

Ralf Spenneberg

Linux-Firewalls mit iptables & Co.

Sicherheit mit Kernel 2.4 und 2.6
für Linux-Server und -Netzwerke



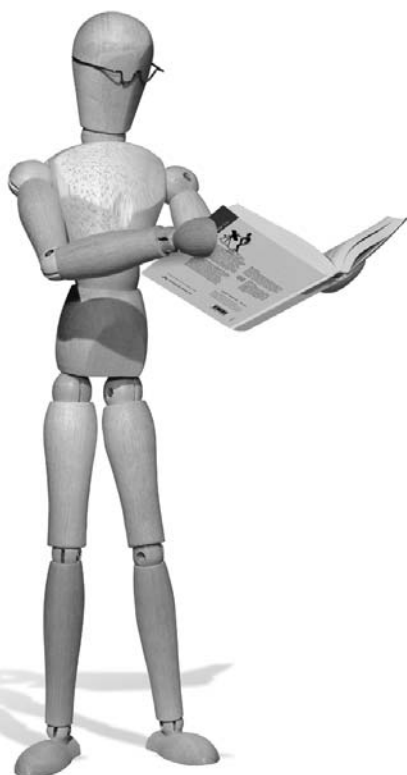
 ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Teil V

Fortgeschrittene Konfiguration





16 Die Iptables-Standardtests

Dieses Kapitel führt alle Paketprüfungen auf, die in dem Linux-Kernel 2.6.14 und Iptables enthalten sind. Alle weiteren Paketprüfungen, die über Patch-O-Matic zur Verfügung gestellt werden, werden im Kapitel 18 besprochen.

Im Folgenden werden die verschiedenen Tests alphabetisch aufgeführt.

16.1 Eingebaute Tests

Iptables verfügt über eingebaute Tests und Erweiterungen. Während die Erweiterungen über Kernelmodule realisiert werden und immer mit der Option `-m <extension>` in jeder Regel aktiviert werden müssen, stehen die eingebauten Tests immer zur Verfügung. Die folgenden eingebauten Tests können Sie immer verwenden.

16.1.1 `-p, -protocol`

Dieser Test prüft das in dem Paket verwendete IP-Protokoll. Hier können Sie den Namen des Protokolls (`tcp`, `udp`, etc.) oder die Protokollnummer verwenden. Die Definition der Protokollnamen erfolgt in der Datei `/etc/protocols`. Wenn Sie den Test negieren möchten, können Sie das Ausrufezeichen dem Protokoll voranstellen:
`--protocol ! tcp`.

16.1.2 `-s, -source`

Dieser Test prüft die Quell-IP-Adresse des Pakets. Dies kann eine IP-Adresse, ein Netzwerk oder ein DNS-Name sein. Bei der Angabe des Netzwerks können Sie die Netzmaske ausschreiben (`255.255.255.0`) oder die CIDR-Notation (`/24`) verwenden. Wenn Sie einen DNS-Namen verwenden, wird Iptables den DNS-Namen in dem Moment, in dem Sie die Regel hinzufügen, auflösen und anstelle des DNS-Namens die aufgelöste IP-Adresse nutzen. Dynamische DNS-Namen, deren IP-Adresse sich ändert, werden nicht unterstützt.

16.1.3 `-d, -destination`

Hiermit prüfen Sie die Ziel-IP-Adresse. Im Weiteren gelten die Beschränkungen wie bei der Quell-IP-Adresse.

16.1.4 -i, -in-interface

Hiermit können Sie prüfen, über welche Netzwerkkarte das Paket Ihren Rechner erreicht hat. Diesen Test können Sie nur in den `PREROUTING-`, `INPUT-` und `FORWARD-`Ketten benutzen. Die Verwendung in den `OUTPUT-` oder `POSTROUTING-`Ketten ist nicht erlaubt.

Die Netzwerkkarte muss in dem Moment, in dem Sie die Regel hinzufügen, noch nicht existieren. Wenn Sie mehrere Netzwerkkarten in einer Regel testen möchten, können Sie das `+` als Wildcard nutzen. Ein `eth+` trifft dann auf alle Ethernet-Netzwerkkarten zu.

16.1.5 -o, -out-interface

Hiermit können Sie prüfen, über welche Netzwerkkarte ein Paket den Rechner verlässt. Diesen Test dürfen Sie nur in den `FORWARD-`, `OUTPUT-` und `POSTROUTING-`Ketten verwenden. Ansonsten gelten die gleichen Einschränkungen wie oben.

16.1.6 -f, -fragment

Hiermit prüfen Sie, ob es sich bei einem fragmentierten Paket um das zweite oder ein weiteres Fragment handelt. Da Iptables ab dem zweiten Fragment keinen Zugriff mehr auf die Informationen des Transport-Protokolls hat, können Sie hiermit diese Fragmente herausfiltern und bei Bedarf zulassen.



Achtung

Da sich fragmentierte Pakete nur schlecht filtern lassen, sollten grundsätzlich alle Pakete vor der Filterung defragmentiert werden. Während Sie auf alten Linux-Kerneln dies speziell anschalten müssen, ist diese Funktion auf dem Linux-Kernel 2.4. und 2.6 automatisch aktiv, sobald Sie Connection Tracking nutzen. Sobald das Kernelmodul `ip_conntrack` geladen wurde, werden alle Pakete defragmentiert!

16.2 TCP-Tests

Wenn es sich beim Paket um ein TCP-Paket handelt und Sie dies mit der Option `-p tcp` in der Regel prüfen, können Sie weitere Prüfungen des TCP-Headers vornehmen. Dabei können Sie die Ports, die TCP-Flags und die TCP-Optionen prüfen.

16.2.1 sourceport

Mit der Option `--sourceport` (oder auch `--sport`) können Sie den Quellport prüfen. Hier können Sie einen Port oder einen Portbereich (`!t;port>:<port>`) angeben. Mehrere verschiedene Ports können Sie nicht angeben. Hierfür benötigen Sie den `multiport-`Test.

16.2.2 destinationport

Mit der Option `--destinationport` (oder auch `--dport`) können Sie den Zielport prüfen. Hier können Sie einen Port oder einen Portbereich (`{lt;port}<port>`) angeben. Mehrere verschiedene Ports können Sie nicht angeben. Hierfür benötigen Sie den `multiport`-Test.

16.2.3 tcp-flags

Mit dieser Option können Sie jedes TCP-Flag prüfen. Insgesamt gibt es sechs verschiedene: `SYN`, `ACK`, `FIN`, `RST`, `URG` und `PSH`. Dieser Test unterstützt auch noch `NONE` und `ALL`. Um die Flags zu testen, müssen Sie zunächst eine Maske mit den Flags definieren, die Sie prüfen möchten, und geben dann, durch ein Leerzeichen getrennt, die Flags an, die gesetzt sein müssen. Wenn Sie zum Beispiel prüfen möchten, ob das `SYN`-Flag gesetzt ist, aber das `RST`- und `ACK`-Flag nicht gesetzt sind, verwenden Sie: `--tcp-flags SYN,ACK,RST SYN`. Dieser Test betrachtet alle drei Flags (`SYN`, `ACK` und `RST`). Aber nur `SYN` darf gesetzt sein. Der Zustand der anderen Flags ist unerheblich.

Tipp



Das Netzwerk-Scan-Werkzeug `hping2` (<http://www.hping.org>) verwendet per Default TCP-Pakete, in denen kein einziges Flag gesetzt ist. Wenn Sie diese Pakete erkennen möchten, können Sie den folgenden Test nutzen:

```
$IPTABLES -A INPUT -p tcp --tcp-flags ALL NONE -j LOG \
--log-prefix "Hping2 Scan:"
```

16.2.4 syn

Dieser Test (`--syn`) ist eine Kurzform für `--tcp-flags SYN,ACK,RST SYN` und prüft, ob ein Paket das erste `SYN`-Paket in einer Verbindung ist. Wenn Sie diesen Test negieren möchten, setzen Sie ein Ausrufezeichen vor ihn.

16.2.5 tcp-options

Das TCP-Protokoll unterstützt verschiedene Optionen, die das Verhalten des TCP-Protokolls beeinflussen. Eine dieser Optionen ist zum Beispiel die Maximum Segment Size (MSS, siehe Abschnitt 16.31). Mit diesem Test können Sie prüfen, ob eine bestimmte Option gesetzt ist.

16.3 UDP-Tests

Das UDP-Protokoll verfügt nicht über so viele Informationen wie das TCP-Protokoll. Daher können Sie hier nur die Ports testen.

16.3.1 sourceport

Mit der Option `--sourceport` (oder auch `--sport`) können Sie den Quellport prüfen. Hier können Sie einen Port oder einen Portbereich (`!t:port>:<port>`) angeben. Mehrere verschiedene Ports können Sie nicht angeben. Hierfür benötigen Sie den `multiport`-Test.

16.3.2 destinationport

Mit der Option `--destinationport` (oder auch `--dport`) können Sie den Zielport prüfen. Hier können Sie einen Port oder einen Portbereich (`!t:port>:<port>`) angeben. Mehrere verschiedene Ports können Sie nicht angeben. Hierfür benötigen Sie den `multiport`-Test.

16.4 ICMP-Tests

Das ICMP-Protokoll besitzt keine Ports. Bei ICMP werden die verschiedenen Nachrichten durch Typ und Code unterschieden. Damit auch Sie prüfen können, um was für eine ICMP-Nachricht es sich handelt, unterstützt Iptables für ICMP-Pakete den Test `--icmp-type`. Die verschiedenen ICMP-Typen, die Sie testen können, zeigt der Iptables-Befehl an:

```
$ iptables -p icmp -h
...
ICMP v1.3.0 options:
  --icmp-type [!] typename          match icmp type
                                     (or numeric type or type/code)
```

```
Valid ICMP Types:
any
echo-reply (pong)
destination-unreachable
  network-unreachable
  host-unreachable
  protocol-unreachable
  port-unreachable
  fragmentation-needed
  source-route-failed
  network-unknown
  host-unknown
  network-prohibited
  host-prohibited
TOS-network-unreachable
TOS-host-unreachable
communication-prohibited
```

```

    host-precedence-violation
    precedence-cutoff
source-quench
redirect
    network-redirect
    host-redirect
    TOS-network-redirect
    TOS-host-redirect
echo-request (ping)
router-advertisement
router-solicitation
time-exceeded (ttl-exceeded)
    ttl-zero-during-transit
    ttl-zero-during-reassembly
parameter-problem
    ip-header-bad
    required-option-missing
timestamp-request
timestamp-reply
address-mask-request
address-mask-reply

```

Die Filterung des ICMP-Protokolls wird ausführlich im Kapitel 34 besprochen.

16.5 addrtype

Der Routing-Code des Linux-Kernels kategorisiert jede IP-Adresse im Netzwerk-stack. Dieser Test kann diese Kategorien testen und entsprechend Pakete verwerfen oder protokollieren. Die folgenden Kategorien werden von Linux unterstützt: UNSPEC, UNICAST, LOCAL, BROADCAST, ANYCAST, MULTICAST, BLACKHOLE, UNREACHABLE, PROHIBIT, THROW, NAT und XRESOLVE. Leider existiert kaum Dokumentation, in der die Kategorien erläutert werden.

Sie können sowohl die Quell-IP-Adresse als auch die Ziel-IP-Adresse testen:

```

-m addrtype --src-type <type>
-m addrtype --dst-type <type>

```

Ich kenne keine sinnvolle Verwendung für diesen Test.

16.6 ah

Mit diesem Test können Sie die Security-Parameter-Indices (SPI) des IPsec-AH-Protokolls testen. Da der Security-Parameter-Index üblicherweise dynamisch von dem IKE-Daemon ausgehandelt wird, ist eine statische Prüfung nicht besonders

sinnvoll, außer Sie verwenden manuell definierte IPsec-AH-Verbindungen (z.B. mit `setkey`).

```
-m ah --ahspi [!] <spi>[:<spi>]
```

16.7 comment

Dies ist kein wirklicher Test, da Sie hiermit keine Eigenschaft des Pakets prüfen können. Vielmehr können Sie einen Kommentar zu einer Regel hinzufügen. Damit können Sie nun die Regeln selbst dokumentieren, so dass bei einer späteren Auflistung der Regeln mit `iptables -vnl` die Kommentare angezeigt werden. Der Kommentar darf maximal 256 Zeichen lang sein.

```
-m comment --comment "Kommentar"
```

16.8 connbytes

Mit diesem Test können Sie prüfen, wie viele Pakete oder Bytes in einer Verbindung bereits übertragen wurden. So können Sie langlebige Downloads mit einer geringeren Priorität versehen, so dass sie den restlichen Verkehr nicht negativ beeinflussen. Sie können mit diesem Test die übertragenen Pakete oder Bytes in einer oder beiden Richtungen analysieren. Außerdem können Sie die durchschnittliche Paketgröße testen.

Dieser Test verfügt über drei zusätzliche Optionen:

- `[!] --connbytes <from>[:<to>]`: Hiermit prüfen Sie, ob die übertragenen Daten sich zwischen `from` und `to` befinden. Wenn Sie `to` nicht angeben, ist die obere Grenze nicht gesetzt.
- `--connbytes-dir [original|reply|both]`: Hiermit geben Sie die zu überwachenden Richtungen an.
- `--connbytes-mode [packets|bytes|avgpkt]`: Hiermit wählen Sie den Modus aus. Entweder zählt `Connbytes` die übertragenen Pakete oder Bytes, oder `Connbytes` ermittelt die durchschnittliche Paketgröße.

```
-m connbytes --connbytes 100000: --connbytes-dir both --connbytes-mode bytes
```

Tipp



Da die Zähler 64-Bit-Zähler sind, ist ein Überlauf der Zähler sehr unwahrscheinlich. Sie können hiermit sehr langlebige Verbindungen überwachen.

16.9 connmark

Verbindungen, die Sie in der NAT-Tabelle mit dem Target `CONNMARK` markiert haben, können Sie hiermit testen. Dazu können Sie entweder exakt die Markierung angeben oder auch zusätzlich noch eine Maske definieren. Diese Maske wird vor dem Test mit der Markierung der Verbindung Und-verknüpft.

```
-m connmark --mark <markierung>[<maske>]
```

16.10 conntrack

Dieses Modul gibt Ihnen erweiterten Zugriff auf die internen Werte der Connection Tracking-Tabelle. Hiermit können Sie diese Werte in Regeln ausnutzen. Sie können auf diese Weise prüfen, welchen Zustand (`INVALID RELATED`, `NEW`, `SNAT`, `DNAT`, `UNTRACKED`), welche originale und welche genattete Adresse die dazugehörige Verbindung in der Zustandstabelle aufweist.

Da mir keine sinnvolle Anwendung bekannt ist, verweise ich den interessierten Leser auf die `iptables(8)`-Manpage.

16.11 dccp

Das Datagram Congestion Control Protocol (DCCP) ist ein nachrichtenorientiertes Transport-Protokoll. Es implementiert wie TCP einen Datenfluss, aber bietet keine Übertragungssicherheit. DCCP befindet sich aktuell in der Entwicklung und wird meines Wissens noch nicht produktiv eingesetzt.

Mit diesem Test können Sie das DCCP-Protokoll testen. Der Test unterstützt die folgenden Optionen:

- `--source-port`: Prüft den Quellport.
- `--destination-port`: Prüft den Zielport.
- `--dccp-types`: DCCP unterstützt die folgenden Pakettypen: `REQUEST`, `RESPONSE`, `DATA`, `ACK`, `DATAACK`, `CLOSEREQ`, `CLOSE`, `RESET`, `SYNC`, `SYNACK` und `INVALID`.
- `--dccp-option`: Hiermit können Sie prüfen, ob eine numerische DCCP-Option gesetzt ist.

16.12 dscp

Das 6 Bit lange DiffServ-Code-Point-(DSCP-)Feld befindet sich in dem TOS-Feld des IP-Headers. DSCP hat TOS in den aktuellen IP-Protokoll-Standards abgelöst.

Sie können entweder den Wert numerisch oder durch Angabe der DSCP-Klasse testen:

```
-m dscp --dscp <nummer>
-m dscp --dscp-class <kategorie>
```

16.13 ecn

Mit diesem Test können Sie prüfen, ob die Explicit Congestion Notification-(ECN-) Bits in dem Paket gesetzt sind. Mit den folgenden Optionen können Sie die einzelnen Bits testen:

- `-m ecn --ecn-tcp-cwr`: Ist das Congestion-Window-Received-Bit gesetzt?
- `-m ecn --ecn-tcp-ecce`: Ist das ECN-Echo-Bit gesetzt?
- `-m ecn --ip-ect <nummer>`: Ist ein ECN-Capable-Transport-Bit gesetzt?

Weitere Informationen über ECN und die verwendeten Bits im IP- und TCP-Header finden Sie auf <http://www.icir.org/floyd/ecn.html>. Eine sinnvolle Anwendung ist mir unbekannt, da Sie die ECN-Bits entfernen können, ohne vorher ihr Vorhandensein zu prüfen.

16.14 esp

Hiermit können Sie ähnlich dem `ah`-Test die Security-Parameter-Indices des IPsec-ESP-Protokolls testen. Da auch diese dynamisch ausgehandelt werden, ist die Anwendung nicht sinnvoll.

```
-m esp --espspi [!] <spi>[:<spi>]
```

16.15 hashlimit

Dieser sehr interessante Test ähnelt dem `limit`-Test. Wenn Sie den `limit`-Test noch nicht kennen, lesen Sie erst dort nach, und kehren Sie anschließend hierher zurück.

Der `hashlimit`-Test ermöglicht Ihnen im Gegensatz zum allgemeinen `Limit`-Test die Definition von Grenzwerten in Abhängigkeit von der Ziel/Quell-IP-Adresse und des -Ports.

Der Test besitzt eigene Zähler für jede IP-Adresse und/oder jeden Port und nicht nur einen Zähler für die gesamte Regel.

Der Test unterstützt die folgenden Optionen:

- `--hashlimit <rate>`: siehe `limit`.
- `--hashlimit-burst <burst>`: siehe `limit`.
- `--hashlimit-mode destip|srcip|dstport|srcport`: Hashlimit-Modus.
- `--hashlimit-name <name>`: Dies erzeugt einen Eintrag in `/proc/net/ipt_hashlimit/<name>` für diese Regel.
- `--hashlimit-htable-size <num>`: Größe der Hash-Tabelle in Buckets.
- `--hashlimit-htable-max <max>`: Maximale Anzahl der Einträge in der Hash-Tabelle.

- `--hashlimithtable-gcinterval <intvl>`: Garbage Collector Interval (ms).
- `--hashlimithtable-expire <ttl>`: Lebensdauer (ms) der Einträge in der Hash-Tabelle.

Mit diesem Test lassen sich zum Beispiel die SSH-Brute-Force-Angriffe des letzten Jahres hervorragend abwehren:

```
$IPTABLES -A INPUT -m tcp -p tcp --dport 22 \
  -m hashlimit --hashlimit 1/min --hashlimit-mode srcip
  --hashlimit-name ssh -m state --state NEW -j ACCEPT
```

Diese Regel akzeptiert nur eine SSH-Verbindungsanfrage pro Minute und pro Quell-IP-Adresse. Allerdings kann natürlich ein Angreifer auch einen DoS ausführen, falls er die IP-Adresse erraten kann, von der aus Sie sich mit dem System verbinden möchten. Er spooft dann einfach Ihre IP-Adresse und führt häufige Verbindungsaufbauten durch!



Achtung

Die Manpage des Iptables-Kommandos hat hier in einigen Versionen einen Fehler und behauptet, dieser Test könnte nur die Ziel-IP-Adresse mit einer Beschränkung belegen. Testen Sie Ihren Kernel mit:

```
$ iptables -m hashlimit -h
hashlimit v1.3.0 options:
--hashlimit <avg>                max average match rate
                                   [Packets per second unless followed by
                                   /sec /minute /hour /day postfixes]
--hashlimit-mode <mode>          mode is a comma-separated list of
                                   dstip,srcip,dstport,srcport
--hashlimit-name <name>         name for /proc/net/ipt_hashlimit/
[--hashlimit-burst <num>]       number to match in a burst, default 5
[--hashlimithtable-size <num>] number of hashtable buckets
[--hashlimithtable-max <num>]  number of hashtable entries
[--hashlimithtable-gcinterval] interval between garbage collection runs
[--hashlimithtable-expire]     after which time are idle entries expired?
```

16.16 helper

Dieser Test prüft, ob ein Paket durch ein bestimmtes Conntrack-Helfermodul erkannt wurde. Hierzu geben Sie lediglich den Namen des Helfermoduls an. Um alle FTP-Pakete (Control- und Data-Connection) zu erkennen, können Sie folgenden Ausdruck verwenden: `-m helper --helper ftp`. Wenn Sie von den Default-Ports abgewichen sind, können Sie den Port mit angeben: `-m helper --helper ftp-2121`. Andere Helfermodule arbeiten genauso.

Damit können Sie zum Beispiel RELATED-Pakete nur akzeptieren, wenn es sich gleichzeitig um Pakete einer FTP-Verbindung handelt. ICMP-Fehlermeldungen oder IRC-Datenverbindungen werden von der folgenden Regel nicht akzeptiert:

```
$IPTABLES -A FORWARD -m state --state RELATED -m helper --helper ftp -j ACCEPT
```

16.17 iprange

In vielen Fällen möchte man prüfen, ob eine IP-Adresse sich innerhalb eines bestimmten Bereichs befindet. Jedoch kann in vielen Fällen nicht ein Netzwerk mit Subnetzmaske verwendet werden, da der Bereich nicht auf Netzwerkgrenzen beginnt und endet. Dann können Sie diesen Test verwenden. Sie können die Quell- und Ziel-IP-Adresse prüfen:

```
-m iprange [!] --src-range <ip>-<ip>  
-m iprange [!] --dst-range <ip>-<ip>
```

Wenn Ihr DHCP-Server zum Beispiel im Bereich von 192.168.0.100-150 dynamische IP-Adressen verteilt und Sie Regeln für die dynamischen Clients definieren möchten, können Sie folgenden Ausdruck verwenden: `-m iprange --src-range 192.168.0.100-192.168.0.150`.

16.18 length

Dieser Test prüft einfach die Länge eines Pakets. Sie können einen spezifischen Wert oder einen Bereich angeben:

```
-m length --length <länge>[:<länge>]
```

Damit ist es zum Beispiel möglich, ungewöhnlich große Ping-Pakete zu erkennen, die auf einen versteckten Kommunikationskanal hinweisen:

```
$IPTABLES -A FORWARD -p icmp --icmp-type echo-request -m length --length  
100:1500 -j LOG --log-prefix "Unusual Ping Size: "
```

16.19 limit

Dies ist der klassische Limit-Test des Iptables-Kommandos. Dieser Test ist in modernen Linux-Kerneln durch den `hashlimit`-Test im Wesentlichen abgelöst worden. Eine Regel, die diesen Test verwendet, trifft so lange zu, bis das Limit für die Regel erreicht wurde. Dabei besitzt die Regel nur einen einzigen Zähler, der unabhängig von den verwendeten IP-Adressen und -Ports die Ereignisse zählt. Hashlimit kann hier für jede IP-Adresse und jeden Port eigene Zähler verwalten. Dieser Test verfügt über zwei Optionen:

- `--limit <rate>`: Dies ist die maximale Paketrate. Sie können die Einheit in `/second`, `/minute`, `/hour` oder `/day` angeben. Default ist `3/hour`.

- `--limit-burst <rate>`: Hiermit können Sie einen initialen Schwellenwert definieren, der erst erreicht werden muss, bevor das Limit greift. Der Default-Wert ist 5.

Dieser Test eignet sich speziell für die Beschränkung der Protokollierung einzelner Pakete.

```
$IPTABLES -A FORWARD -p icmp --icmp-type echo-request \
-m limit --limit 1/minute -j LOG --log-prefix "Ping-Paket: "
```

16.20 mac

Mit diesem Test können Sie die Quell-MAC-Adresse testen. So können Sie sich vor ARP-Spoofing schützen, da Sie nur Pakete annehmen, die von einer bestimmten Quell-IP- und -MAC-Adresse stammen.

```
-m mac --mac-source <XX:XX:XX:XX:XX:XX>
```

Tipp



Wenn Sie diese Überprüfung für viele Systeme durchführen möchten, sollten Sie sich den `ipset`-Befehl mit der `macipmap` genauer ansehen (siehe Kapitel 25).

16.21 mark

Sie können in der Mangle-Tabelle Pakete mit dem `MARK`-Target markieren. Diese Firewall-Markierung können Sie mit diesem Test prüfen. Hierzu können Sie die Markierung genau angeben oder zusätzlich auch noch eine Maske definieren, die vor der Überprüfung mit der Markierung der Pakete Und-verknüpft wird.

```
-m mark --mark <markierung>[!<maske>]
```

Achtung



Die Paket-Markierung ist nur auf dem lokalen System gültig. Sobald das Paket das System verlässt, ist die Markierung verloren.

16.22 multiport

Diese Option erlaubt es Ihnen, mehrere Quell- oder Zielports in einer Regel anzugeben. Normalerweise erlaubt Iptables nur die Angabe eines Ports oder eines Port-Bereichs. Aktuelle Kernel ($\geq 2.6.11$) können bei dieser Option auch Port-Bereiche verwenden. Insgesamt dürfen Sie 15 Ports angeben. Ein Port-Bereich zählt als zwei Ports.

- `--source-ports [!] <port>[,<port>[,<port>:<port>]]`
- `--destination-ports [!] <port>[,<port>[,<port>:<port>]]`
- `--ports [!] <port>[,<port>[,<port>:<port>]]`

Mit diesem Test können Sie mehrere Regeln in einer zusammenfassen:

```
$IPTABLES -A FORWARD -p tcp -m multiport \  
  --destination-ports 21,22,25,80,443 -j ACCEPT
```

16.23 owner

Dieser Test kann nur in der OUTPUT-Kette angewendet werden, da der Linux-Kernel nur den Erzeuger lokaler Pakete ermitteln kann. Mit diesem Test können Sie prüfen, welcher Benutzer (`--uid-owner <uid>`), welche Gruppe (`--gid-owner <gid>`), welcher Prozess (`--pid-owner <pid>`), welche Sitzung (`--sid-owner <sid>`) oder welches Kommando (`--cmd-owner <cmd>`) ein Paket erzeugt hat. Leider funktionieren die letzten drei Funktionen aktuell nicht auf Mehrprozessorsystemen.

Damit können Sie aber dem Webserver verbieten, selbst Verbindungen aufzubauen:

```
$IPTABLES -A OUTPUT -m owner --cmd-owner httpd -j REJECT
```

16.24 physdev

Diese Option wird beim Linux-Kernel 2.6 verwendet, um die in einer Bridge verwendeten Netzwerkkarten zu erkennen und in einer Regel zu testen. Die genaue Verwendung dieses Tests wird daher in dem entsprechenden Kapitel erläutert und soll hier nicht wiederholt werden (siehe Kapitel 30).

16.25 pkttype

Dieser Test prüft den Pakettyp der Schicht 2. Sie können damit prüfen, ob es sich um ein unicast-, broadcast- oder ein multicast-Paket handelt.

Mit diesem Test können Sie alle lokalen Broadcast-Pakete einfach auf Ihrer Firewall verwerfen.

```
$IPTABLES -A INPUT -m pkttype --pkt-type broadcast -j DROP
```

16.26 realm

Dynamische Routing-Protokolle wie das Border-Gateway-Protokoll (BGP) verwenden Routing Realms für die Konfiguration. Dieser Test kann prüfen, ob das Paket zu einem bestimmten Realm gehört. Wenn Sie auf Ihrem System kein BGP einsetzen, benötigen Sie diesen Test nicht.

```
-m realm --realm [!] <realm>[<maske>]
```

16.27 recent

Dieser Test erlaubt es Ihnen, dynamisch eine Liste der gerade aktiven Quell-IP-Adressen zu erzeugen und in Ihren Regeln zu testen. Der Test hat die folgenden Optionen:

- `--name <name>`: Sie können verschiedene Listen für unterschiedliche Zwecke anlegen. Wenn Sie den Namen nicht angeben, wird die `DEFAULT`-Liste verwendet.
- `[!] --set`: Dies fügt die IP-Adresse zur Liste hinzu. Falls die IP-Adresse bereits in der Liste enthalten ist, wird der Eintrag aktualisiert. Dieser Test ist immer wahr, außer Sie geben das Ausrufezeichen an.
- `[!] --rcheck`: Dies prüft, ob die aktuelle IP-Adresse in der Liste enthalten ist.
- `[!] --update`: Identisch mit `--rcheck`, jedoch wird zusätzlich der Eintrag aktualisiert.
- `[!] --remove`: Hiermit entfernen Sie eine IP-Adresse aus der Liste. Falls die IP-Adresse nicht in der Liste vorhanden ist, trifft die Regel nicht zu.
- `[!] --seconds <sekunden>`: Diese Option können Sie gemeinsam mit `--rcheck` oder `--update` verwenden. Diese Optionen treffen dann nur zu, wenn die IP-Adresse innerhalb der angegebenen Zeitdauer gesehen wurde.
- `[!] --hitcount <treffer>`: Diese Option kann gemeinsam mit `--rcheck` und `--update` verwendet werden und prüft, ob die IP-Adresse bereits so häufig gesehen wurde. Mit `--seconds` können Sie zusätzlich den Zeitraum einschränken.
- `--rttl`: Hiermit prüfen Sie, ob der TTL-Wert des aktuellen Pakets identisch mit dem Paket ist, das mit `--set` hinzugefügt wurde.

Das Modul legt in `/proc/net/ipt_recent/<name>` Dateien an, die auch direkt gelesen und geschrieben werden können. Um eine IP-Adresse hinzuzufügen, können Sie diese in die Datei schreiben:

```
echo <ip> > /proc/net/ipt_recent/<name>
```

Genauso können Sie auch eine IP-Adresse entfernen:

```
echo -<ip> > /proc/net/ipt_recent/<name>
```

Um die gesamte Liste zu löschen, schreiben Sie das Wort `clear` in die entsprechende Datei.

Das Modul `ipt_recent` unterstützt beim Laden auch noch einige Optionen. Die Klammern geben die Default-Werte an:

- `ip_list_tot`: Anzahl der IP-Adressen pro Liste (100).
- `ip_pkt_list_tot`: Anzahl der Pakete pro IP-Adresse (20).
- `ip_list_hash_size`: Größe der Hash-Tabelle (0, Berechnung in Abhängigkeit von der `ip_list_tot`).
- `ip_list_perms`: Rechte der Dateien in `/proc/net/ipt_recent/*` (0644).
- `debug`: Debug-Informationen (0).

Um diese Optionen zu setzen, müssen Sie das Modul laden, bevor Sie in Ihren Regeln den Test verwenden:

```
# modprobe ipt_recent ip_list_tot=200
```

Dieses Modul können Sie nun einsetzen, um sich in Ihren Regeln spezifische IP-Adressen zu merken und zusätzliche Regeln auf diese IP-Adressen anzuwenden.

Um zum Beispiel Brute-Force-SSH-Angriffe abzuwehren, könnten Sie anstelle des `hashlimit`-Tests auch die folgenden beiden Regeln verwenden:

```
iptables -I INPUT -p tcp --dport 22 -m state --state NEW \  
-m recent --name SSH --set
```

```
iptables -I INPUT -p tcp --dport 22 -m state --state NEW \  
-m recent --name SSH --update --seconds 60 --hitcount 4 -j DROP
```

Die erste Regel fügt jede neue Quell-IP-Adresse, die eine SSH-Verbindung öffnet, der Liste `SSH` hinzu. Die zweite Regel prüft, ob von einer IP-Adresse innerhalb der letzten 60 Sekunden 4 Verbindungsanfragen erhalten wurden, und verwirft alle weiteren Anfragen. Sie können diese Regeln einsetzen, wenn der `hashlimit`-Test auf Ihrem Kernel nicht zur Verfügung steht.

16.28 sctp

Das Stream Control Transmission Protocol (SCTP, RFC 2960) ist ein Protokoll, das ähnlich wie TCP die Datenübertragung garantiert und verlorene oder beschädigte Pakete automatisch erneut sendet. Im Gegensatz zu TCP unterstützt SCTP mehrere Streams in einer Verbindung und Multihoming. Multihoming bedeutet, dass ein Kommunikationsendpunkt unter verschiedenen IP-Adressen erreicht werden kann.

SCTP wird aktuell von MPI-Parallel-Programmen und Telephonie-Software genutzt.

Mit diesem Test können Sie die Eigenschaften des Protokolls testen. Hierzu haben Sie die folgenden Optionen:

- `--source-port [!] <port>[:<port>]`

- `--destination-port [!] <port>[:<port>]`
- `--chunk-types [!] all|any|only <chunktype>[:<flags>]`

16.29 state

Dieser Test ermittelt den Zustand der Verbindung, zu der dieses Paket gehört. Die folgenden Zustände sind möglich:

- **NEW**: Das Paket gehört zu einer neuen Verbindung.
- **ESTABLISHED**: Das Paket gehört zu einer aufgebauten Verbindung.
- **RELATED**: Das Paket bezieht sich auf eine aufgebaute Verbindung (z.B. ICMP-Fehlernachricht).
- **INVALID**: Der Zustand des Pakets kann nicht ermittelt werden (z.B. ICMP-Fehlernachricht, die sich nicht auf eine bekannte Verbindung bezieht).

Wenn Ihr Kernel über die Raw-Tabelle verfügt, existiert zusätzlich noch der Zustand **UNTRACKED**. Der Zustand dieser Verbindung wird von dem Kernel nicht überwacht. Weitere Informationen finden Sie in Abschnitt 5.5 und in Kapitel 19.

16.30 string

Ab dem Kernel 2.6.14 ist der `string`-Test im Linux-Kernel fest verfügbar. Leider ist der entsprechende Code weder im aktuellen `iptables`-Kommando (1.3.3) noch im Patch-O-Matic verfügbar.

Ich gehe jedoch davon aus, dass zukünftige Versionen diesen Code enthalten werden. Ich vermute, dass die folgende Syntax verwendet werden wird:

- `--from <offset>`: Suche ab diesem Offset im Paket.
- `--to <offset>`: Suche bis zu diesem Offset im Paket.
- `--algo <algorithmus>`: Der Kernel 2.6.14 unterstützt im Moment drei verschiedene Algorithmen: Boyer-Moore (BM), Knuth-Morris-Pratt (KMP) und Finite-State-Machine (FSM).
- `--string <muster>`: Suche diese Zeichenkette.

16.31 tcpmss

Hiermit können Sie prüfen, ob in dem Paket eine Maximum Segment Size (MSS) gesetzt wurde. Dieser Test ist nur bei TCP-Paketen erlaubt. Sie können exakt einen Wert oder einen Bereich prüfen.

```
-m tcpmss [!] --mss <wert>[:<wert>]
```

Dieser Test ist nicht besonders hilfreich. Ich empfehle, bei Problemen mit der MSS grundsätzlich in allen Paketen die MSS zu ändern. Hierzu können Sie das MSS-Target einsetzen (siehe Abschnitt 17.21).

16.32 tos

Dieser Test ermöglicht die Prüfung des Type-of-Service-Wertes eines Pakets. Meines Erachtens gibt es keine wirklich sinnvolle Anwendung.

```
-m tos --tos <tos>
```

Sie können die folgenden Werte testen:

```
$ iptables -m tos -h
TOS match v1.3.0 options:
[!] --tos value                Match Type of Service field from one of the
                                following numeric or descriptive values:
                                Minimize-Delay 16 (0x10)
                                Maximize-Throughput 8 (0x08)
                                Maximize-Reliability 4 (0x04)
                                Minimize-Cost 2 (0x02)
                                Normal-Service 0 (0x00)
```

16.33 ttl

Mit dem ttl-Test können Sie den TTL-Wert eines Pakets prüfen. Sie können den genauen Wert testen oder prüfen, ob der Wert größer oder kleiner einem Vergleichswert ist. Hiermit können Sie zum Beispiel Windows-Systeme im lokalen Netz von Linux-Systemen unterscheiden, da Windows-Systeme eine andere initiale TTL verwenden. Es gibt aber bessere Methoden (z.B. `osf` in Patch-O-Matic, siehe Abschnitt 18.3.8).

```
-m ttl --ttl-eq <ttl> # gleich
-m ttl --ttl-gt <ttl> # größer
-m ttl --ttl-lt <ttl> # kleiner
```