15. Firewalls unter Unix

Gliederung

- Grundlagen
- **Linux - iptables**
- OpenBSD - PF Toolkit
- BSD, Solaris - IPF Toolkit
- Löcher in Firewalls - Virtuelle Private Netzwerke

LINUX

Geschichte:

- Paketfilter existieren seit 1994 im Kernel. Sie basierten auf ipfw von BSD. Die Administration erfolgte ab Kernel 2.0 mittels ipfwadm.
- 1998 wurde eine vollständige Überarbeitung für den Kernel 2.2 vorgenommen. Als Administrationswerkzeug wurde ipchains bereitgestellt.
- Mit Einführung des Kernels 2.4 wurde das Paket netfilter für den Kernel als Firewall-Software bereitgestellt. Dieses wird mit dem Tool iptables administriert, das die Verwaltung der Regeln mittels einer Tabellenstruktur unterstützt.

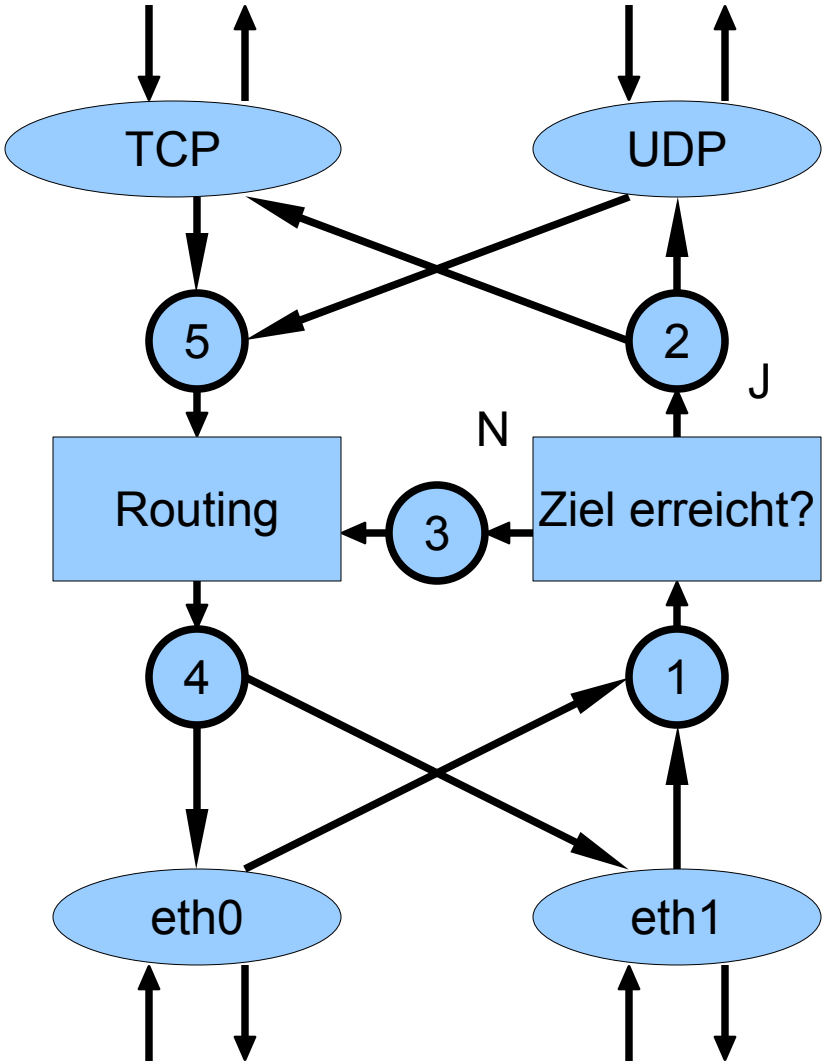
Leistungen von netfilter

- stateless Paket filtering, stateful Paket filtering
- NAT (source NAT, destination NAT, transparent Proxy mit Redirect
- Pakete manipulieren
- unterstützt Traffic Control und QoS
- unterstützt Ipv4 und IPv6

netfilter-Architektur(1)

- Die Verarbeitung von Datenpaketen wird durch Linux ab Kernel 2.4 durch ***netfilter*** gesteuert:
  - Bei Ipv4 sind 5 Hooks im Protokollstack definiert, an denen Kernelmodule zur Überwachung des Paketflusses angedockt werden können.
  - Jeder angedockte Kernelmodul kann ein Paket an dem entsprechenden Punkt analysieren und das Paket anschließend durchlassen, verändern oder verwerfen.
  - An jedem Hook kann eine Liste von Regeln zur Anwendung kommen (Chain).
- Mit Hilfe von ***netfilter*** sind folgende Funktionen realisierbar:
  - Firewall zum Filtern von Paketen nach unterschiedlichen Kriterien einschließlich zustandsbezogener Filterung
  - Network Address Translation (NAT) zum Ändern von Adressen und Portnummern bei eingehenden und ausgehenden Paketen für Masquerading und Load Balancing
  - Erfassung von Daten für Statistik und Fehlerdiagnose

netfilter Architektur (2)



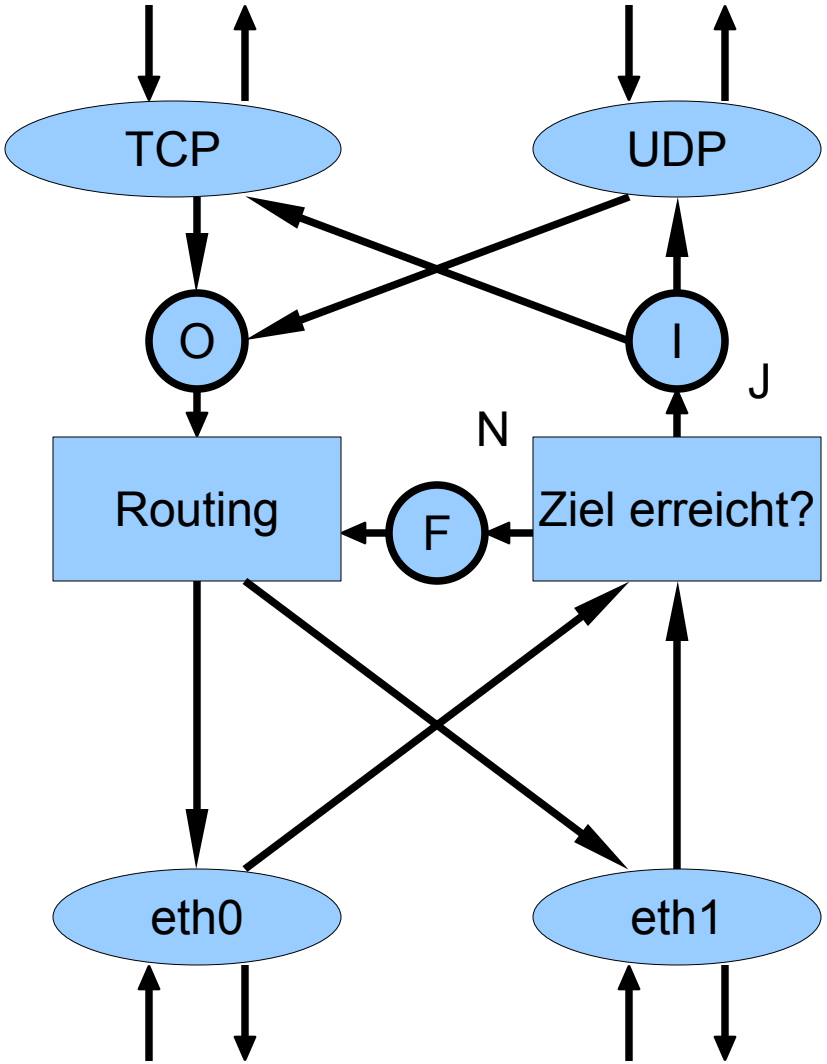
1 Pre-Routing
2 Local input
3 Forward
4 Post-Routing
5 Local Output

netfilter Architektur(3)

iptables setzt auf netfilter auf:

- ***iptables*** liefert verschiedene Tabellen mit, in denen Chains vordefiniert sind. z.B. ***filter*** (ist Standard):
  - INPUT
  - OUTPUT
  - FORWARD
- Erreicht ein Paket einen Hook entscheiden die Regeln in der Chain über den weiteren Weg des Paketes:
  - ACCEPT - das Paket wird weitergeleitet
  - DROP - das Paket wird gelöscht
- Der Defaultzustand für alle Chains ist ACCEPT.
- Jede Regel einer Chain kann in Abhängigkeit von Merkmalen eines Paketes die weiter Verfahrensweise für dieses Paket bestimmen.
- Regeln können zu einer Chain hinzugefügt werden.
- Regeln gehen bei einem Neustart verloren!!! Müssen also in Scripten für den Neustart fixiert werden.

netfilter Architektur (4)



Vordefinierte Chains von ***iptables*** in der Tabelle ***filter***:

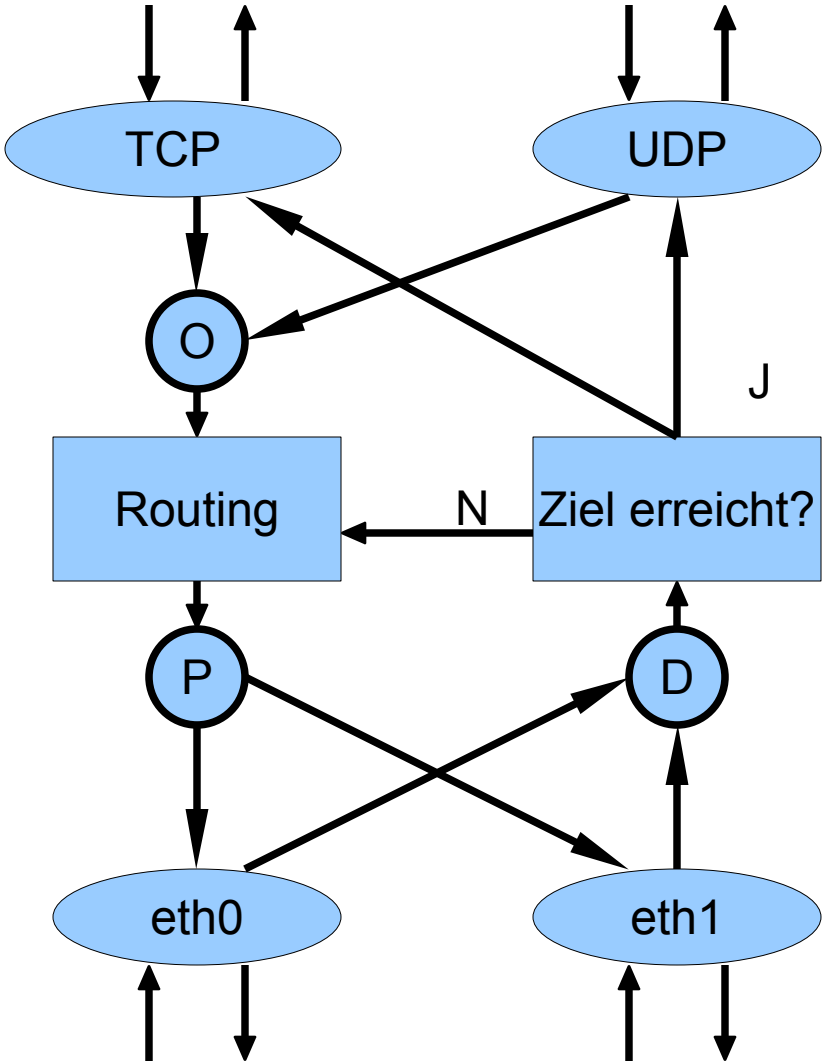
I INPUT
O OUTPUT
F FORWARD

netfilter Architektur(5)

iptables und NAT:

- NAT kann Adressen und Portnummern in IP-Paketen verändern
- NAT war in der ursprünglichen Konzeption von IP nicht vorgesehen und widerspricht dem Prinzip der „Ende zu Ende“ Kommunikation. Dadurch kann es bei höheren Protokollschichten zu Problemen kommen!!!
- Arten von NAT
  - Source NAT (SNAT, Masquerading): Verändert die Source-Adresse eines Paketes. Wird benutzt um Absenderadressen zu verschleiern. Ist notwendig wenn Pakete aus einem privaten Netz in das Internet gesendet werden sollen. Wird ausgeführt bevor ein Paket einem Interface zum Transport übergeben wird (Postrouting)
  - Destination NAT (DNAT): Verändert die Destination-Adresse eines Paketes. Wird für Load Balancing und Port Forwarding benötigt. Wird ausgeführt nachdem ein Paket von einem Interface übernommen wurde.
- Die ***nat***-Tabelle bei ***iptables*** definiert 3 Chains: PREROUTING, POSTROUTING und OUTPUT

netfilter Architektur (6)



Vordefinierte Chains von ***iptables*** in der Tabelle ***nat***:

D PREROUTING
P POSTROUTING
O OUTPUT

15.1 Firewall - Linux

iptables(1)

Einführendes Beispiel: Schützen eines Rechners vor eingehenden Verbindungen über Interface „eth0“

```
# neue Regelkette „block“ erzeugen
iptables -N block
# bestehende Verbindungen erlauben
iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
# neue Verbindungen, die nicht von eth0 kommen, erlauben
iptables -A block -m state --state  NEW -i !eth0 -j ACCEPT
# den Rest verbieten
iptables -A block -j DROP
# die Regelkette „block“ an Standard-Chain INPUT und FORWARD
# anhängen
iptables -A INPUT -j block
iptables -A FORWARD -j block
```

iptables(2)

Probleme bei ***iptables***:

Bei jedem „reboot“ gehen unsere mühsam erzeugten Regeln verloren. Deshalb gibt es zwei schöne Hilfsprogramme:

iptables-save [-c] [-t tables]

Schreiben der Inhalte der Tabellen auf die Standardausgabe

-c mit aktuellen Zählerständen

-t tables nur für die angegebene Tabelle schreiben

iptables-restore [-c] [-n]

Rücksetzen der Tabellen mit den Werten von der Stadardeingabe

-c Rückspeichern der Zählerstände aus dem File

-n kein Löschen (flush) der Ketten vor dem Rückspeichern (append-Modus)

15.1 Firewall - Linux

iptables(3)

- Module (nicht alle)
  - ip_tables - Filtering, NAT, Masquerading (Basis)
  - ip_conntrack - Connection Tracking
  - iptable_filter - INPUT, OUTPUT, FORWARD-Chains
  - iptable_mangle - Veränderung von Inhalten (TOS, TTL)
  - iptable_nat - NAT-Support
  - ipt_LOG - erfassen von Protokolldaten
  - ipt_limit - Limit-Unterstützung für DoS Abwehr
  - ipt_MASQUERADE - Masquerade für DHCP, PPP, SLIP
  - ipt_owner - owner von Sockets werden untersucht
  - ipt_REJECT - reply anstelle von DROP
  - ipt_conntrack_ftp - Connection Tracking FTP
  - ip_conntrack_irc - Connection Tracking irc
  - ip_nat_ftp - NAT bei ftp
  - ip_nat_irc - NAT bei irc

Das Kommando **iptables**(1)

iptables [**-t** *table*] **-**[**AD**] *chain rule-specification* [*options*]
iptables [**-t** *table*] **-I** *chain [rulenum] rule-specification* [*options*]
iptables [**-t** *table*] **-R** *chain rulenum rule-specification [*options*]*
iptables [**-t** *table*] **-D** *chain rulenum* [*options*]
iptables [**-t** *table*] **-**[**LFZ**] [chain] [*options*]
iptables [**-t** *table*] **-N** *chain*
iptables [**-t** *table*] **-X** [*chain*]
iptables [-t *table*] **-P** *chain target* [*options*]
iptables [**-t** *table*] **-E** *old-chain-name new-chain-name*

Das Kommando **iptables**(2)

-t *table* Spezifiziert auf welche Tabelle das Kommando angewendet werden soll. Ist der zugehörige Kernelmodul nicht geladen, wird dieser automatisch geladen. Folgende Tabellen *table* gibt es:

- **filter**: Standard-Tabelle, wenn nicht angegeben. Definiert folgende Chains:
  - INPUT: für lokal zuzustellende Pakete
  - OUTPUT: für Pakete, die lokal erzeugt wurden
  - FORWARD: für Pakete die nicht lokal zustellbar sind
- **nat**: Tabelle für Adressumsetzung (beim Erzeugen einer neuen Verbindung). Definiert folgende Chains:
  - PREROUTING: Umsetzung der Zieladresse
  - POSTROUTING: Verschleierung der Absenderadresse
  - OUTPUT: Behandlung der lokal erzeugten Pakete
- **mangle**
- **raw**

Das Kommando **iptables**(3)

-t *table* Spezifiziert auf welche Tabelle das Kommando angewendet werden soll. Ist der zugehörige Kernelmodul nicht geladen, wird dieser automatisch geladen. Folgende Tabellen *table* gibt es:

- **filter**
- **nat**
- **mangle**: Tabelle für die gezielte Veränderung von Pakten (TOS,TTL, MARK). Definiert folgende Chains:
  - PREROUTING: Umsetzung der Zieladresse
  - OUTPUT: Modifizierung der lokal erzeugten Pakete
  - POSTROUTING: Umsetzung der Absenderadresse
- **raw**: Wird vor allen anderen Tabellen eingehängt. Definiert folgende Chains:
  - OUTPUT: für ausgehende Pakete
  - PREROUTING: für eingehende Pakete

Das Kommando **iptables**(4)

-N *chain*

Neue leere Kette *chain* erzeugen. Diese muss dann mit Regeln gefüllt werden und kann an eine andere Kette angefügt werden.
Beispiel:

```
iptables -N neu
iptables -A neu -i eth1 -j ACCEPT
iptables -A neu -i !eth1 -j DROP
iptables -A INPUT -j neu
```

-A *chain rule-specification*

Anfügen der durch *rule-specification* angegeben Regel an eine bestehende Kette *chain*.
Beispiel:

```
iptables -A INPUT -p udp --sport 53 -j ACCEPT
```

Das Kommando **iptables**(5)

-D *chain rule-specification*

-D *chain rulenum*

Löschen von Regeln aus der Kette chain mit der Regelspezifikation *rule-specification* bzw. der Nummer *rulenum*.

Beispiele:

```
iptables -D INPUT --dport 53 -j DROP
iptables -D INPUT 8
```

-R *chain rulenum rule-specification*

Ersetzen der Regel *rulenum* in der Kette *chain* durch eine neue Regel *rule-specification*

Beispiel:

```
iptables -R INPUT 1 -s 141.20.20.51 -j DROP
```

Das Kommando **iptables**(5)

-I *chain [rulenum] rule-specification*

Einfügen einer durch *rule-specification angegebenen* neuen Regel in die Kette *chain* als Regel Nummer *rulenum*.

-L [*chain*] [*options*]

Anzeigen aller Regeln der spezifizierten Kette *chain*.

Optionen:**-v** (verbose) und **-n** (numerische Ausgabe)

Beispiel:

```
iptables -L
iptabels -L INPUT
iptables -t filter -L INPUT -v
iptables -t filter -L INPUT -v -n
( /sbin/SuSEfirewall2 debug )
```

Das Kommando **iptables**(6)

-F [chain]

Löschen aller Regeln der angegebenen Kette chain.
Wenn keine Kette angegeben ist, werden alle Ketten der angegebenen Table gelöscht (standart: filter):
Beispiel:

```
iptables -t nat -F
```

-Z [chain]

Löschen aller Paketzähler in der angegebenen Kette chain.
Wenn keine Kette angegeben ist, werden alle Zähler aller Ketten der angegebenen Table gelöscht.
Beispiel:

```
iptables -t filter -Z INPUT
iptabels -t filter -LZ INPUT  # vorher Zählerstand ausgeben
```

Das Kommando **iptables**(7)

-X [*chain* **]**

Streichen der nutzerdefinierten Kette *chain*. Alle Referenzen auf diese Kette müssen vorher gelöscht worden sein!!

Beispiel:

```
iptables -F ; iptables -X    # alles weg
```

-P *chain target*

Setzen der Standard-Policy *target* für die angegeben Kette *chain*, wenn keine Regel innerhalb der Kette zuschlägt.

Zulässige Standard-Policy sind:

ACCEPT - Erlaubt QUEUE - in Nutzerspace

DROP - Verboten RETURN – verlassen der Kette

Beispiel:

```
iptables -P INPUT DROP
```

Das Kommando **iptables**(8)

-E *old-chain new-chain*

Umbenennen der Kette old-chain in die Kette new-chain. Dadurch wird nur der Name der Kette geändert. Für die Kette old-chain darf keine Referenz existieren.

Beispiel:

iptables -E alt neu

Das Kommando **iptables**(9)

Optionen

-v

Ausführliche Informationen bei der Ausführung des Kommandos.

Gültig für folgende Kommandos:

-L, -A, -I, -D, -R

-x

Vollständige Zahlenangaben bei Zählern.

Gültig für folgende Kommandos:

-L

-n

Numerische Angaben für Adressen und Ports.

Gültig für folgende Kommandos:

-L

Das Kommando **iptables**(10)

Optionen

--line-numbers

Erzeugen von Zeilennummern bei einer Ausgabe.

Gültig für folgende Kommandos:

-L

z.B. iptables -L INPUT -v -x --line-numbers

-c counter

Setzen von des Zählers für eine Regel. Notwendig beim Rückspeichern von Regeln (iptables-restore).

Gültig für folgende Kommandos:

-I, -A, -R

Das Kommando **iptables**(11)

Regelspezifikation(1)

Die Regelspezifikation besteht in der Regel aus mehreren „Matches“ und einem Target-Teil. Die „Matches“ werden logisch durch ***und*** verknüpft. Die „Matches“ dienen dazu Eigenschaften von Paketen festzustellen. Wenn alle „Matches“ einer Regel erfüllt sind, wird der zugehörige Target-Teil ausgeführt. Sonst wird die nächste Regel der Kette ausgeführt.

rule-specification ::= matches { matches } target

target ::= **-j ACCEPT** | **-j DROP** | **-j** *chain*

ACCEPT -Paket durchlassen

DROP - Paket wegwerfen

chain - Verzweigen in eine neue Kette *chain*

Das Kommando **iptables**(12)

Regelspezifikation(2)

-p [**!**] *protokol*

Paket enthält Informationen zu dem Protokoll *protokol*.

Für protokoll zulässige Werte:

UDP, TCP, ICMP, ... - Werte aus */etc/protocols*

Nummer des Protokolls(ICMP = 1, TCP = 6, UDP = 7, ...)

! vor dem Protokoll bedeutet: Paket entspricht **nicht** dem Protokoll *protokoll*

Beispiel:

```
iptabels -A INPUT -p tcp -j ACCEPT
iptabels -A INPUT -p ! udp -j DROP
```

Das Kommando **iptables**(13)
Regelspezifikationen(3)

-s [!] *ipaddress*
--src [!] *ipaddress*
--source [!] *ipaddress*

Die Source-IP-Adresse des Paketes stimmt mit der angegebenen IP-Adresse *ipaddress* (mit **!** nicht) überein. Folgende Formate sind für die Spezifikation einer IP-Adresse zugelassen:

nnn.nnn.nnn.nnn - Reine IP-Adresse
141.20.20.50
nnn.nnn.nnn.nnn/mm - Subnetz mit mm signifikanten Bits
141.20.20.0/24
nnn.nnn.nnn.nnn/mmm.mmm.mmm.mmm
- Subnetz mit Netzmaske

Beispiel:

iptables -A INPUT -s 141.20.1.3 -j ACCEPT

Das Kommando **iptables**(14)
Regelspezifikationen(4)

-d [**!**] *ipaddress*

Die Ziel-IP-Adresse des Paketes stimmt mit der angegebenen IP-Adresse ipaddress (bei **!** nicht) überein. Folgende Formate sind für die Spezifikation einer IP-Adresse zugelassen:

nnn.nnn.nnn.nnn - Reine IP-Adresse
141.20.20.50

nnn.nnn.nnn.nnn/mm - Subnetz mit mm signifikanten Bits
141.20.20.0/24

nnn.nnn.nnn.nnn/mmm.mmm.mmm.mmm
- Subnetz mit Netzmaske

Beispiel:

```
iptables -A INPUT -d 141.20.21.141 -j ACCEPT
iptables -A INPUT -d 141.20.21.0/255.255.255.0 -j ACCEPT
iptables -A INPUT -d  ! 141.20.21.0/24 -j DROP
```

Das Kommando **iptables**(15)
Regelspezifikationen(5)

-i [!] *interface*

Das Paket kommt über das Interface *interface* in den Rechner herein (mit **!** nicht über dieses Interface). Nur für INPUT-, FORWARD- und PREROUTING-Ketten zulässig. „+“ als Platzhalter für Buchstaben und Ziffern zulässig

Beispiele:

```
iptables -A INPUT -i eth0 -j ACCEPT
iptables -A INPUT -i ! eth0 -j DROP
iptables -A INPUT -i eth+ -j ACCEPT
iptables -A INPUT -i ! ppp0 -j ACCEPT
```

Das Kommando **iptables**(16)
Regelspezifikationen(6)

-o [!] *interface*

Das Paket wird über das Interface *interface* gesendet (mit **!** nicht über dieses Interface). Nur für OUTPUT-, FORWARD- und POSTROUTING-Ketten zulässig. „+“ als Platzhalter für Buchstaben und Ziffern zulässig

Beispiele:

```
iptables -A OUTPUT -o eth0 -j ACCEPT
iptables -A OUTPUT -o ! eth0 -j DROP
iptables -A OUTPUT -o eth+ -j ACCEPT
iptables -A OUTPUT -o ! ppp0 -j ACCEPT
```

Das Kommando **iptables**(17)

Regelspezifikationen(7)

[!]-f

Das Paket ist das 2. oder 3. Fragment eines bereits empfangenen Pakets oder bei **!** der erste Fragment eines Paketes.

Beispiel:

```
iptables -A INPUT -f -j ACCEPT
```

Das Kommando **iptables**(18)

Implizite Regelspezifikationen für TCP(1)

Für TCP-Pakete gelten die folgenden Matches. Die Option **-p TCP** ist dafür in jedem Fall notwendig!!!

--sport [!] *port* {,*port*}

--sport [!] *port:[port]*

Paket kommt von dem angegebenen Port. Es können mehrere Ports spezifiziert werden. port:port spezifiziert einen Bereich von Ports. Neben den Portnummern sind auch die Namen aus der */etc/services* benutzbar.

Beispiele:

```
iptables -A INPUT -p TCP --sport  999 -j ACCEPT
iptables -A INPUT -p TCP --sport 1:1023 -j ACCEPT
```

Das Kommando **iptables**(18)

Implizite Regelspezifikationen für TCP(2)

--dport [**!**] *port* {,*port*}

--dport [**!**] *port*:[*port*]

Paket wird von dem angegebenen Port gesendet. Es können mehrere Ports spezifiziert werden. *port:port* spezifiziert einen Bereich von Ports. Neben den Portnummern sind auch die Namen aus der */etc/services* verwendbar.

Beispiele:

```
iptables -A INPUT -p tcp --dport  999 -j ACCEPT
iptables -A INPUT -p tcp --dport 1:1023 -j ACCEPT
iptables -A INPUT -p tcp --dport smtp -j ACCEPT
iptables -A INPUT -p tcp --dport 1024: -j ACCEPT
```

Das Kommando **iptables**(19)

Implizite Regelspezifikationen für tcp(3)

--tcp-flags [!] *mask flag*

Es wird geprüft, ob das spezifizierte Flag *flag* in der durch die Maske *mask* spezifizierte Menge von Flags in dem TCP-Paket enthalten (bei „!“ nicht enthalten) ist. Maske *mask* ist eine durch Komma getrennte Folge von Flags. Folgende Flags sind möglich:

SYN, RST, ACK, FIN

ALL – Alle Flags (für Maske)

NONE – kein Flag (für Flagspezifikation)

Beispiele:

iptables -A INPUT -p TCP --tcp-flags SYN,FIN,ACK SYN -j DROP

iptables -A INPUT -p TCP --tcp-flags ALL SYN -j ACCEPT

Das Kommando **iptables**(20)

Implizite Regelspezifikationen für tcp(4)

--tcp-option [**!**] *number*

Es wird geprüft ob die angegebenen Optionsnummer *number* in dem TCP-Paket vorhanden ist (bei „**!**“ nicht vorhanden ist).

Beispiele:

iptable -A INPUT -p tcp --tcp-option 16

iptabel -A INPUT -p tcp --tcp-option ! 16

Das Kommando **iptables**(21)

Implizite Regelspezifikationen für UDP

-p udp ist für UDP-Pakete in diesem Zusammenhang notwendig!

--sport [**!**] *port*

--sport [**!**] *port*:[*port*]

Wie **--sport** bei TCP.

Beispiel:

```
iptables -A INPUT -p udp --sport 53 -j ACCEPT
```

--dport [**!**] *port*

--dport [**!**] *port*:[*port*]

Wie **--dport** bei TCP.

Beispiel:

```
iptables -A OUTPUT -p udp --dort 53 -j ACCEPT
```

Das Kommando **iptables**(22)

Implizite Regelspezifikationen für ICMP

Die Option **-p icmp** ist in diesem Zusammenhang notwendig!!

--icmp-type [**!**] *typename*

--icmp-type [**!**] *typenummer*

Überprüfung von ICMP-Paketen auf den Typ *typename* bzw. *typenummer*. Liste der gültigen Typename *typename* erhält man durch:

iptables -p icmp -h

Beispiel:

iptables -A INPUT -p icmp –icmp-type 8 -j ACCEPT

iptables -A INPUT -p icmp –icmp-type echo-request -j ACCEPT

Das Kommando **iptables**(23)
Explizite Regelspezifikationen(1)

Bei expliziten Regelspezifikationen muss durch die Option **-m** bzw. **--match** explizite eine spezielle Menge von Regeln vorbestimmt werden.

-m ah

-m esp

Wird für die Protokolle AH und ESP bei IPSEC benötigt.

--ahspi

```
iptables -A INPUT -p 51 -m ah --ahspi 500 -j ACCEPT
```

--espspi

```
iptables -A INPUT -p 51 -m esp --espspi 500 -j ACCEPT
```

Das Kommando **iptables**(24)

Explizite Regelspezifikationen(2)

Conntrack-Optionen zur Auswertung des Zustands einer Verbindung(1)

-m conntrack

Zustände von Verbindungen für den Firewall:

NEW - neu, erstmaliges Auftreten eines Paketes für eine Verbindung

ESTABLISHED - Verbindung ist installiert (hin und zurück)

RELATED - Paket steht in Verbindung zu einer bestehenden Verbindung (z.B. ftp-Datenverbindungen)

INVALID - Paket kann nicht identifiziert werden – ist nichts. Solche Pakete sollten sofort weggeworfen werden!!!

cat /proc/net/ip_conntrack

Das Kommando **iptables**(25)

Explizite Regelspezifikationen(3)

Conntract-Optionen(2)

--ctstate *ctstate*

Es wird geprüft ob die Verbindung, die zu diesem Paket gehört, den Zustand *ctstate* hat. Werte für ctstate:

NEW, INVALID, ESTABLISHED, RELATED

Beispiel:

```
iptables -A INPUT -p tcp -m contrack --ctstate NEW -j ACCEPT
iptables -A INPUT -p tcp -m contrack ! --ctstate NEW -j ACCEPT
iptables -A INPUT -p tcp -m contrack --ctstat INVALID -j DROP
```

Das Kommando **iptables**(26)

Explizite Regelspezifikationen(4)

Conntract-Optionen(3)

--ctproto *proto*

Protokoll *proto* wird aktzeptiert. Identisch mit Option -p

Beispiel:

```
iptable -A INPUT -m conntrack --ctproto TCP -j ACCEPT
```

--ctorigsrc [!] *ipaddress*

Überprüfen gegenüber der Orginal-Source-IP-Adresse.

--ctorigdst [!] *ipaddress*

Überprüfen gegenüber der Orginal-Destination-IP-Adresse.

u.s.w.

Das Kommando **iptables**(27)
Explizite Regelspezifikationen(5)

IP-Range Optionen(1)

-m iprange

[!] **--src-range** *ipaddress1-ipaddress2*

Die Source-IP-Adresse des Paketes liegt im Bereich von *ipaddress1* und *ipaddress2*.

Beispiel:

```
iptables -A INPUT -p tcp -m iprange --src-range 141.20.20.0-
                                    141.20.39.255 -j ACCEPT
iptables -A INPUT -p tcp -m iprange --src-range ! 141.20.20.0-
                                    141.20.39.255 -j DROP
```

Das Kommando **iptables**(28)

Explizite Regelspezifikationen(6)

IP-Range Optionen(2)

-m iprange

[!] **--dst-range** *ipaddress1-ipaddress2*

Die Destination-IP-Adresse des Paketes liegt im Bereich von *ipaddress1* und *ipaddress2*.

Beispiel:

```
iptables -A INPUT -p tcp -m iprange --dst-range 141.20.20.0-
                                    141.20.39.255 -j ACCEPT
iptables -A INPUT -p tcp -m iprange --dst-range ! 141.20.20.0-
                                    141.20.39.255 -j DROP
```

Das Kommando **iptables**(29)
Explizite Regelspezifikationen(7)

Längen-Überprüfung

-m length

--length *length1:length2*
Länge des Paketes muss im Bereich *length1-length2* liegen.
Beispiel:
iptables -A INPUT -p tcp -m length --length 1490:1516 -j DROP

Das Kommando **iptables**(29)
Explizite Regelspezifikationen(7)

Begrenzung der Pakethäufigkeit

-m limit

--limit *rate*
Anzahl der Pakete pro Zeiteinheit. Folgende Angaben sind möglich:
*paketel***second**, *paketel***minute**, *paketel***hour**, *paketel***day**
Beispiel:
iptables -A INPUT -m limit --limit 10/second -j ACCEPT

--limit-burst *number*
Anzahl der gleichzeitigen Initial-Pakete – Standard ist 5.
iptables -A INPUT -m limit --limit-burst 10

Das Kommando **iptables**(30)
Explizite Regelspezifikationen(8)

Weitere:

-m mac --mac-source *mac-adresse*
MAC-Adressenüberprüfung
iptables -A INPUT -m mac --mac-source 00:0E:73:13:24:ab -j DROP

-m multiport --source-port *portnummern*
-m multiport --destination-port *portnummern*
-m multiport --port *portnummern*
Mehrere Ports gleich behandeln. Nur für TCP und UDP
iptables -A INPUT -p tcp -m multiport --port 22,53,80,110 -j ACCEPT

Das Kommando **iptables**(31)
Explizite Regelspezifikationen(9)

Weitere:

-m owner --uid-owner *uid*
-m owner --gid-owner *gid*
-m owner --pid-owner *pid*
-m owner --sid-owner *sid*

Überprüfen der Eigentumsverhältnisse des Prozesses zu dem das Paket gehört. Nur für lokal generierte Pakete zulässig (OUTPUT).

```
iptables -A OUTPUT -m owner --uid-owner 501 -j ACCEPT
```

-m pkttype --pkttype [**unicast** | **broadcast** | **multicast**]

Das Kommando **iptables**(32)
Explizite Regelspezifikationen(11)

Weitere:

-m state --state *state*

Verbindungszustand testen.

state kann folgende Werte annehmen:

NEW, ESTABLISHED, INVALID, RELATED

Beispiel:

```
iptables -A INPUT -m state --state NEW -j ACCEPT
```

-m tcpmss --mss *min-seg-size[:max-seg-size]*

Testet die Segmentlänge eines Paketes.

```
iptables -A INPUT -p tcp -m tcpmss --mss 100:2000
```

Das Kommando **iptables**(33)

Explizite Regelspezifikationen(10)

Merken von Paketen

-m recent

Verwalten von Paketadressen in Listen und Prüfen ob eine Adresse innerhalb einer vorgegeben Zeit in einer Liste eingetragen wurde.

Operationen(1)

--name *listen-name* - Name der Liste, sonst DEFAULT

--set - Erzeugen eines neuen Listeneintrags -merken

--rcheck - Prüfen ob Sourceadresse eingetragen ist

--update - Testen ob Sourceadresse vorhanden und Zeit updaten

--remove - testen ob Sourceadresse vorhanden ist und löschen der Adresse

Das Kommando **iptables**(34)

Explizite Regelspezifikationen(10)

Merken von Paketen

-m recent

Operationen(2)

--seconds *sekunden* - zusammen mit **--check** und **--update**
Anzahl der Sekunden

--hitcount *zahl* - zusammen mit **--check** und **--update**
Anzahl der Pakete

Beispiel

```
iptable -A FORWARD -m recent --name routed
        --seconds 60 --hitcount 100 -j DROP
```

- mindestens 100 Pakete in den letzten 60 Sekunden gesehen

Das Kommando **iptables**(35)
Explizite Regelspezifikationen(12)

Weitere:

-m tos --tos *value*

Testet das TOS-Feld (Type of Service).

Beispiel:

```
iptables -A INPUT -p tcp -m tos --tos 0x15
```

-m ttl --ttl *ttl-value*

Testet den TTL-Wert eines Paketes.

Beispiel:

```
iptables -A OUTPUT -m ttl --ttl 40
```

Das Kommando **iptables**(36)
Aktionen (target)(1)

Am Ende einer Regelspezifikation steht normalerweise eine Aktion (*target*), die ausgeführt wird, wenn alle Matches der Regelspezifikation erfüllt sind. Folgende Aktionen stehen zu Verfügung:

ACCEPT	**CLASSIFY**	**DNAT**
DROP	**DSCP**	**ECN**
LOG	**MARK**	**MASQUERADE**
MIRROR	**NETMAP**	**QUEUE**
REDIRECT	**REJECT**	**RETURN**
SAME	**SNAT**	**TCPMSS**
TOS	**TTL**	**ULOG**

Es kann jeweils eine Aktion durch „**-j** *aktionsname*“ am Ende eine Regelspezifikation angefügt werden. Die einzelnen Aktionen können weitere Parameter haben.

Das Kommando **iptables**(37)

Aktionen(2)

-j ACCEPT

ACCEPT bewirkt, dass keine weiteren Regeln der Kette der selben Tabelle abgearbeitet werden. D.h. dass das Paket noch von anderen Ketten nachgeordneter Tabellen durchaus untersucht werden kann und dadurch modifiziert oder anderwärtig behandelt werden kann (**DROP**).

-j DROP

DROP bewirkt, dass ein Paket weggeworfen wird. Keine weitere Regel kommt mehr zur Anwendung. Keinerlei Reaktion des Firewalls!!!!
Falls dies nicht gewünscht ist, **REJECT** benutzen.

Das Kommando **iptables**(38)

Aktionen(3)

-j REJECT [**--reject-with tcp-reset**]

REJECT bewirkt, dass ein Paket weggeworfen wird und die ICMP-Message *port-unreachable* an den Absender gesendet wird.
Für tcp-Pakete kann zusätzlich tcp-reset angegeben werden, das die Erzeugung eines TCP RST Pakets bewirkt und damit die tcp-Verbindung schließt. **REJECT** kann nur in den Chains: **INPUT**, **OUTPUT** und **FORWARD** benutzt werden.

Beispiel:

```
iptables -A FORWARD -p TCP --dport -j REJECT --reject-with tcp-reset
```

Das Kommando **iptables**(39)

Aktionen(4)

-j RETURN

RETURN bewirkt das Beenden einer Chain und die Rückkehr zu der eventuell vorhanden übergeordneten Chain. Ist keine übergeordnete Chain vorhanden, wird die Default-Aktion (**ACCEPT**, **DROP**) der aktuellen Chain benutzt(z.B. iptables -P chain ACCEPT).

Beispiel:

iptables -A INPUT -p TCP -j RETURN

Das Kommando **iptables**(40)

Aktionen(5)

Umleitungen(1)

-j REDIRECT --to-ports *portnummer*

Umleitung eines Ports auf den Port *portnummer*. Nur gültig für **PREROUTING** und **OUTPUT**.

Beispiel:

```
iptables -t nat -A PREROUTING -p TCP --dport 80 -j REDIRECT
                                        --to-ports 8080
```

Das Kommando **iptables**(41)

Aktionen(6)

Umleitungen(2)

-j DNAT --to-destination *ip-addresse*

Umsetzung der Zieladresse des aktuellen Paketes in *ip-addresse*.
Kann nur innerhalb von **PREROUTING** und **OUTPUT** benutzt werden.

Beispiel:

```
iptables -t nat -A PREROUTING -p tcp -d 141.20.20.50 --dport 80
                -j DNAT --to-destination 141.20.20.55-141.20.20.59
```

Das Kommando **iptables**(42)

Aktionen(7)

Umleitungen(3)

-j SNAT --to-source *ip-addresse*

Die Sourceadresse in dem Paket wird auf die Adresse *ip-addresse* umgesetzt. Dies ist nur für **POSTROUTING** zulässig.

Beispiel:

```
iptables -t nat -A POSTROUTING -p TCP -0 eth0
         -j SNAT --to-source 193.192.1.200
iptables -t nat -A POSTROUTING -p TCP -o eth0
         -j SNAT --to-source 194.192.1.100-194.192.1.10:1024-32000
```

Das Kommando **iptables**(43)

Aktionen(8)

Umleitungen(4)

-j MASQUERADE --to-ports *portnumbers*

Ähnlich wie SNAT. IP-Adresse kommt vom Netzwerkinterface. Ports können angegeben werden. Kann nur für **POSTROUTING** benutzt werden. Sinnvoll bei dynamischer Adresse des Firewall-Systems.

Beispiel:

```
iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE
        --to-ports 1024-32000
```

Das Kommando **iptables**(44)

Aktionen(9)

-j DSCP --set-dscp *number* --**set-dscp-class** *class*

Manipulieren des Differentiate Services Field in dem Paket.
Benutzbar bei **FORWARD**.

-j ECN --ecn-tcp-remove

Rücksetzen des ECN-Bits bei Ipv4-Headern.

-j MARK --set-mark *marke*

Setzen einer Marke (integer-wert). Nur bei **mangle** zulässig.

Das Kommando **iptables**(45)

Aktionen(10)

-j LOG

Aufzeichnen von Logging-Informationen des Firewalls über **syslogd**.

Weiter Optionen:

--log-level *debug-level* - festlegen des debug-Levels für syslogd

--log-prefix *prefix* - Prefix für die LOG-Nachrichten

--log-tcp-sequence - Logging der tcp-sequence Nummer

--log-tcp-options - logging der tcp-options

--log-ip-options - logging der ip-options

Beispiel:

```
iptables -A INPUT -p tcp -j LOG --log-tcp-sequence
iptables -A FORWARD -p tcp -j LOG --log-tcp-options
```

Das Kommando **iptables**(46)

Aktionen(11)

-j TCPMSS --set-mss *size*

Setzen der maximalen Segment Size auf *size*.

-j TOS –set-tos *value*

Setzen des TOS-Feldes auf den Wert *value*.

-j TTL

Modifikation des TTL-Feldes des Paketes. Folgende Möglichkeiten gibt es:

--ttl-set *value* - Setze den TTL-Wert auf *value*

--ttl-dec *value* - Verringere den TTL-Wert um (*value*+1)

--ttl-inc *value* - Erhöhe den TTL-Wert um (*value*-1), d.h. **--ttl-inc 1** bewirkt, daß der TTL-Wert nicht verändert wird.

Das Kommando **iptables**(47)

Aktionen(12)

-j NETMAP --to *IP-Addresse*

Neu!! 1:1 Umsetzung der Netzwerkadresse bei SNAT bzw. DNAT. Dabei wird der Hostpart der IP-Adresse unverändert übernommen und nur der Netzwerkteil wird übersetzt.

Beispiel:

```
iptables -t mangle -A PREROUTING -s 192.168.2.0/24
        -j NETMAP --to 141.20.39.0/24
```

-j SAME --to *ip-address-range*

Wie SNAT, aber genauer, versucht immer die gleiche Source-Adresse für eine Destination-Adresse zuzuordnen. Sinnvoll, wenn mehrere Source-Adressen zur Verfügung stehen.

```
-j SAME --to 141.20.30.20-141.20.30.30
```

iptables Kochrezepte(1):

iptables initialisieren:

```
modprobe ip_tables
modprobe ip_conntrack_ftp
```

Alles löschen:

```
iptables -F                    # Regeln löschen (--flush)
iptables -X    # eigene Chains löschen (--delete-chain)
```

Standardaktionen für Standard-Chains festlegen

```
iptables -P INPUT DROP
iptables -P OUTUT DROP
iptables -P FORWARD DROP
```

Jetzt ist der Rechner sicher. Nichts geht rein und nichts geht raus.

iptables Kochrezepte(2):

Loopback-Device erlauben (127.0.0.1 - Gerät lo)

```
iptables -A INPUT     -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

Regel zum Ablehnen einer Quelle

```
iptables -A INPUT -s 200.20.10.0/8 -j LOG --log-prefix "Spammer"
iptables -A INPUT -s 200.20.10.0/8 -j DROP
```

Stealth-Scans Ablehnen (falsche TCP-Pakete)

```
iptables -A INPUT -p tcp ! --syn -m state --state NEW -j LOG \
                --log-prefix "Stealth-Scan"
iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
```

iptables Kochrezepte(3):

INPUT-Regeln

Erlauben eingehender Pakete für eine zuvor erlaubte Verbindung

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED
                  -j ACCEPT
```

Erlauben von SSH-Verbindungen

```
iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT
```

Das Gleiche für FTP

```
iptables -A INPUT -p tcp --dport 21 -m state --state NEW -j ACCEPT
```

Erlauben von SSH-Verbindungen von einer speziellen Quelle

```
iptables -A INPUT -p tcp --dport 22 -s 141.20.20.0/24 \
          -m state --state NEW -j ACCEPT
```

iptables Kochrezepte(4):

OUTPUT-Regeln

Zulassen von Paketen für eine zuvor zugelassene Verbindung

```
iptables -I OUTPUT  -m state --state ESTABLISHED,RELATED \
         -j ACCEPT
```

Erlauben ausgehender Pings

```
iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
```

Erlauben ausgehende DNS-Abfragen

```
iptables -A OUTPUT -p udp --dport 53 -m state --state NEW \
         -j ACCEPT
```

iptables Kochrezepte(5):

Notmaßnahme bei passiven FTP-Verbindungen

```
iptables -A INPUT -p tcp --sport 1024: --dport 1024:  \
        -m state --state ESTABLISHED -j ACCEPT

iptables -A OUTPUT -p tcp --sport 1024: --dport 1024:  \
        -m state --state ESTABLISHED,RELATED -j ACCEPT
```

iptables Kochrezepte(6):

Einfaches NAT

Umleitung eines internen Netzes an eth0 auf eine PPP-Verbindung ppp0.

```
# laden des NAT-Modules
modprobe iptable_nat
# masquerading aktivieren
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
# Rechner zum Router machen
echo 1 >/proc/sys/net/ipv4/ip_forward
```

iptables Beispiel Suse-Firewall – 9.3 (1)

```
#  private Chains
iptables -N input_ext
iptables -N reject_func
# Belegung von INPUT
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j
   ACCEPT
iptables -A INPUT -i eth0 -j input_ext
iptables -A INPUT -i eth1 -j input_ext
iptables -A INPUT -m limit --limit 3/min -j LOG \
        --log-prefix "SFW2-IN-ILL-TARGET " \
        --log-tcp-options --log-ip-options
iptables -A INPUT -j DROP
```

iptables Beispiel Suse-Firewall – 9.3 (2)

```
# Belegung von FORWARD
iptables -A FORWARD -m limit --limit 3/min -j LOG \
        --log-prefix "SFW2-FWD-ILL-ROUTING " \
        --log-tcp-options --log-ip-options

# Belegung von OUTPUT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A OUTPUT -m state --state NEW,RELATED,ESTABLISHED \
            -j ACCEPT
iptables -A OUTPUT -m limit --limit 3/min -j LOG \
        --log-prefix "SFW2-OUT-ERROR " \
        --log-tcp-options --log-ip-options
```

iptables Beispiel Suse-Firewall – 9.3 (3)

```
# Belegung der input_ext chain(1)
iptables -A input_ext -m pkttype --pkt-type broadcast -j DROP
iptables -A input_ext -p icmp -m state --state RELATED,ESTABLISHED \
        -m icmp --icmp-type 0 -j ACCEPT
iptables -A input_ext -p icmp -m state --state RELATED,ESTABLISHED \
        -m icmp --icmp-type 3 -j ACCEPT
iptables -A input_ext -p icmp -m state --state RELATED,ESTABLISHED \
        -m icmp --icmp-type 11 -j ACCEPT
iptables -A input_ext -p icmp -m state --state RELATED,ESTABLISHED \
        -m icmp --icmp-type 12 -j ACCEPT
iptables -A input_ext -p icmp -m state --state RELATED,ESTABLISHED \
        -m icmp --icmp-type 14 -j ACCEPT
iptables -A input_ext -p icmp -m state --state RELATED,ESTABLISHED \
        -m icmp --icmp-type 18 -j ACCEPT
```

iptables Beispiel Suse-Firewall – 9.3 (4)

```
# Belegung der input_ext chain (2)
iptables -A input_ext -p icmp -m state \
          --state RELATED,ESTABLISHED \
           -m icmp --icmp-type 3/2 -j ACCEPT
iptables -A input_ext -p icmp -m state --state RELATED,ESTABLISHED \
          -m icmp --icmp-type 5 -j ACCEPT
iptables -A input_ext -p tcp -m limit --limit 3/min -m tcp --dport 22 \
          --tcp-flags FIN,SYN,RST,ACK SYN -j LOG \
          --log-prefix "SFW2-INext-ACC-TCP " \
          --log-tcp-options --log-ip-options
iptables -A input_ext -p tcp -m tcp --dport 22 -j ACCEPT
iptables -A input_ext -p tcp -m limit --limit 3/min -m tcp --dport 23 \
          --tcp-flags FIN,SYN,RST,ACK SYN -j LOG \
          --log-prefix "SFW2-INext-ACC-TCP " \
          --log-tcp-options --log-ip-options
```

iptables Beispiel Suse-Firewall – 9.3 (5)

```
# Belegung der input_ext chain(3)
iptables -A input_ext -p tcp -m tcp --dport 23 -j ACCEPT
iptables -A input_ext -p tcp -m tcp --dport 113 -m state \
        --state NEW -j reject_func
iptables -A input_ext -p tcp -m limit --limit 3/min -m tcp \
        --tcp-flags FIN,SYN,RST,ACK SYN -j LOG \
        --log-prefix "SFW2-INext-DROP-DEFLT " \
        --log-tcp-options --log-ip-options
iptables -A input_ext -p icmp -m limit --limit 3/min -j LOG \
        --log-prefix "SFW2-INext-DROP-DEFLT " \
        --log-tcp-options --log-ip-options
iptables -A input_ext -p udp -m limit --limit 3/min -j LOG \
       --log-prefix "SFW2-INext-DROP-DEFLT " \
       --log-tcp-options --log-ip-options
```

iptables Beispiel Suse-Firewall – 9.3 (6)

```
# Belegung der input_ext chain(4)
iptables -A input_ext -p udp -m limit --limit 3/min -j LOG \
        --log-prefix "SFW2-INext-DROP-DEFLT " \
        --log-tcp-options --log-ip-options
iptables -A input_ext -m limit --limit 3/min -m state \
        --state INVALID -j LOG \
        --log-prefix "SFW2-INext-DROP-DEFLT-INV " \
        --log-tcp-options --log-ip-options
iptables -A input_ext -j DROP
# Belegung der reject_ext chain
iptables -A reject_func -p tcp -j REJECT --reject-with tcp-reset
iptables -A reject_func -p udp -j REJECT--reject-with icmp-port-unreachable
iptables -A reject_func -j REJECT --reject-with icmp-proto-unreachable
```