netfilter.ip_conntrack_udp_timeout_stream=720
```

- besser als Eintrag in **`/etc/sysctl.conf`**
- Hintergrund:
    - default-Wert: 180 Sekunden
    - AFS-Server kontaktiert AFS-Client, um diesem mitzuteilen, dass er seine Filekopie im lokalen Cache ungültig ist (z.B. weil ein anderer Client das File modifiziert hat)
    - bei unverändertem default-Wert, würden diese Pakete nur passieren können, wenn der Client seinerseits innerhalb der letzten 180 Sekunden den Server kontaktiert hat - und damit entsprechender Eintrag in Cor vorhanden ist
    - AFS-Client kontaktiert seinerseits im Abstand von 10 Minuten alle ihm bekannten AFS-Server - um deren Erreichbarkeit zu prüfen
    - durch Erhöhen des Timeout-Wertes auf 12 Minuten ist gesichert, dass Einträge in der Connection-Tracking-Table des Kernels nicht verfallen

## FTP-Client

- komplexes Protokoll (Control Connection, Data Connection)
- zwei Modi des Protokolls: aktiv vs. passiv (aus Sicht des Servers)
- passiv
    - Client initiiert Control Connection zum Server (TCP-Port 21)
    - Client wählt passiv-Modus (FTP-Kommando **`PASV`**)
    - Server teilt Client Port-Nummer für Data Connection mit, sobald Client eine Data Connection wünscht
    - Client initiiert Data Connection zum Server
    - keine Änderung der Filter-Einstellung auf Client-Seite erforderlich
- aktiv
    - Client initiiert Control Connection zum Server (TCP-Port 21)
    - Client teilt Server TCP-Port mit, der für Data Connection benutzt werden kann (FTP-Kommando **`PORT`**)
    - Server initiiert Data Connection zum Client
    - Verbindungsaufbau zum Client wird durch filter-Regeln unterbunden
    - Lösung:
        - Kernel-Module **`ip_conntrack_ftp`** analysiert **`PORT`** -Kommando und kennzeichnet die entsprechende IP-Connection in **`/proc/net/ip_conntrack`** als **`RELATED`**
        - filter-Regeln lassen den Verbindungsaufbau zu
        - Laden des Moduls
            - **`modprobe ip_conntrack_ftp`**
            - Eintrag in **`/etc/sysconfig/iptables-config`**

## Web-Server

- Default-Policy: was nicht nicht explizit erlaubt ist, ist verboten
- Dienste http/https (TCP-Ports 80/443)
- Gateway zu whois (TCP-Port 43) und finger (TCP-Port 79)
- keine Einschränkungen innerhalb des Campusnetzes
- Besonderheit: abgehender Verkehr ist eingeschränkt

```
# cat /etc/sysconfig/iptables
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
```

```
#
-N LOGREJECT
-A LOGREJECT -j LOG --log-level info  --log-prefix "iptables REJECT:"
-A LOGREJECT -p tcp -j REJECT --reject-with tcp-reset
-A LOGREJECT -j REJECT --reject-with icmp-port-unreachable
#
-N CLEANCHECK
-A CLEANCHECK -m state --state RELATED,ESTABLISHED -j ACCEPT
-A CLEANCHECK -s 127.0.0.1 -d 127.0.0.1 -j ACCEPT
-A CLEANCHECK -p tcp ! --syn -m state --state NEW -j REJECT --reject-with tcp-reset
#
-A INPUT -j CLEANCHECK
-A INPUT -s 134.109.0.0/16 -p icmp -j ACCEPT
-A INPUT -s 134.109.0.0/16 -p tcp -m state --state NEW -j ACCEPT
-A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
-A INPUT -p icmp -m icmp --icmp-type port-unreachable -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -j LOGREJECT
#
#
-A OUTPUT -j CLEANCHECK
-A OUTPUT -p tcp --tcp-flags RST RST -m multiport --sports 80,443 -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -d 134.109.0.0/16 -p udp -j ACCEPT
-A OUTPUT -d 134.109.0.0/16 -p tcp -m state --state NEW -j ACCEPT
-A OUTPUT -p tcp -m multiport --dports 43,79 -j ACCEPT
-A OUTPUT -p udp -m udp --dport 53 -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -j LOGREJECT
COMMIT
```

## Literatur und Verweise

### Bücher

Barth, Wolfgang
Das Firewall-Buch
Grundlagen, Aufbau und Betrieb sicherer Netzwerke mit Linux
ISBN 3-89990-128-2
Nicolaus Millin Verlag GmbH, 2004

Cheswick, W.R.; Bellovin, S.M.; Rubin, A.D.
Firewalls und Sicherheit im Internet
Schutz vor cleveren Hackern
ISBN 3-8273-2117-4
Addison-Wesley Verlag, 2004

### Links

http://de.wikipedia.org/wiki/Netfilter/iptables - *gute deutsche Zusammenfassung bei Wikipedia*

http://www.pro-linux.de/t_netzwerk/print/iptables.html - *deutsches Tutorial*

http://iptables-tutorial.frozentux.net/ - *umfangreiches Tutorial*

http://www.kalamazoolinux.org/presentations/20010417/conntrack.html - *Connection Tracking*

http://www.netfilter.org/ - *Homepage von netfilter/iptables*