NAME

iptables-extensions — list of extensions in the standard iptables distribution

SYNOPSIS

ip6tables [**−m** *name* [*module-options*...]] [**−j** *target-name* [*target-options*...]

iptables [**−m** *name* [*module-options*...]] [**−j** *target-name* [*target-options*...]

MATCH EXTENSIONS

iptables can use extended packet matching modules with the **−m** or **−−match** options, followed by the matching module name; after these, various extra command line options become available, depending on the specific module. You can specify multiple extended match modules in one line, and you can use the **−h** or **−−help** options after the module has been specified to receive help specific to that module. The extended match modules are evaluated in the order they are specified in the rule.

If the **−p** or **−−protocol** was specified and if and only if an unknown option is encountered, iptables will try load a match module of the same name as the protocol, to try making the option available.

addrtype

This module matches packets based on their **address type.** Address types are used within the k ernel networking stack and categorize addresses into various groups. The exact definition of that group depends on the specific layer three protocol.

The following address types are possible:

UNSPEC
an unspecified address (i.e. 0.0.0.0)

UNICAST
an unicast address

LOCAL
a local address

BROADCAST
a broadcast address

ANYCAST
an anycast packet

MULTICAST
a multicast address

BLACKHOLE
a blackhole address

UNREACHABLE
an unreachable address

PROHIBIT
a prohibited address

THROW
FIXME

NAT FIXME

XRESOLVE

[**!**] **−−src−type** *type*
Matches if the source address is of given type

[**!**] **−−dst−type** *type*
Matches if the destination address is of given type

−−limit−iface−in
The address type checking can be limited to the interface the packet is coming in. This option is only valid in the **PREROUTING**, **INPUT** and **FORWARD** chains. It cannot be specified with the **−−limit−iface−out** option.

−−limit−iface−out
The address type checking can be limited to the interface the packet is going out. This option is only valid in the **POSTROUTING**, **OUTPUT** and **FORWARD** chains. It cannot be specified with the **−−limit−iface−in** option.

ah (IPv6-specific)
This module matches the parameters in Authentication header of IPsec packets.

[**!**] **−−ahspi** *spi*[**:***spi*]
Matches SPI.

[**!**] **−−ahlen** *length*
Total length of this header in octets.

−−ahres
Matches if the reserved field is filled with zero.

ah (IPv4-specific)
This module matches the SPIs in Authentication header of IPsec packets.

[**!**] **−−ahspi** *spi*[**:***spi*]

bpf
Match using Linux Socket Filter. Expects a path to an eBPF object or a cBPF program in decimal format.

−−object−pinned *path*
Pass a path to a pinned eBPF object.

Applications load eBPF programs into the kernel with the bpf() system call and BPF_PROG_LOAD command and can pin them in a virtual filesystem with BPF_OBJ_PIN. To use a pinned object in iptables, mount the bpf filesystem using

mount −t bpf bpf ${BPF_MOUNT}

then insert the filter in iptables by path:

iptables −A OUTPUT −m bpf −−object−pinned ${BPF_MOUNT}/{PINNED_PATH} −j ACCEPT

−−bytecode *code*
Pass the BPF byte code format as generated by the **nfbpf_compile** utility.

The code format is similar to the output of the tcpdump −ddd command: one line that stores the number of instructions, followed by one line for each instruction. Instruction lines follow the pattern 'u16 u8 u8 u32' in decimal notation. Fields encode the operation, jump offset if true, jump offset if false and generic multiuse field 'K'. Comments are not supported.

For example, to read only packets matching 'ip proto 6', insert the following, without the comments or trailing whitespace:

```
4            # number of instructions
48 0 0 9      # load byte  ip->proto
21 0 1 6      # jump equal IPPROTO_TCP
6 0 0 1       # return     pass (non-zero)
6 0 0 0       # return     fail (zero)
```

You can pass this filter to the bpf match with the following command:

iptables −A OUTPUT −m bpf −−bytecode '4,48 0 0 9,21 0 1 6,6 0 0 1,6 0 0 0' −j ACCEPT

Or instead, you can invoke the nfbpf_compile utility.

iptables −A OUTPUT −m bpf −−bytecode "‘nfbpf_compile RAW ’ip proto 6’‘" −j ACCEPT

Or use tcpdump -ddd. In that case, generate BPF targeting a device with the same data link type as the xtables match. Iptables passes packets from the network layer up, without mac layer. Select a device with data link type RAW, such as a tun device:

ip tuntap add tun0 mode tun
ip link set tun0 up
tcpdump -ddd -i tun0 ip proto 6

See tcpdump -L -i $dev for a list of known data link types for a given device.

You may want to learn more about BPF from FreeBSD’s bpf(4) manpage.

cgroup

[**!**] **−−path** *path*
Match cgroup2 membership.

Each socket is associated with the v2 cgroup of the creating process. This matches packets coming from or going to all sockets in the sub-hierarchy of the specified path. The path should be relative to the root of the cgroup2 hierarchy.

[**!**] **−−cgroup** *classid*
Match cgroup net_cls classid.

classid is the marker set through the cgroup net_cls controller. This option and −−path can’t be used together.

Example:

iptables −A OUTPUT −p tcp −−sport 80 −m cgroup ! −−path service/http-server −j DROP

iptables −A OUTPUT −p tcp −−sport 80 −m cgroup ! −−cgroup 1 −j DROP

IMPORTANT: when being used in the INPUT chain, the cgroup matcher is currently only of limited functionality, meaning it will only match on packets that are processed for local sockets through early socket demuxing. Therefore, general usage on the INPUT chain is not advised unless the implications are well understood.

Available since Linux 3.14.

cluster

Allows you to deploy gateway and back-end load-sharing clusters without the need of load-balancers.

This match requires that all the nodes see the same packets. Thus, the cluster match decides if this node has to handle a packet given the following options:

−−cluster−total−nodes *num*
Set number of total nodes in cluster.

[**!**] **−−cluster−local−node** *num*
Set the local node number ID.

[**!**] **−−cluster−local−nodemask** *mask*
Set the local node number ID mask. You can use this option instead of **−−cluster−local−node**.

−−cluster−hash−seed *value*
Set seed value of the Jenkins hash.

Example:

iptables −A PREROUTING −t mangle −i eth1 −m cluster −−cluster−total−nodes 2 −−cluster−local−node 1 −−cluster−hash−seed 0xdeadbeef −j MARK −−set-mark 0xffff

iptables −A PREROUTING −t mangle −i eth2 −m cluster −−cluster−total−nodes 2 −−cluster−local−node 1 −−cluster−hash−seed 0xdeadbeef −j MARK −−set−mark 0xffff

iptables −A PREROUTING −t mangle −i eth1 −m mark ! −−mark 0xffff −j DROP

iptables −A PREROUTING −t mangle −i eth2 −m mark ! −−mark 0xffff −j DROP

And the following commands to make all nodes see the same packets:

ip maddr add 01:00:5e:00:01:01 dev eth1

ip maddr add 01:00:5e:00:01:02 dev eth2

arptables −A OUTPUT −o eth1 −−h−length 6 −j mangle −−mangle-mac-s 01:00:5e:00:01:01

arptables −A INPUT −i eth1 −−h-length 6 −−destination-mac 01:00:5e:00:01:01 −j mangle −−mangle−mac−d 00:zz:yy:xx:5a:27

arptables −A OUTPUT −o eth2 −−h−length 6 −j mangle −−mangle−mac−s 01:00:5e:00:01:02

arptables −A INPUT −i eth2 −−h−length 6 −−destination−mac 01:00:5e:00:01:02 −j mangle −−mangle−mac−d 00:zz:yy:xx:5a:27

NOTE: the arptables commands above use mainstream syntax. If you are using arptables-jf included in some RedHat, CentOS and Fedora versions, you will hit syntax errors. Therefore, you'll have to adapt these to the arptables-jf syntax to get them working.

In the case of TCP connections, pickup facility has to be disabled to avoid marking TCP ACK packets coming in the reply direction as valid.

echo 0 > /proc/sys/net/netfilter/nf_conntrack_tcp_loose

comment

Allows you to add comments (up to 256 characters) to any rule.

−−comment *comment*

Example:

iptables −A INPUT −i eth1 −m comment −−comment "my local LAN"

connbytes

Match by how many bytes or packets a connection (or one of the two flows constituting the connection) has transferred so far, or by average bytes per packet.

The counters are 64-bit and are thus not expected to overflow ;)

The primary use is to detect long-lived downloads and mark them to be scheduled using a lower priority band in traffic control.

The transferred bytes per connection can also be viewed through 'conntrack −L' and accessed via ctnetlink.

NOTE that for connections which have no accounting information, the match will always return false. The "net.netfilter.nf_conntrack_acct" sysctl flag controls whether **new** connections will be byte/packet counted. Existing connection flows will not be gaining/losing a/the accounting structure when be sysctl flag is flipped.

[**!**] **−−connbytes** *from*[**:***to*]

match packets from a connection whose packets/bytes/average packet size is more than FROM and less than TO bytes/packets. if TO is omitted only FROM check is done. "!" is used to match packets not falling in the range.

−−connbytes−dir {**original**|**reply**|**both**}

which packets to consider

−−connbytes−mode {**packets**|**bytes**|**avgpkt**}

whether to check the amount of packets, number of bytes transferred or the average size (in bytes) of all packets received so far. Note that when "both" is used together with "avgpkt", and data is going (mainly) only in one direction (for example HTTP), the average packet size will be about half of the actual data packets.

Example:

iptables .. −m connbytes −−connbytes 10000:100000 −−connbytes−dir both −−connbytes−mode bytes ...

connlabel

Module matches or adds connlabels to a connection. connlabels are similar to connmarks, except labels are bit-based; i.e. all labels may be attached to a flow at the same time. Up to 128 unique labels are currently supported.

[**!**] **−−label name**

matches if label **name** has been set on a connection. Instead of a name (which will be translated to a number, see EXAMPLE below), a number may be used instead. Using a number always overrides connlabel.conf.

−−set if the label has not been set on the connection, set it. Note that setting a label can fail. This is because the kernel allocates the conntrack label storage area when the connection is created, and it only reserves the amount of memory required by the ruleset that exists at the time the connection is created. In this case, the match will fail (or succeed, in case **−−label** option was negated).

This match depends on libnetfilter_conntrack 1.0.4 or later. Label translation is done via the**/etc/xtables/connlabel.conf** configuration file.

Example:

0	eth0-in
1	eth0-out
2	ppp-in
3	ppp-out
4	bulk-traffic
5	interactive

connlimit

Allows you to restrict the number of parallel connections to a server per client IP address (or client address block).

−−connlimit−upto *n*

Match if the number of existing connections is below or equal *n*.

−−connlimit−above *n*

Match if the number of existing connections is above *n*.

−−connlimit−mask *prefix_length*

Group hosts using the prefix length. For IPv4, this must be a number between (including) 0 and 32. For IPv6, between 0 and 128. If not specified, the maximum prefix length for the applicable protocol is used.

−−connlimit−saddr

Apply the limit onto the source group. This is the default if −−connlimit−daddr is not specified.

−−connlimit−daddr

Apply the limit onto the destination group.

Examples:

allow 2 telnet connections per client host

iptables −A INPUT −p tcp −−syn −−dport 23 −m connlimit −−connlimit−above 2 −j REJECT

you can also match the other way around:

iptables −A INPUT −p tcp −−syn −−dport 23 −m connlimit −−connlimit−upto 2 −j ACCEPT

limit the number of parallel HTTP requests to 16 per class C sized source network (24 bit netmask)

iptables −p tcp −−syn −−dport 80 −m connlimit −−connlimit−above 16 −−connlimit−mask 24 −j REJECT

limit the number of parallel HTTP requests to 16 for the link local network
(ipv6) ip6tables −p tcp −−syn −−dport 80 −s fe80::/64 −m connlimit −−connlimit−above 16 −−connlimit−mask 64 −j REJECT

Limit the number of connections to a particular host:
ip6tables −p tcp −−syn −−dport 49152:65535 −d 2001:db8::1 −m connlimit −−connlimit-above 100 −j REJECT

connmark

This module matches the netfilter mark field associated with a connection (which can be set using the **CONNMARK** target below).

[**!**] **−−mark** *value*[/*mask*]
Matches packets in connections with the given mark value (if a mask is specified, this is logically ANDed with the mark before the comparison).

conntrack

This module, when combined with connection tracking, allows access to the connection tracking state for this packet/connection.

[**!**] **−−ctstate** *statelist*
statelist is a comma separated list of the connection states to match. Possible states are listed below.

[**!**] **−−ctproto** *l4proto*
Layer-4 protocol to match (by number or name)

[**!**] **−−ctorigsrc** *address*[/*mask*]

[**!**] **−−ctorigdst** *address*[/*mask*]

[**!**] **−−ctreplsrc** *address*[/*mask*]

[**!**] **−−ctrepldst** *address*[/*mask*]
Match against original/reply source/destination address

[**!**] **−−ctorigsrcport** *port*[**:***port*]

[**!**] **−−ctorigdstport** *port*[**:***port*]

[**!**] **−−ctreplsrcport** *port*[**:***port*]

[**!**] **−−ctrepldstport** *port*[**:***port*]
Match against original/reply source/destination port (TCP/UDP/etc.) or GRE key. Matching against port ranges is only supported in kernel versions above 2.6.38.

[**!**] **−−ctstatus** *statelist*
statuslist is a comma separated list of the connection statuses to match. Possible statuses are listed below.

[**!**] **−−ctexpire** *time*[**:***time*]
Match remaining lifetime in seconds against given value or range of values (inclusive)

−−ctdir {**ORIGINAL**|**REPLY**}
Match packets that are flowing in the specified direction. If this flag is not specified at all, matches packets in both directions.

States for **−−ctstate**:

INVALID
The packet is associated with no known connection.

NEW The packet has started a new connection or otherwise associated with a connection which has not seen packets in both directions.

ESTABLISHED
The packet is associated with a connection which has seen packets in both directions.

RELATED
The packet is starting a new connection, but is associated with an existing connection, such as an FTP data transfer or an ICMP error.

UNTRACKED
The packet is not tracked at all, which happens if you explicitly untrack it by using −j CT −−notrack in the raw table.

SNAT A virtual state, matching if the original source address differs from the reply destination.

DNAT A virtual state, matching if the original destination differs from the reply source.

Statuses for **−−ctstatus**:

NONE None of the below.

EXPECTED
This is an expected connection (i.e. a conntrack helper set it up).

SEEN_REPLY
Conntrack has seen packets in both directions.

ASSURED
Conntrack entry should never be early-expired.

CONFIRMED
Connection is confirmed: originating packet has left box.

cpu

[**!**] **−−cpu** *number*
Match cpu handling this packet. cpus are numbered from 0 to NR_CPUS-1 Can be used in combination with RPS (Remote Packet Steering) or multiqueue NICs to spread network traffic on different queues.

Example:

iptables −t nat −A PREROUTING −p tcp −−dport 80 −m cpu −−cpu 0 −j REDIRECT −−to−port 8080

iptables −t nat −A PREROUTING −p tcp −−dport 80 −m cpu −−cpu 1 −j REDIRECT −−to−port 8081

Available since Linux 2.6.36.

dccp

[**!**] **−−source−port**,**−−sport** *port*[**:***port*]

[**!**] **−−destination−port**,**−−dport** *port*[**:***port*]

[**!**] **−−dccp−types** *mask*
Match when the DCCP packet type is one of 'mask'. 'mask' is a comma-separated list of packet types. Packet types are: **REQUEST RESPONSE DATA ACK DATAACK CLOSEREQ CLOSE RESET SYNC SYNCACK INVALID**.

[**!**] **−−dccp−option** *number*
Match if DCCP option set.

devgroup

Match device group of a packets incoming/outgoing interface.

[**!**] **−−src−group** *name*
Match device group of incoming device

[**!**] **−−dst−group** *name*
Match device group of outgoing device

dscp

This module matches the 6 bit DSCP field within the TOS field in the IP header. DSCP has superseded TOS within the IETF.

[**!**] **−−dscp** *value*

Match against a numeric (decimal or hex) value [0-63].

[**!**] **−−dscp−class** *class*

Match the DiffServ class. This value may be any of the BE, EF, AFxx or CSx classes. It will then be converted into its according numeric value.

dst (IPv6-specific)

This module matches the parameters in Destination Options header

[**!**] **−−dst−len** *length*

Total length of this header in octets.

−−dst−opts *type*[**:***length*][**,***type*[**:***length*]...]

numeric type of option and the length of the option data in octets.

ecn

This allows you to match the ECN bits of the IPv4/IPv6 and TCP header. ECN is the Explicit Congestion Notification mechanism as specified in RFC3168

[**!**] **−−ecn−tcp−cwr**

This matches if the TCP ECN CWR (Congestion Window Received) bit is set.

[**!**] **−−ecn−tcp−ece**

This matches if the TCP ECN ECE (ECN Echo) bit is set.

[**!**] **−−ecn−ip−ect** *num*

This matches a particular IPv4/IPv6 ECT (ECN-Capable Transport). You have to specify a number between ‘0’ and ‘3’.

esp

This module matches the SPIs in ESP header of IPsec packets.

[**!**] **−−espspi** *spi*[**:***spi*]

eui64 (IPv6-specific)

This module matches the EUI-64 part of a stateless autoconfigured IPv6 address. It compares the EUI-64 derived from the source MAC address in Ethernet frame with the lower 64 bits of the IPv6 source address. But "Universal/Local" bit is not compared. This module doesn’t match other link layer frame, and is only valid in the **PREROUTING**, **INPUT** and **FORWARD** chains.

frag (IPv6-specific)

This module matches the parameters in Fragment header.

[**!**] **−−fragid** *id*[**:***id*]

Matches the given Identification or range of it.

[**!**] **−−fraglen** *length*

This option cannot be used with kernel version 2.6.10 or later. The length of Fragment header is static and this option doesn’t make sense.

−−fragres

Matches if the reserved fields are filled with zero.

−−fragfirst

Matches on the first fragment.

−−fragmore

Matches if there are more fragments.

−−fraglast
Matches if this is the last fragment.

hashlimit

hashlimit uses hash buckets to express a rate limiting match (like the **limit** match) for a group of connections using a **single** iptables rule. Grouping can be done per-hostgroup (source and/or destination address) and/or per-port. It gives you the ability to express "*N* packets per time quantum per group" or "*N* bytes per seconds" (see below for some examples).

A hash limit option (**−−hashlimit−upto**, **−−hashlimit−above**) and **−−hashlimit−name** are required.

−−hashlimit−upto *amount*[**/second**|**/minute**|**/hour**|**/day**]
Match if the rate is below or equal to *amount*/quantum. It is specified either as a number, with an optional time quantum suffix (the default is 3/hour), or as *amount*b/second (number of bytes per second).

−−hashlimit−above *amount*[**/second**|**/minute**|**/hour**|**/day**]
Match if the rate is above *amount*/quantum.

−−hashlimit−burst *amount*
Maximum initial number of packets to match: this number gets recharged by one every time the limit specified above is not reached, up to this number; the default is 5. When byte-based rate matching is requested, this option specifies the amount of bytes that can exceed the given rate. This option should be used with caution -- if the entry expires, the burst value is reset too.

−−hashlimit−mode {**srcip**|**srcport**|**dstip**|**dstport**}**,**...
A comma-separated list of objects to take into consideration. If no −−hashlimit−mode option is given, hashlimit acts like limit, but at the expensive of doing the hash housekeeping.

−−hashlimit−srcmask *prefix*
When −−hashlimit−mode srcip is used, all source addresses encountered will be grouped according to the given prefix length and the so-created subnet will be subject to hashlimit. *prefix* must be between (inclusive) 0 and 32. Note that −−hashlimit−srcmask 0 is basically doing the same thing as not specifying srcip for −−hashlimit−mode, but is technically more expensive.

−−hashlimit−dstmask *prefix*
Like −−hashlimit−srcmask, but for destination addresses.

−−hashlimit−name *foo*
The name for the /proc/net/ipt_hashlimit/foo entry.

−−hashlimit−htable−size *buckets*
The number of buckets of the hash table

−−hashlimit−htable−max *entries*
Maximum entries in the hash.

−−hashlimit−htable−expire *msec*
After how many milliseconds do hash entries expire.

−−hashlimit−htable−gcinterval *msec*
How many milliseconds between garbage collection intervals.

−−hashlimit−rate−match
Classify the flow instead of rate-limiting it. This acts like a true/false match on whether the rate is above/below a certain number

−−hashlimit−rate−interval *sec*
Can be used with −−hashlimit−rate−match to specify the interval at which the rate should be sampled

Examples:

matching on source host
"1000 packets per second for every host in 192.168.0.0/16" => −s 192.168.0.0/16 −−hashlimit−mode srcip −−hashlimit−upto 1000/sec

matching on source port
"100 packets per second for every service of 192.168.1.1" => −s 192.168.1.1 −−hashlimit−mode srcport −−hashlimit−upto 100/sec

matching on subnet
"10000 packets per minute for every /28 subnet (groups of 8 addresses) in 10.0.0.0/8" => −s 10.0.0.0/8 −−hashlimit−mask 28 −−hashlimit−upto 10000/min

matching bytes per second
"flows exceeding 512kbyte/s" => −−hashlimit-mode srcip,dstip,srcport,dstport −−hashlimit−above 512kb/s

matching bytes per second
"hosts that exceed 512kbyte/s, but permit up to 1Megabytes without matching" −−hashlimit-mode dstip −−hashlimit−above 512kb/s −−hashlimit-burst 1mb

hbh (IPv6-specific)
This module matches the parameters in Hop-by-Hop Options header

[**!**] **−−hbh−len** *length*
Total length of this header in octets.

−−hbh−opts *type*[**:***length*][**,***type*[**:***length*]...]
numeric type of option and the length of the option data in octets.

helper
This module matches packets related to a specific conntrack-helper.

[**!**] **−−helper** *string*
Matches packets related to the specified conntrack-helper.

string can be "ftp" for packets related to a ftp-session on default port. For other ports append −portnr to the value, ie. "ftp−2121".

Same rules apply for other conntrack-helpers.

hl (IPv6-specific)
This module matches the Hop Limit field in the IPv6 header.

[**!**] **−−hl−eq** *value*
Matches if Hop Limit equals *value*.

−−hl−lt *value*
Matches if Hop Limit is less than *value*.

−−hl−gt *value*
Matches if Hop Limit is greater than *value*.

icmp (IPv4-specific)
This extension can be used if ‘−−protocol icmp’ is specified. It provides the following option:

[**!**] **−−icmp−type** {*type*[/*code*]|*typename*}
This allows specification of the ICMP type, which can be a numeric ICMP type, type/code pair, or one of the ICMP type names shown by the command
iptables −p icmp −h

icmp6 (IPv6-specific)
This extension can be used if ‘−−protocol ipv6−icmp’ or ‘−−protocol icmpv6’ is specified. It provides the following option:

[**!**] **−−icmpv6−type** *type*[*/code*]|*typename*
This allows specification of the ICMPv6 type, which can be a numeric ICMPv6 *type*, *type* and *code*, or one of the ICMPv6 type names shown by the command
ip6tables −p ipv6−icmp −h

iprange
This matches on a given arbitrary range of IP addresses.

[**!**] **−−src−range** *from*[*−to*]
Match source IP in the specified range.

[**!**] **−−dst−range** *from*[*−to*]
Match destination IP in the specified range.

ipv6header (IPv6-specific)
This module matches IPv6 extension headers and/or upper layer header.

−−soft Matches if the packet includes **any** of the headers specified with **−−header**.

[**!**] **−−header** *header*[**,***header*...]
Matches the packet which EXACTLY includes all specified headers. The headers encapsulated with ESP header are out of scope. Possible *header* types can be:

hop|**hop−by−hop**
Hop-by-Hop Options header

dst Destination Options header

route Routing header

frag Fragment header

auth Authentication header

esp Encapsulating Security Payload header

none No Next header which matches 59 in the 'Next Header field' of IPv6 header or any IPv6 extension headers

prot which matches any upper layer protocol header. A protocol name from /etc/protocols and numeric value also allowed. The number 255 is equivalent to **prot**.

ipvs
Match IPVS connection properties.

[**!**] **−−ipvs**
packet belongs to an IPVS connection

Any of the following options implies −−ipvs (even negated)

[**!**] **−−vproto** *protocol*
VIP protocol to match; by number or name, e.g. "tcp"

[**!**] **−−vaddr** *address*[*/mask*]
VIP address to match

[**!**] **−−vport** *port*
VIP port to match; by number or name, e.g. "http"

−−vdir {**ORIGINAL**|**REPLY**}
flow direction of packet

[**!**] **−−vmethod** {**GATE**|**IPIP**|**MASQ**}
IPVS forwarding method used

[**!**] **−−vportctl** *port*
VIP port of the controlling connection to match, e.g. 21 for FTP

length
This module matches the length of the layer-3 payload (e.g. layer-4 packet) of a packet against a specific value or range of values.

[**!**] **−−length** *length*[**:***length*]

limit
This module matches at a limited rate using a token bucket filter. A rule using this extension will match until this limit is reached. It can be used in combination with the **LOG** target to give limited logging, for example.

xt_limit has no negation support - you will have to use −m hashlimit ! −−hashlimit *rate* in this case whilst omitting −−hashlimit−mode.

−−limit *rate*[**/second**|**/minute**|**/hour**|**/day**]
Maximum average matching rate: specified as a number, with an optional '/second', '/minute', '/hour', or '/day' suffix; the default is 3/hour.

−−limit−burst *number*
Maximum initial number of packets to match: this number gets recharged by one every time the limit specified above is not reached, up to this number; the default is 5.

mac
[**!**] **−−mac−source** *address*
Match source MAC address. It must be of the form XX:XX:XX:XX:XX:XX. Note that this only makes sense for packets coming from an Ethernet device and entering the **PREROUTING**, **FORWARD** or **INPUT** chains.

mark
This module matches the netfilter mark field associated with a packet (which can be set using the **MARK** target below).

[**!**] **−−mark** *value*[**/***mask*]
Matches packets with the given unsigned mark value (if a *mask* is specified, this is logically ANDed with the *mask* before the comparison).

mh (IPv6-specific)
This extension is loaded if '−−protocol ipv6−mh' or '−−protocol mh' is specified. It provides the following option:

[**!**] **−−mh−type** *type*[**:***type*]
This allows specification of the Mobility Header(MH) type, which can be a numeric MH *type*, *type* or one of the MH type names shown by the command
ip6tables −p mh −h

multiport
This module matches a set of source or destination ports. Up to 15 ports can be specified. A port range (port:port) counts as two ports. It can only be used in conjunction with one of the following protocols: **tcp**, **udp**, **udplite**, **dccp** and **sctp**.

[**!**] **−−source−ports**,**−−sports** *port*[**,***port*|**,***port***:***port*]...
Match if the source port is one of the given ports. The flag **−−sports** is a convenient alias for this option. Multiple ports or port ranges are separated using a comma, and a port range is specified using a colon. **53,1024:65535** would therefore match ports 53 and all from 1024 through 65535.

[**!**] **−−destination−ports**,**−−dports** *port*[**,***port*|**,***port***:***port*]...
Match if the destination port is one of the given ports. The flag **−−dports** is a convenient alias for this option.

[**!**] **−−ports** *port*[**,***port*|**,***port***:***port*]...
Match if either the source or destination ports are equal to one of the given ports.

nfacct

The nfacct match provides the extended accounting infrastructure for iptables. You have to use this match together with the standalone user-space utility **nfacct(8)**

The only option available for this match is the following:

−−nfacct−name *name*

This allows you to specify the existing object name that will be use for accounting the traffic that this rule-set is matching.

To use this extension, you have to create an accounting object:

nfacct add http−traffic

Then, you have to attach it to the accounting object via iptables:

iptables −I INPUT −p tcp −−sport 80 −m nfacct −−nfacct−name http−traffic

iptables −I OUTPUT −p tcp −−dport 80 −m nfacct −−nfacct−name http−traffic

Then, you can check for the amount of traffic that the rules match:

nfacct get http−traffic

{ pkts = 0000000000000000156, bytes = 0000000000000151786 } = http-traffic;

You can obtain **nfacct(8)** from http://www.netfilter.org or, alternatively, from the git.netfilter.org repository.

osf

The osf module does passive operating system fingerprinting. This module compares some data (Window Size, MSS, options and their order, TTL, DF, and others) from packets with the SYN bit set.

[**!**] **−−genre** *string*

Match an operating system genre by using a passive fingerprinting.

−−ttl *level*

Do additional TTL checks on the packet to determine the operating system. *le vel* can be one of the following values:

- 0 - True IP address and fingerprint TTL comparison. This generally works for LANs.
- 1 - Check if the IP header's TTL is less than the fingerprint one. Works for globally-routable addresses.
- 2 - Do not compare the TTL at all.

−−log *level*

Log determined genres into dmesg even if they do not match the desired one. *le vel* can be one of the following values:

- 0 - Log all matched or unknown signatures
- 1 - Log only the first one
- 2 - Log all known matched signatures

You may find something like this in syslog:

Windows [2000:SP3:Windows XP Pro SP1, 2000 SP3]: 11.22.33.55:4024 -> 11.22.33.44:139 hops=3 Linux [2.5-2.6:] : 1.2.3.4:42624 -> 1.2.3.5:22 hops=4

OS fingerprints are loadable using the **nfnl_osf** program. To load fingerprints from a file, use:

nfnl_osf −f /usr/share/xtables/pf.os

To remove them again,

nfnl_osf −f /usr/share/xtables/pf.os −d

The fingerprint database can be downloaded from http://www.openbsd.org/cgi-bin/cvsweb/src/etc/pf.os .

owner

This module attempts to match various characteristics of the packet creator, for locally generated packets. This match is only valid in the OUTPUT and POSTROUTING chains. Forwarded packets do not have any socket associated with them. Packets from kernel threads do have a socket, but usually no owner.

[**!**] **−−uid−owner** *username*

[**!**] **−−uid−owner** *userid*[**−***userid*]

Matches if the packet socket's file structure (if it has one) is owned by the given user. You may also specify a numerical UID, or an UID range.

[**!**] **−−gid−owner** *groupname*

[**!**] **−−gid−owner** *groupid*[**−***groupid*]

Matches if the packet socket's file structure is owned by the given group. You may also specify a numerical GID, or a GID range.

−−suppl−groups

Causes group(s) specified with **−−gid-owner** to be also checked in the supplementary groups of a process.

[**!**] **−−socket−exists**

Matches if the packet is associated with a socket.

physdev

This module matches on the bridge port input and output devices enslaved to a bridge device. This module is a part of the infrastructure that enables a transparent bridging IP firewall and is only useful for kernel versions above version 2.5.44.

[**!**] **−−physdev−in** *name*

Name of a bridge port via which a packet is received (only for packets entering the **INPUT**, **FORWARD** and **PREROUTING** chains). If the interface name ends in a "+", then any interface which begins with this name will match. If the packet didn't arrive through a bridge device, this packet won't match this option, unless '!' is used.

[**!**] **−−physdev−out** *name*

Name of a bridge port via which a packet is going to be sent (for bridged packets entering the **FORWARD** and **POSTROUTING** chains). If the interface name ends in a "+", then any interface which begins with this name will match.

[**!**] **−−physdev−is−in**

Matches if the packet has entered through a bridge interface.

[**!**] **−−physdev−is−out**

Matches if the packet will leave through a bridge interface.

[**!**] **−−physdev−is−bridged**

Matches if the packet is being bridged and therefore is not being routed. This is only useful in the FORWARD and POSTROUTING chains.

pkttype

This module matches the link-layer packet type.

[**!**] **−−pkt−type** {**unicast**|**broadcast**|**multicast**}

policy

This module matches the policy used by IPsec for handling a packet.

−−dir {**in**|**out**}

Used to select whether to match the policy used for decapsulation or the policy that will be used for encapsulation. **in** is valid in the **PREROUTING, INPUT and FORWARD** chains, **out** is valid in the **POSTROUTING, OUTPUT and FORWARD** chains.

−−pol {**none**|**ipsec**}
Matches if the packet is subject to IPsec processing. **−−pol none** cannot be combined with **−−strict**.

−−strict
Selects whether to match the exact policy or match if any rule of the policy matches the given policy.

For each policy element that is to be described, one can use one or more of the following options. When **−−strict** is in effect, at least one must be used per element.

[**!**] **−−reqid** *id*
Matches the reqid of the policy rule. The reqid can be specified with **setkey(8)** using **unique:id** as level.

[**!**] **−−spi** *spi*
Matches the SPI of the SA.

[**!**] **−−proto** {**ah**|**esp**|**ipcomp**}
Matches the encapsulation protocol.

[**!**] **−−mode** {**tunnel**|**transport**}
Matches the encapsulation mode.

[**!**] **−−tunnel−src** *addr*[/*mask*]
Matches the source end-point address of a tunnel mode SA. Only valid with **−−mode tunnel**.

[**!**] **−−tunnel−dst** *addr*[/*mask*]
Matches the destination end-point address of a tunnel mode SA. Only valid with **−−mode tunnel**.

−−next Start the next element in the policy specification. Can only be used with **−−strict**.

quota
Implements network quotas by decrementing a byte counter with each packet. The condition matches until the byte counter reaches zero. Behavior is reversed with negation (i.e. the condition does not match until the byte counter reaches zero).

[**!**] **−−quota** *bytes*
The quota in bytes.

rateest
The rate estimator can match on estimated rates as collected by the RATEEST target. It supports matching on absolute bps/pps values, comparing two rate estimators and matching on the difference between two rate estimators.

For a better understanding of the available options, these are all possible combinations:

- **rateest** *operator* **rateest-bps**
- **rateest** *operator* **rateest-pps**
- (**rateest** minus **rateest-bps1**) *operator* **rateest-bps2**
- (**rateest** minus **rateest-pps1**) *operator* **rateest-pps2**
- **rateest1** *operator* **rateest2 rateest-bps**(without rate!)
- **rateest1** *operator* **rateest2 rateest-pps**(without rate!)
- (**rateest1** minus **rateest-bps1**) *operator* (**rateest2** minus **rateest-bps2**)
- (**rateest1** minus **rateest-pps1**) *operator* (**rateest2** minus **rateest-pps2**)

−−rateest−delta
For each estimator (either absolute or relative mode), calculate the difference between the estimator-determined flow rate and the static value chosen with the BPS/PPS options. If the flow rate is higher than the specified BPS/PPS, 0 will be used instead of a negative value. In other words, "max(0, rateest#_rate - rateest#_bps)" is used.

[**!**] **−−rateest−lt**
Match if rate is less than given rate/estimator.

[**!**] **−−rateest−gt**
Match if rate is greater than given rate/estimator.

[**!**] **−−rateest−eq**
Match if rate is equal to given rate/estimator.

In the so-called "absolute mode", only one rate estimator is used and compared against a static value, while in "relative mode", two rate estimators are compared against another.

−−rateest *name*
Name of the one rate estimator for absolute mode.

−−rateest1 *name*

−−rateest2 *name*
The names of the two rate estimators for relative mode.

−−rateest−bps [*value*]

−−rateest−pps [*value*]

−−rateest−bps1 [*value*]

−−rateest−bps2 [*value*]

−−rateest−pps1 [*value*]

−−rateest−pps2 [*value*]
Compare the estimator(s) by bytes or packets per second, and compare against the chosen value. See the above bullet list for which option is to be used in which case. A unit suffix may be used - available ones are: bit, [kmgt]bit, [KMGT]ibit, Bps, [KMGT]Bps, [KMGT]iBps.

Example: This is what can be used to route outgoing data connections from an FTP server over two lines based on the available bandwidth at the time the data connection was started:

Estimate outgoing rates

iptables −t mangle −A POSTROUTING −o eth0 −j RATEEST −−rateest−name eth0 −−rateest−interval 250ms −−rateest−ewma 0.5s

iptables −t mangle −A POSTROUTING −o ppp0 −j RATEEST −−rateest−name ppp0 −−rateest−interval 250ms −−rateest−ewma 0.5s

Mark based on available bandwidth

iptables −t mangle −A balance −m conntrack −−ctstate NEW −m helper −−helper ftp −m rateest −−rateest−delta −−rateest1 eth0 −−rateest−bps1 2.5mbit −−rateest−gt −−rateest2 ppp0 −−rateest−bps2 2mbit −j CONNMARK −−set−mark 1

iptables −t mangle −A balance −m conntrack −−ctstate NEW −m helper −−helper ftp −m rateest −−rateest−delta −−rateest1 ppp0 −−rateest−bps1 2mbit −−rateest−gt −−rateest2 eth0 −−rateest−bps2 2.5mbit −j CONNMARK −−set−mark 2

iptables −t mangle −A balance −j CONNMARK −−restore−mark

realm (IPv4-specific)
This matches the routing realm. Routing realms are used in complex routing setups involving dynamic routing protocols like BGP.

[**!**] **−−realm** *value*[/*mask*]
Matches a given realm number (and optionally mask). If not a number, value can be a named realm from /etc/iproute2/rt_realms (mask can not be used in that case). Both value and mask are four byte unsigned integers and may be specified in decimal, hex (by prefixing with "0x") or octal (if a leading zero is given).

recent

Allows you to dynamically create a list of IP addresses and then match against that list in a few different ways.

For example, you can create a "badguy" list out of people attempting to connect to port 139 on your firewall and then DROP all future packets from them without considering them.

−−set, **−−rcheck**, **−−update** and **−−remove** are mutually exclusive.

−−name *name*
Specify the list to use for the commands. If no name is given then **DEFAULT** will be used.

[**!**] **−−set**
This will add the source address of the packet to the list. If the source address is already in the list, this will update the existing entry. This will always return success (or failure if **!** is passed in).

−−rsource
Match/save the source address of each packet in the recent list table. This is the default.

−−rdest
Match/save the destination address of each packet in the recent list table.

−−mask *netmask*
Netmask that will be applied to this recent list.

[**!**] **−−rcheck**
Check if the source address of the packet is currently in the list.

[**!**] **−−update**
Like **−−rcheck**, except it will update the "last seen" timestamp if it matches.

[**!**] **−−remove**
Check if the source address of the packet is currently in the list and if so that address will be removed from the list and the rule will return true. If the address is not found, false is returned.

−−seconds *seconds*
This option must be used in conjunction with one of **−−rcheck** or **−−update**. When used, this will narrow the match to only happen when the address is in the list and was seen within the last given number of seconds.

−−reap This option can only be used in conjunction with **−−seconds**. When used, this will cause entries older than the last given number of seconds to be purged.

−−hitcount *hits*
This option must be used in conjunction with one of **−−rcheck** or **−−update**. When used, this will narrow the match to only happen when the address is in the list and packets had been received greater than or equal to the given value. This option may be used along with **−−seconds** to create an even narrower match requiring a certain number of hits within a specific time frame. The maximum value for the hitcount parameter is given by the "ip_pkt_list_tot" parameter of the xt_recent kernel module. Exceeding this value on the command line will cause the rule to be rejected.

−−rttl This option may only be used in conjunction with one of **−−rcheck** or **−−update**. When used, this will narrow the match to only happen when the address is in the list and the TTL of the current packet matches that of the packet which hit the **−−set** rule. This may be useful if you have problems with people faking their source address in order to DoS you via this module by disallowing others access to your site by sending bogus packets to you.

Examples:

iptables −A FORWARD −m recent −−name badguy −−rcheck −−seconds 60 −j DROP

iptables −A FORWARD −p tcp −i eth0 −−dport 139 −m recent −−name badguy −−set −j DROP

/proc/net/xt_recent/* are the current lists of addresses and information about each entry of each list.

Each file in **/proc/net/xt_recent/** can be read from to see the current list or written two using the following

commands to modify the list:

echo +*addr* **>/proc/net/xt_recent/DEFAULT**
to add *addr* to the DEFAULT list

echo −*addr* **>/proc/net/xt_recent/DEFAULT**
to remove *addr* from the DEFAULT list

echo / >/proc/net/xt_recent/DEFAULT
to flush the DEFAULT list (remove all entries).

The module itself accepts parameters, defaults shown:

ip_list_tot=*100*
Number of addresses remembered per table.

ip_pkt_list_tot=*20*
Number of packets per address remembered.

ip_list_hash_size=*0*
Hash table size. 0 means to calculate it based on ip_list_tot, default: 512.

ip_list_perms=*0644*
Permissions for /proc/net/xt_recent/* files.

ip_list_uid=*0*
Numerical UID for ownership of /proc/net/xt_recent/* files.

ip_list_gid=*0*
Numerical GID for ownership of /proc/net/xt_recent/* files.

rpfilter

Performs a reverse path filter test on a packet. If a reply to the packet would be sent via the same interface that the packet arrived on, the packet will match. Note that, unlike the in-kernel rp_filter, packets protected by IPSec are not treated specially. Combine this match with the policy match if you want this. Also, packets arriving via the loopback interface are always permitted. This match can only be used in the PREROUTING chain of the raw or mangle table.

−−loose
Used to specify that the reverse path filter test should match even if the selected output device is not the expected one.

−−validmark
Also use the packets' nfmark value when performing the reverse path route lookup.

−−accept−local
This will permit packets arriving from the network with a source address that is also assigned to the local machine.

−−invert
This will invert the sense of the match. Instead of matching packets that passed the reverse path filter test, match those that have failed it.

Example to log and drop packets failing the reverse path filter test:

iptables −t raw −N RPFILTER

iptables −t raw −A RPFILTER −m rpfilter −j RETURN

iptables −t raw −A RPFILTER −m limit −−limit 10/minute −j NFLOG −−nflog−prefix "rpfilter drop"

iptables −t raw −A RPFILTER −j DROP

iptables −t raw −A PREROUTING −j RPFILTER

Example to drop failed packets, without logging:

iptables −t raw −A RPFILTER −m rpfilter −−invert −j DROP

rt (IPv6-specific)

Match on IPv6 routing header

[**!**] **−−rt−type** *type*
Match the type (numeric).

[**!**] **−−rt−segsleft** *num*[**:***num*]
Match the ‘segments left’ field (range).

[**!**] **−−rt−len** *length*
Match the length of this header.

−−rt−0−res
Match the reserved field, too (type=0)

−−rt−0−addrs *addr*[**,***addr*...]
Match type=0 addresses (list).

−−rt−0−not−strict
List of type=0 addresses is not a strict list.

sctp

This module matches Stream Control Transmission Protocol headers.

[**!**] **−−source−port**,**−−sport** *port*[**:***port*]

[**!**] **−−destination−port**,**−−dport** *port*[**:***port*]

[**!**] **−−chunk−types** {**all**|**any**|**only**} *chunktype*[**:***flags*] [...]
The flag letter in upper case indicates that the flag is to match if set, in the lower case indicates to match if unset.

Chunk types: DATA INIT INIT_ACK SACK HEARTBEAT HEARTBEAT_ACK ABORT SHUTDOWN SHUTDOWN_ACK ERROR COOKIE_ECHO COOKIE_ACK ECN_ECNE ECN_CWR SHUTDOWN_COMPLETE ASCONF ASCONF_ACK FORWARD_TSN

chunk type	available flags
DATA	I U B E i u b e
ABORT	T t
SHUTDOWN_COMPLETE	T t

(lowercase means flag should be "off", uppercase means "on")

Examples:

iptables −A INPUT −p sctp −−dport 80 −j DROP

iptables −A INPUT −p sctp −−chunk−types any DATA,INIT −j DROP

iptables −A INPUT −p sctp −−chunk−types any DATA:Be −j ACCEPT

set

This module matches IP sets which can be defined by ipset(8).

[**!**] **−−match−set** *setname flag*[**,***flag*]...
where flags are the comma separated list of **src** and/or **dst** specifications and there can be no more than six of them. Hence the command

iptables −A FORWARD −m set −−match−set test src,dst

will match packets, for which (if the set type is ipportmap) the source address and destination port pair can be found in the specified set. If the set type of the specified set is single dimension (for example ipmap), then the command will match packets for which the source address can be found in the specified set.

−−return−nomatch
If the **−−return−nomatch** option is specified and the set type supports the **nomatch** flag, then the matching is reversed: a match with an element flagged with **nomatch** returns **true**, while a match with a plain element returns **false**.

! −−update−counters
If the **−−update−counters** flag is negated, then the packet and byte counters of the matching element in the set won't be updated. Default the packet and byte counters are updated.

! −−update−subcounters
If the **−−update−subcounters** flag is negated, then the packet and byte counters of the matching element in the member set of a list type of set won't be updated. Default the packet and byte counters are updated.

[**!**] **−−packets−eq** *value*
If the packet is matched an element in the set, match only if the packet counter of the element matches the given value too.

−−packets−lt *value*
If the packet is matched an element in the set, match only if the packet counter of the element is less than the given value as well.

−−packets−gt *value*
If the packet is matched an element in the set, match only if the packet counter of the element is greater than the given value as well.

[**!**] **−−bytes−eq** *value*
If the packet is matched an element in the set, match only if the byte counter of the element matches the given value too.

−−bytes−lt *value*
If the packet is matched an element in the set, match only if the byte counter of the element is less than the given value as well.

−−bytes−gt *value*
If the packet is matched an element in the set, match only if the byte counter of the element is greater than the given value as well.

The packet and byte counters related options and flags are ignored when the set was defined without counter support.

The option **−−match−set** can be replaced by **−−set** if that does not clash with an option of other extensions.

Use of −m set requires that ipset kernel support is provided, which, for standard kernels, is the case since Linux 2.6.39.

socket
This matches if an open TCP/UDP socket can be found by doing a socket lookup on the packet. It matches if there is an established or non−zero bound listening socket (possibly with a non−local address). The lookup is performed using the **packet** tuple of TCP/UDP packets, or the original TCP/UDP header **embedded** in an ICMP/ICPMv6 error packet.

−−transparent
Ignore non-transparent sockets.

−−nowildcard
Do not ignore sockets bound to 'any' address. The socket match won't accept zero−bound listeners by default, since then local services could intercept traffic that would otherwise be forwarded. This option therefore has security implications when used to match traffic being forwarded to redirect such packets to local machine with policy routing. When using the socket match to implement fully transparent proxies bound to non−local addresses it is recommended to use the −−transparent option instead.

Example (assuming packets with mark 1 are delivered locally):

−t mangle −A PREROUTING −m socket −−transparent −j MARK −−set−mark 1

−−restore−skmark
Set the packet mark to the matching socket's mark. Can be combined with the **−−transparent** and **−−nowildcard** options to restrict the sockets to be matched when restoring the packet mark.

Example: An application opens 2 transparent (**IP_TRANSPARENT**) sockets and sets a mark on them with **SO_MARK** socket option. We can filter matching packets:

−t mangle −I PREROUTING −m socket −−transparent −−restore-skmark −j action

−t mangle −A action −m mark −−mark 10 −j action2

−t mangle −A action −m mark −−mark 11 −j action3

state
The "state" extension is a subset of the "conntrack" module. "state" allows access to the connection tracking state for this packet.

[**!**] **−−state** *state*
Where state is a comma separated list of the connection states to match. Only a subset of the states unterstood by "conntrack" are recognized: **INVALID**, **ESTABLISHED**, **NEW**, **RELATED** or **UNTRACKED**. For their description, see the "conntrack" heading in this manpage.

statistic
This module matches packets based on some statistic condition. It supports two distinct modes settable with the **−−mode** option.

Supported options:

−−mode *mode*
Set the matching mode of the matching rule, supported modes are **random** and **nth.**

[**!**] **−−probability** *p*
Set the probability for a packet to be randomly matched. It only works with the **random** mode. *p* must be within 0.0 and 1.0. The supported granularity is in 1/2147483648th increments.

[**!**] **−−every** *n*
Match one packet every nth packet. It works only with the **nth** mode (see also the **−−packet** option).

−−packet *p*
Set the initial counter value (0 <= p <= n−1, default 0) for the **nth** mode.

string
This module matches a given string by using some pattern matching strategy. It requires a linux kernel >= 2.6.14.

−−algo {**bm**|**kmp**}
Select the pattern matching strategy. (bm = Boyer-Moore, kmp = Knuth-Pratt-Morris)

−−from *offset*
Set the offset from which it starts looking for any matching. If not passed, default is 0.

−−to *offset*
Set the offset up to which should be scanned. That is, byte *offset*-1 (counting from 0) is the last one that is scanned. If not passed, default is the packet size.

[**!**] **−−string** *pattern*
Matches the given pattern.

[**!**] **−−hex−string** *pattern*
Matches the given pattern in hex notation.

−−icase
Ignore case when searching.

Examples:

```
# The string pattern can be used for simple text characters.
iptables −A INPUT −p tcp −−dport 80 −m string −−algo bm −−string 'GET /index.html' −j LOG

# The hex string pattern can be used for non-printable characters, like |0D 0A| or |0D0A|.
iptables −p udp −−dport 53 −m string −−algo bm −−from 40 −−to 57 −−hex−string '|03|www|09|netfilter|03|org|00|'
```

tcp

These extensions can be used if '−−protocol tcp' is specified. It provides the following options:

[**!**] **−−source−port**,**−−sport** *port*[**:***port*]
Source port or port range specification. This can either be a service name or a port number. An inclusive range can also be specified, using the format *first***:***last*. If the first port is omitted, "0" is assumed; if the last is omitted, "65535" is assumed. The flag **−−sport** is a convenient alias for this option.

[**!**] **−−destination−port**,**−−dport** *port*[**:***port*]
Destination port or port range specification. The flag **−−dport** is a convenient alias for this option.

[**!**] **−−tcp−flags** *mask comp*
Match when the TCP flags are as specified. The first argument *mask* is the flags which we should examine, written as a comma-separated list, and the second argument *comp* is a comma-separated list of flags which must be set. Flags are: **SYN ACK FIN RST URG PSH ALL NONE**. Hence the command
iptables −A FORWARD −p tcp −−tcp−flags SYN,ACK,FIN,RST SYN
will only match packets with the SYN flag set, and the ACK, FIN and RST flags unset.

[**!**] **−−syn**
Only match TCP packets with the SYN bit set and the ACK,RST and FIN bits cleared. Such packets are used to request TCP connection initiation; for example, blocking such packets coming in an interface will prevent incoming TCP connections, but outgoing TCP connections will be unaffected. It is equivalent to **−−tcp−flags SYN,RST,ACK,FIN SYN**. If the "!" flag precedes the "−−syn", the sense of the option is inverted.

[**!**] **−−tcp−option** *number*
Match if TCP option set.

tcpmss

This matches the TCP MSS (maximum segment size) field of the TCP header. You can only use this on TCP SYN or SYN/ACK packets, since the MSS is only negotiated during the TCP handshake at connection startup time.

[**!**] **−−mss** *value*[**:***value*]
Match a given TCP MSS value or range. If a range is given, the second *value* must be greater than or equal to the first *value*.

time

This matches if the packet arrival time/date is within a given range. All options are optional, but are ANDed when specified. All times are interpreted as UTC by default.

−−datestart *YYYY*[*−MM*[*−DD*[**T***hh*[**:***mm*[**:***ss*]]]]]

−−datestop *YYYY*[*−MM*[*−DD*[**T***hh*[**:***mm*[**:***ss*]]]]]

Only match during the given time, which must be in ISO 8601 "T" notation. The possible time range is 1970-01-01T00:00:00 to 2038-01-19T04:17:07.

If −−datestart or −−datestop are not specified, it will default to 1970-01-01 and 2038-01-19, respectively.

−−timestart *hh***:***mm*[**:***ss*]

−−timestop *hh***:***mm*[**:***ss*]

Only match during the given daytime. The possible time range is 00:00:00 to 23:59:59. Leading zeroes are allowed (e.g. "06:03") and correctly interpreted as base-10.

[**!**] **−−monthdays** *day*[**,***day*...]

Only match on the given days of the month. Possible values are **1** to **31**. Note that specifying **31** will of course not match on months which do not have a 31st day; the same goes for 28- or 29-day February.

[**!**] **−−weekdays** *day*[**,***day*...]

Only match on the given weekdays. Possible values are **Mon**, **Tue**, **Wed**, **Thu**, **Fri**, **Sat**, **Sun**, or values from **1** to **7**, respectively. You may also use two-character variants (**Mo**, **Tu**, etc.).

−−contiguous

When **−−timestop** is smaller than **−−timestart** value, match this as a single time period instead distinct intervals. See EXAMPLES.

−−kerneltz

Use the kernel timezone instead of UTC to determine whether a packet meets the time regulations.

About kernel timezones: Linux keeps the system time in UTC, and always does so. On boot, system time is initialized from a referential time source. Where this time source has no timezone information, such as the x86 CMOS RTC, UTC will be assumed. If the time source is however not in UTC, userspace should provide the correct system time and timezone to the kernel once it has the information.

Local time is a feature on top of the (timezone independent) system time. Each process has its own idea of local time, specified via the TZ environment variable. The kernel also has its own timezone offset variable. The TZ userspace environment variable specifies how the UTC-based system time is displayed, e.g. when you run date(1), or what you see on your desktop clock. The TZ string may resolve to different offsets at different dates, which is what enables the automatic time-jumping in userspace. when DST changes. The kernel's timezone offset variable is used when it has to convert between non-UTC sources, such as FAT filesystems, to UTC (since the latter is what the rest of the system uses).

The caveat with the kernel timezone is that Linux distributions may ignore to set the kernel timezone, and instead only set the system time. Even if a particular distribution does set the timezone at boot, it is usually does not keep the kernel timezone offset - which is what changes on DST - up to date. ntpd will not touch the kernel timezone, so running it will not resolve the issue. As such, one may encounter a timezone that is always +0000, or one that is wrong half of the time of the year. As such, **using −−kerneltz is highly discouraged.**

EXAMPLES. To match on weekends, use:

−m time −−weekdays Sa,Su

Or, to match (once) on a national holiday block:

−m time −−datestart 2007−12−24 −−datestop 2007−12−27

Since the stop time is actually inclusive, you would need the following stop time to not match the first second of the new day:

−m time −−datestart 2007−01−01T17:00 −−datestop 2007−01−01T23:59:59

During lunch hour:

−m time −−timestart 12:30 −−timestop 13:30

The fourth Friday in the month:

−m time −−weekdays Fr −−monthdays 22,23,24,25,26,27,28

(Note that this exploits a certain mathematical property. It is not possible to say "fourth Thursday OR fourth Friday" in one rule. It is possible with multiple rules, though.)

Matching across days might not do what is expected. For instance,

−m time −−weekdays Mo −−timestart 23:00 −−timestop 01:00 Will match Monday, for one hour from midnight to 1 a.m., and then again for another hour from 23:00 onwards. If this is unwanted, e.g. if you would like 'match for two hours from Montay 23:00 onwards' you need to also specify the −−contiguous option in the example above.

tos

This module matches the 8-bit Type of Service field in the IPv4 header (i.e. including the "Precedence" bits) or the (also 8-bit) Priority field in the IPv6 header.

[**!**] **−−tos** *value*[*/mask*]
Matches packets with the given TOS mark value. If a mask is specified, it is logically ANDed with the TOS mark before the comparison.

[**!**] **−−tos** *symbol*
You can specify a symbolic name when using the tos match for IPv4. The list of recognized TOS names can be obtained by calling iptables with **−m tos −h**. Note that this implies a mask of 0x3F, i.e. all but the ECN bits.

ttl (IPv4-specific)

This module matches the time to live field in the IP header.

[**!**] **−−ttl−eq** *ttl*
Matches the given TTL value.

−−ttl−gt *ttl*
Matches if TTL is greater than the given TTL value.

−−ttl−lt *ttl*
Matches if TTL is less than the given TTL value.

u32

U32 tests whether quantities of up to 4 bytes extracted from a packet have specified values. The specification of what to extract is general enough to find data at given offsets from tcp headers or payloads.

[**!**] **−−u32** *tests*
The argument amounts to a program in a small language described below.

tests := location "=" value | tests "&&" location "=" value

value := range | value "," range

range := number | number ":" number

a single number, *n*, is interpreted the same as *n:n*. *n:m* is interpreted as the range of numbers **>=n** and **<=m**.

location := number | location operator number

operator := "&" | "<<" | ">>" | "@"

The operators **&**, <<, >> and **&&** mean the same as in C. The **=** is really a set membership operator and the value syntax describes a set. The **@** operator is what allows moving to the next header and is described further below.

There are currently some artificial implementation limits on the size of the tests:

* no more than 10 of "=" (and 9 "**&&**"s) in the u32 argument
* no more than 10 ranges (and 9 commas) per value
* no more than 10 numbers (and 9 operators) per location

To describe the meaning of location, imagine the following machine that interprets it. There are three registers:

A is of type **char ***, initially the address of the IP header

B and C are unsigned 32 bit integers, initially zero

The instructions are:

number
B = number;

C = (*(A+B)<<24) + (*(A+B+1)<<16) + (*(A+B+2)<<8) + *(A+B+3)

&number
C = C & number

<< number
C = C << number

>> number
C = C >> number

@number
A = A + C; then do the instruction number

Any access of memory outside [skb−>data,skb−>end] causes the match to fail. Otherwise the result of the computation is the final value of C.

Whitespace is allowed but not required in the tests. However, the characters that do occur there are likely to require shell quoting, so it is a good idea to enclose the arguments in quotes.

Example:

match IP packets with total length >= 256

The IP header contains a total length field in bytes 2-3.

−−u32 "**0 & 0xFFFF = 0x100:0xFFFF**"

read bytes 0-3

AND that with 0xFFFF (giving bytes 2-3), and test whether that is in the range [0x100:0xFFFF]

Example: (more realistic, hence more complicated)

match ICMP packets with icmp type 0

First test that it is an ICMP packet, true iff byte 9 (protocol) = 1

−−u32 "**6 & 0xFF = 1 &&** ...

read bytes 6-9, use **&** to throw away bytes 6-8 and compare the result to 1. Next test that it is not a fragment. (If so, it might be part of such a packet but we cannot always tell.) N.B.: This test is generally needed if you want to match anything beyond the IP header. The last 6 bits of byte 6 and all of byte 7 are 0 iff this is a complete packet (not a fragment). Alternatively, you can allow first fragments by only testing the last 5 bits of byte 6.

... **4 & 0x3FFF = 0 &&** ...

Last test: the first byte past the IP header (the type) is 0. This is where we have to use the @syntax. The length of the IP header (IHL) in 32 bit words is stored in the right half of byte 0 of the IP header itself.

... **0 >> 22 & 0x3C @ 0 >> 24 = 0**"

The first 0 means read bytes 0-3, **>>22** means shift that 22 bits to the right. Shifting 24 bits would give the first byte, so only 22 bits is four times that plus a few more bits. **&3C** then eliminates the two extra bits on the right and the first four bits of the first byte. For instance, if IHL=5, then the IP header is 20 (4 x 5) bytes long. In this case, bytes 0-1 are (in binary) xxxx0101 yyzzzzzz, **>>22** gives the 10 bit value xxxx0101yy and **&3C** gives 010100. **@** means to use this number as a new offset into the packet, and read four bytes starting from there. This is the first 4 bytes of the ICMP payload, of which byte 0 is the ICMP type. Therefore, we simply shift the value 24 to the right to throw out all but the first byte and compare the result with 0.

Example:

TCP payload bytes 8-12 is any of 1, 2, 5 or 8

First we test that the packet is a tcp packet (similar to ICMP).

−−u32 "**6 & 0xFF = 6 &&** ...

Next, test that it is not a fragment (same as above).

... **0 >> 22 & 0x3C @ 12 >> 26 & 0x3C @ 8 = 1,2,5,8**"

0>>22&3C as above computes the number of bytes in the IP header. **@** makes this the new offset into the packet, which is the start of the TCP header. The length of the TCP header (again in 32 bit words) is the left half of byte 12 of the TCP header. The **12>>26&3C** computes this length in bytes (similar to the IP header before). "@" makes this the new offset, which is the start of the TCP payload. Finally, 8 reads bytes 8-12 of the payload and **=** checks whether the result is any of 1, 2, 5 or 8.

udp

These extensions can be used if '−−protocol udp' is specified. It provides the following options:

[**!**] **−−source−port**,**−−sport** *port*[**:***port*]
Source port or port range specification. See the description of the **−−source−port** option of the TCP extension for details.

[**!**] **−−destination−port**,**−−dport** *port*[**:***port*]
Destination port or port range specification. See the description of the **−−destination−port** option of the TCP extension for details.

TARGET EXTENSIONS

iptables can use extended target modules: the following are included in the standard distribution.

AUDIT

This target creates audit records for packets hitting the target. It can be used to record accepted, dropped, and rejected packets. See auditd(8) for additional details.

−−type {**accept**|**drop**|**reject**}
Set type of audit record. Starting with linux-4.12, this option has no effect on generated audit messages anymore. It is still accepted by iptables for compatibility reasons, but ignored.

Example:

iptables −N AUDIT_DROP

iptables −A AUDIT_DROP −j AUDIT

iptables −A AUDIT_DROP −j DROP

CHECKSUM

This target selectively works around broken/old applications. It can only be used in the mangle table.

−−checksum−fill
Compute and fill in the checksum in a packet that lacks a checksum. This is particularly useful, if you need to work around old applications such as dhcp clients, that do not work well with

checksum offloads, but don't want to disable checksum offload in your device.

CLASSIFY

This module allows you to set the skb−>priority value (and thus classify the packet into a specific CBQ class).

−−set−class *major*:*minor*

Set the major and minor class value. The values are always interpreted as hexadecimal even if no 0x prefix is given.

CLUSTERIP (IPv4-specific)

This module allows you to configure a simple cluster of nodes that share a certain IP and MAC address without an explicit load balancer in front of them. Connections are statically distributed between the nodes in this cluster.

Please note that CLUSTERIP target is considered deprecated in favour of cluster match which is more flexible and not limited to IPv4.

−−new Create a new ClusterIP. You always have to set this on the first rule for a given ClusterIP.

−−hashmode *mode*

Specify the hashing mode. Has to be one of **sourceip**, **sourceip−sourceport**, **sourceip−sourceport−destport**.

−−clustermac *mac*

Specify the ClusterIP MAC address. Has to be a link−layer multicast address

−−total−nodes *num*

Number of total nodes within this cluster.

−−local−node *num*

Local node number within this cluster.

−−hash−init *rnd*

Specify the random seed used for hash initialization.

CONNMARK

This module sets the netfilter mark value associated with a connection. The mark is 32 bits wide.

−−set−xmark *value*[/*mask*]

Zero out the bits given by *mask* and XOR *value* into the ctmark.

−−save−mark [**−−nfmask** *nfmask*] [**−−ctmask** *ctmask*]

Copy the packet mark (nfmark) to the connection mark (ctmark) using the given masks. The new nfmark value is determined as follows:

ctmark = (ctmark & ˜ctmask) ^ (nfmark & nfmask)

i.e. *ctmask* defines what bits to clear and *nfmask* what bits of the nfmark to XOR into the ctmark. *ctmask* and *nfmask* default to 0xFFFFFFFF.

−−restore−mark [**−−nfmask** *nfmask*] [**−−ctmask** *ctmask*]

Copy the connection mark (ctmark) to the packet mark (nfmark) using the given masks. The new ctmark value is determined as follows:

nfmark = (nfmark & ˜*nfmask*) ^ (ctmark & *ctmask*);

i.e. *nfmask* defines what bits to clear and *ctmask* what bits of the ctmark to XOR into the nfmark. *ctmask* and *nfmask* default to 0xFFFFFFFF.

−−restore−mark is only valid in the **mangle** table.

The following mnemonics are available for **−−set−xmark**:

−−and−mark *bits*

Binary AND the ctmark with *bits*. (Mnemonic for **−−set−xmark 0**/*invbits*, where *invbits* is the binary negation of *bits*.)

−−or−mark *bits*
Binary OR the ctmark with *bits*. (Mnemonic for **−−set−xmark** *bits*/*bits*.)

−−xor−mark *bits*
Binary XOR the ctmark with *bits*. (Mnemonic for **−−set−xmark** *bits*/**0**.)

−−set−mark *value*[/*mask*]
Set the connection mark. If a mask is specified then only those bits set in the mask are modified.

−−save−mark [**−−mask** *mask*]
Copy the nfmark to the ctmark. If a mask is specified, only those bits are copied.

−−restore−mark [**−−mask** *mask*]
Copy the ctmark to the nfmark. If a mask is specified, only those bits are copied. This is only valid in the **mangle** table.

CONNSECMARK
This module copies security markings from packets to connections (if unlabeled), and from connections back to packets (also only if unlabeled). Typically used in conjunction with SECMARK, it is valid in the **security** table (for backwards compatibility with older kernels, it is also valid in the **mangle** table).

−−save If the packet has a security marking, copy it to the connection if the connection is not marked.

−−restore
If the packet does not have a security marking, and the connection does, copy the security marking from the connection to the packet.

CT
The CT target sets parameters for a packet or its associated connection. The target attaches a "template" connection tracking entry to the packet, which is then used by the conntrack core when initializing a new ct entry. This target is thus only valid in the "raw" table.

−−notrack
Disables connection tracking for this packet.

−−helper *name*
Use the helper identified by *name* for the connection. This is more flexible than loading the conntrack helper modules with preset ports.

−−ctevents *event*[**,**...]
Only generate the specified conntrack events for this connection. Possible event types are: **new**, **related**, **destroy**, **reply**, **assured**, **protoinfo**, **helper**, **mark** (this refers to the ctmark, not nfmark), **natseqinfo**, **secmark** (ctsecmark).

−−expevents *event*[**,**...]
Only generate the specified expectation events for this connection. Possible event types are: **new**.

−−zone-orig {*id*|**mark**}
For traffic coming from ORIGINAL direction, assign this packet to zone *id* and only have lookups done in that zone. If **mark** is used instead of *id*, the zone is derived from the packet nfmark.

−−zone-reply {*id*|**mark**}
For traffic coming from REPLY direction, assign this packet to zone *id* and only have lookups done in that zone. If **mark** is used instead of *id*, the zone is derived from the packet nfmark.

−−zone {*id*|**mark**}
Assign this packet to zone *id* and only have lookups done in that zone. If **mark** is used instead of *id*, the zone is derived from the packet nfmark. By default, packets have zone 0. This option applies to both directions.

−−timeout *name*
Use the timeout policy identified by *name* for the connection. This is provides more flexible timeout policy definition than global timeout values available at

/proc/sys/net/netfilter/nf_conntrack_*_timeout_*.

DNAT

This target is only valid in the **nat** table, in the **PREROUTING** and **OUTPUT** chains, and user-defined chains which are only called from those chains. It specifies that the destination address of the packet should be modified (and all future packets in this connection will also be mangled), and rules should cease being examined. It takes the following options:

−−to−destination [*ipaddr*[*−ipaddr*]][**:***port*[*−port*]]

which can specify a single new destination IP address, an inclusive range of IP addresses. Optionally a port range, if the rule also specifies one of the following protocols: **tcp**, **udp**, **dccp** or **sctp**. If no port range is specified, then the destination port will never be modified. If no IP address is specified then only the destination port will be modified. In Kernels up to 2.6.10 you can add several −−to−destination options. For those kernels, if you specify more than one destination address, either via an address range or multiple −−to−destination options, a simple round-robin (one after another in cycle) load balancing takes place between these addresses. Later Kernels (>= 2.6.11-rc1) don't have the ability to NAT to multiple ranges anymore.

−−random

If option **−−random** is used then port mapping will be randomized (kernel >= 2.6.22).

−−persistent

Gives a client the same source-/destination-address for each connection. This supersedes the SAME target. Support for persistent mappings is available from 2.6.29-rc2.

IPv6 support available since Linux kernels >= 3.7.

DNPT (IPv6-specific)

Provides stateless destination IPv6-to-IPv6 Network Prefix Translation (as described by RFC 6296).

You have to use this target in the **mangle** table, not in the **nat** table. It takes the following options:

−−src−pfx [*prefix/length]*

Set source prefix that you want to translate and length

−−dst−pfx [*prefix/length]*

Set destination prefix that you want to use in the translation and length

You have to use the SNPT target to undo the translation. Example:

ip6tables −t mangle −I POSTROUTING −s fd00::/64 −o vboxnet0 −j SNPT −−src-pfx fd00::/64 −−dst-pfx 2001:e20:2000:40f::/64

ip6tables −t mangle −I PREROUTING −i wlan0 −d 2001:e20:2000:40f::/64 −j DNPT −−src-pfx 2001:e20:2000:40f::/64 −−dst-pfx fd00::/64

You may need to enable IPv6 neighbor proxy:

sysctl −w net.ipv6.conf.all.proxy_ndp=1

You also have to use the **NOTRACK** target to disable connection tracking for translated flows.

DSCP

This target alters the value of the DSCP bits within the TOS header of the IPv4 packet. As this manipulates a packet, it can only be used in the mangle table.

−−set−dscp *value*

Set the DSCP field to a numerical value (can be decimal or hex)

−−set−dscp−class *class*

Set the DSCP field to a DiffServ class.

ECN (IPv4-specific)

This target selectively works around known ECN blackholes. It can only be used in the mangle table.

−−ecn−tcp−remove
Remove all ECN bits from the TCP header. Of course, it can only be used in conjunction with **−p tcp**.

HL (IPv6-specific)
This is used to modify the Hop Limit field in IPv6 header. The Hop Limit field is similar to what is known as TTL value in IPv4. Setting or incrementing the Hop Limit field can potentially be very dangerous, so it should be avoided at any cost. This target is only valid in **mangle** table.

Don't ever set or increment the value on packets that leave your local network!

−−hl−set *value*
Set the Hop Limit to 'value'.

−−hl−dec *value*
Decrement the Hop Limit 'value' times.

−−hl−inc *value*
Increment the Hop Limit 'value' times.

HMARK
Like MARK, i.e. set the fwmark, but the mark is calculated from hashing packet selector at choice. You have also to specify the mark range and, optionally, the offset to start from. ICMP error messages are inspected and used to calculate the hashing.

Existing options are:

−−hmark−tuple tuple
Possible tuple members are: **src** meaning source address (IPv4, IPv6 address), **dst** meaning destination address (IPv4, IPv6 address), **sport** meaning source port (TCP, UDP, UDPlite, SCTP, DCCP), **dport** meaning destination port (TCP, UDP, UDPlite, SCTP, DCCP), **spi** meaning Security Parameter Index (AH, ESP), and **ct** meaning the usage of the conntrack tuple instead of the packet selectors.

−−hmark−mod *value (must be > 0)*
Modulus for hash calculation (to limit the range of possible marks)

−−hmark−offset *value*
Offset to start marks from.

For advanced usage, instead of using −−hmark−tuple, you can specify custom
prefixes and masks:

−−hmark−src−prefix *cidr*
The source address mask in CIDR notation.

−−hmark−dst−prefix *cidr*
The destination address mask in CIDR notation.

−−hmark−sport−mask *value*
A 16 bit source port mask in hexadecimal.

−−hmark−dport−mask *value*
A 16 bit destination port mask in hexadecimal.

−−hmark−spi−mask *value*
A 32 bit field with spi mask.

−−hmark−proto−mask *value*
An 8 bit field with layer 4 protocol number.

−−hmark−rnd *value*
A 32 bit random custom value to feed hash calculation.

Examples:

iptables −t mangle −A PREROUTING −m conntrack −−ctstate NEW
−j HMARK −−hmark-tuple ct,src,dst,proto −−hmark-offset 10000 −−hmark−mod 10 −−hmark−rnd 0xfeedcafe

iptables −t mangle −A PREROUTING −j HMARK −−hmark−offset 10000 −−hmark-tuple src,dst,proto −−hmark-mod 10 −−hmark−rnd 0xdeafbeef

IDLETIMER

This target can be used to identify when interfaces have been idle for a certain period of time. Timers are identified by labels and are created when a rule is set with a new label. The rules also take a timeout value (in seconds) as an option. If more than one rule uses the same timer label, the timer will be restarted whenever any of the rules get a hit. One entry for each timer is created in sysfs. This attribute contains the timer remaining for the timer to expire. The attributes are located under the xt_idletimer class:

/sys/class/xt_idletimer/timers/<label>

When the timer expires, the target module sends a sysfs notification to the userspace, which can then decide what to do (eg. disconnect to save power).

−−timeout *amount*
This is the time in seconds that will trigger the notification.

−−label *string*
This is a unique identifier for the timer. The maximum length for the label string is 27 characters.

LED

This creates an LED-trigger that can then be attached to system indicator lights, to blink or illuminate them when certain packets pass through the system. One example might be to light up an LED for a few minutes every time an SSH connection is made to the local machine. The following options control the trigger behavior:

−−led−trigger−id *name*
This is the name given to the LED trigger. The actual name of the trigger will be prefixed with "netfilter-".

−−led-delay *ms*
This indicates how long (in milliseconds) the LED should be left illuminated when a packet arrives before being switched off again. The default is 0 (blink as fast as possible.) The special value *inf* can be given to leave the LED on permanently once activated. (In this case the trigger will need to be manually detached and reattached to the LED device to switch it off again.)

−−led−always−blink
Always make the LED blink on packet arrival, even if the LED is already on. This allows notification of new packets even with long delay values (which otherwise would result in a silent prolonging of the delay time.)

Example:

Create an LED trigger for incoming SSH traffic:
iptables −A INPUT −p tcp −−dport 22 −j LED −−led−trigger−id ssh

Then attach the new trigger to an LED:
echo netfilter−ssh >/sys/class/leds/*ledname*/trigger

LOG

Turn on kernel logging of matching packets. When this option is set for a rule, the Linux kernel will print some information on all matching packets (like most IP/IPv6 header fields) via the kernel log (where it can be read with *dmesg(1)* or read in the syslog).

This is a "non-terminating target", i.e. rule traversal continues at the next rule. So if you want to LOG the packets you refuse, use two separate rules with the same matching criteria, first using target LOG then DROP (or REJECT).

−−log−level *level*
Level of logging, which can be (system-specific) numeric or a mnemonic. Possible values are (in decreasing order of priority): **emerg**, **alert**, **crit**, **error**, **warning**, **notice**, **info** or **debug**.

−−log−prefix *prefix*
Prefix log messages with the specified prefix; up to 29 letters long, and useful for distinguishing messages in the logs.

−−log−tcp−sequence
Log TCP sequence numbers. This is a security risk if the log is readable by users.

−−log−tcp−options
Log options from the TCP packet header.

−−log−ip−options
Log options from the IP/IPv6 packet header.

−−log−uid
Log the userid of the process which generated the packet.

MARK
This target is used to set the Netfilter mark value associated with the packet. It can, for example, be used in conjunction with routing based on fwmark (needs iproute2). If you plan on doing so, note that the mark needs to be set in either the PREROUTING or the OUTPUT chain of the mangle table to affect routing. The mark field is 32 bits wide.

−−set−xmark *value*[/*mask*]
Zeroes out the bits given by *mask* and XORs *value* into the packet mark ("nfmark"). If *mask* is omitted, 0xFFFFFFFF is assumed.

−−set−mark *value*[/*mask*]
Zeroes out the bits given by *mask* and ORs *value* into the packet mark. If *mask* is omitted, 0xFFFFFFFF is assumed.

The following mnemonics are available:

−−and−mark *bits*
Binary AND the nfmark with *bits*. (Mnemonic for **−−set−xmark 0/***invbits*, where *invbits* is the binary negation of *bits*.)

−−or−mark *bits*
Binary OR the nfmark with *bits*. (Mnemonic for **−−set−xmark** *bits*/*bits*.)

−−xor−mark *bits*
Binary XOR the nfmark with *bits*. (Mnemonic for **−−set−xmark** *bits*/**0**.)

MASQUERADE
This target is only valid in the **nat** table, in the **POSTROUTING** chain. It should only be used with dynamically assigned IP (dialup) connections: if you have a static IP address, you should use the SNAT target. Masquerading is equivalent to specifying a mapping to the IP address of the interface the packet is going out, but also has the effect that connections are *forgotten* when the interface goes down. This is the correct behavior when the next dialup is unlikely to have the same interface address (and hence any established connections are lost anyway).

−−to−ports *port*[**−***port*]
This specifies a range of source ports to use, overriding the default **SNAT** source port-selection heuristics (see above). This is only valid if the rule also specifies one of the following protocols: **tcp**, **udp**, **dccp** or **sctp**.

−−random
Randomize source port mapping If option **−−random** is used then port mapping will be randomized (kernel >= 2.6.21). Since kernel 5.0, **−−random** is identical to **−−random-fully**.

−−**random-fully**
Full randomize source port mapping If option −−**random-fully** is used then port mapping will be fully randomized (kernel >= 3.13).

IPv6 support available since Linux kernels >= 3.7.

NETMAP
This target allows you to statically map a whole network of addresses onto another network of addresses. It can only be used from rules in the **nat** table.

−−**to** *address*[/*mask*]
Network address to map to. The resulting address will be constructed in the following way: All 'one' bits in the mask are filled in from the new 'address'. All bits that are zero in the mask are filled in from the original address.

IPv6 support available since Linux kernels >= 3.7.

NFLOG
This target provides logging of matching packets. When this target is set for a rule, the Linux kernel will pass the packet to the loaded logging backend to log the packet. This is usually used in combination with nfnetlink_log as logging backend, which will multicast the packet through a *netlink* socket to the specified multicast group. One or more userspace processes may subscribe to the group to receive the packets. Like LOG, this is a non-terminating target, i.e. rule traversal continues at the next rule.

−−**nflog−group** *nlgroup*
The netlink group (0 - 2^16−1) to which packets are (only applicable for nfnetlink_log). The default value is 0.

−−**nflog−prefix** *prefix*
A prefix string to include in the log message, up to 64 characters long, useful for distinguishing messages in the logs.

−−**nflog−range** *size*
This option has never worked, use --nflog-size instead

−−**nflog−size** *size*
The number of bytes to be copied to userspace (only applicable for nfnetlink_log). nfnetlink_log instances may specify their own range, this option overrides it.

−−**nflog−threshold** *size*
Number of packets to queue inside the kernel before sending them to userspace (only applicable for nfnetlink_log). Higher values result in less overhead per packet, but increase delay until the packets reach userspace. The default value is 1.

NFQUEUE
This target passes the packet to userspace using the **nfnetlink_queue** handler. The packet is put into the queue identified by its 16-bit queue number. Userspace can inspect and modify the packet if desired. Userspace must then drop or reinject the packet into the kernel. Please see libnetfilter_queue for details. **nfnetlink_queue** was added in Linux 2.6.14. The **queue-balance** option was added in Linux 2.6.31, **queue-bypass** in 2.6.39.

−−**queue−num** *value*
This specifies the QUEUE number to use. Valid queue numbers are 0 to 65535. The default value is 0.

−−**queue−balance** *value*:*value*
This specifies a range of queues to use. Packets are then balanced across the given queues. This is useful for multicore systems: start multiple instances of the userspace program on queues x, x+1, .. x+n and use "−−queue−balance *x*:*x*+*n*". Packets belonging to the same connection are put into the same nfqueue.

−−queue−bypass
By default, if no userspace program is listening on an NFQUEUE, then all packets that are to be queued are dropped. When this option is used, the NFQUEUE rule behaves like ACCEPT instead, and the packet will move on to the next table.

−−queue−cpu-fanout
Available starting Linux kernel 3.10. When used together with **--queue-balance** this will use the CPU ID as an index to map packets to the queues. The idea is that you can improve performance if there's a queue per CPU. This requires **--queue-balance** to be specified.

NOTRACK
This extension disables connection tracking for all packets matching that rule. It is equivalent with −j CT −−notrack. Like CT, NOTRACK can only be used in the **raw** table.

RATEEST
The RATEEST target collects statistics, performs rate estimation calculation and saves the results for later evaluation using the **rateest** match.

−−rateest−name *name*
Count matched packets into the pool referred to by *name*, which is freely choosable.

−−rateest−interval *amount*{**s**|**ms**|**us**}
Rate measurement interval, in seconds, milliseconds or microseconds.

−−rateest−ewmalog *value*
Rate measurement averaging time constant.

REDIRECT
This target is only valid in the **nat** table, in the **PREROUTING** and **OUTPUT** chains, and user-defined chains which are only called from those chains. It redirects the packet to the machine itself by changing the destination IP to the primary address of the incoming interface (locally-generated packets are mapped to the localhost address, 127.0.0.1 for IPv4 and ::1 for IPv6, and packets arriving on interfaces that don't have an IP address configured are dropped).

−−to−ports *port*[−*port*]
This specifies a destination port or range of ports to use: without this, the destination port is never altered. This is only valid if the rule also specifies one of the following protocols: **tcp**, **udp**, **dccp** or **sctp**.

−−random
If option **−−random** is used then port mapping will be randomized (kernel >= 2.6.22).

IPv6 support available starting Linux kernels >= 3.7.

REJECT (IPv6-specific)
This is used to send back an error packet in response to the matched packet: otherwise it is equivalent to **DROP** so it is a terminating TARGET, ending rule traversal. This target is only valid in the **INPUT**, **FORWARD** and **OUTPUT** chains, and user-defined chains which are only called from those chains. The following option controls the nature of the error packet returned:

−−reject−with *type*
The type given can be **icmp6−no−route**, **no−route**, **icmp6−adm−prohibited**, **adm−prohibited**, **icmp6−addr−unreachable**, **addr−unreach**, or **icmp6−port−unreachable**, which return the appropriate ICMPv6 error message (**icmp6−port−unreachable** is the default). Finally, the option **tcp−reset** can be used on rules which only match the TCP protocol: this causes a TCP RST packet to be sent back. This is mainly useful for blocking *ident* (113/tcp) probes which frequently occur when sending mail to broken mail hosts (which won't accept your mail otherwise). **tcp−r eset** can only be used with kernel versions 2.6.14 or later.

Warning: You should not indiscriminately apply the REJECT target to packets whose connection state is classified as INVALID; instead, you should only DROP these.

Consider a source host transmitting a packet P, with P experiencing so much delay along its path that the

source host issues a retransmission, P_2, with P_2 being successful in reaching its destination and advancing the connection state normally. It is conceivable that the late-arriving P may be considered not to be associated with any connection tracking entry. Generating a reject response for a packet so classed would then terminate the healthy connection.

So, instead of:

-A INPUT ... -j REJECT

do consider using:

-A INPUT ... -m conntrack --ctstate INVALID -j DROP -A INPUT ... -j REJECT

REJECT (IPv4-specific)

This is used to send back an error packet in response to the matched packet: otherwise it is equivalent to **DROP** so it is a terminating TARGET, ending rule traversal. This target is only valid in the **INPUT**, **FORWARD** and **OUTPUT** chains, and user-defined chains which are only called from those chains. The following option controls the nature of the error packet returned:

−−reject−with *type*

The type given can be **icmp−net−unreachable**, **icmp−host−unreachable**, **icmp−port−unreachable**, **icmp−proto−unreachable**, **icmp−net−prohibited**, **icmp−host−prohibited**, or **icmp−admin−prohibited** (*), which return the appropriate ICMP error message (**icmp−port−unreachable** is the default). The option**tcp−r eset** can be used on rules which only match the TCP protocol: this causes a TCP RST packet to be sent back. This is mainly useful for blocking *ident* (113/tcp) probes which frequently occur when sending mail to broken mail hosts (which won't accept your mail otherwise).

(*) Using icmp−admin−prohibited with kernels that do not support it will result in a plain DROP instead of REJECT

Warning: You should not indiscriminately apply the REJECT target to packets whose connection state is classified as INVALID; instead, you should only DROP these.

Consider a source host transmitting a packet P, with P experiencing so much delay along its path that the source host issues a retransmission, P_2, with P_2 being successful in reaching its destination and advancing the connection state normally. It is conceivable that the late-arriving P may be considered not to be associated with any connection tracking entry. Generating a reject response for a packet so classed would then terminate the healthy connection.

So, instead of:

-A INPUT ... -j REJECT

do consider using:

-A INPUT ... -m conntrack --ctstate INVALID -j DROP -A INPUT ... -j REJECT

SECMARK

This is used to set the security mark value associated with the packet for use by security subsystems such as SELinux. It is valid in the **security** table (for backwards compatibility with older kernels, it is also valid in the **mangle** table). The mark is 32 bits wide.

−−selctx *security_context*

SET

This module adds and/or deletes entries from IP sets which can be defined by ipset(8).

−−add−set *setname flag*[**,***flag*...]

add the address(es)/port(s) of the packet to the set

−−del−set *setname flag*[**,***flag*...]

delete the address(es)/port(s) of the packet from the set

−−**map−set** *setname flag*[**,***flag*...]
[−−map−mark] [−−map−prio] [−−map−queue] map packet properties (firewall mark, tc priority, hardware queue)

where *flag*(s) are **src** and/or **dst** specifications and there can be no more than six of them.

−−**timeout** *value*
when adding an entry, the timeout value to use instead of the default one from the set definition

−−**exist** when adding an entry if it already exists, reset the timeout value to the specified one or to the default from the set definition

−−**map−set** *set−name*
the set-name should be created with --skbinfo option −−**map−mark** map firewall mark to packet by lookup of value in the set −−**map−prio** map traffic control priority to packet by lookup of value in the set −−**map−queue** map hardware NIC queue to packet by lookup of value in the set

The −−**map−set** option can be used from the mangle table only. The −−**map−prio** and −−**map−queue** flags can be used in the OUTPUT, FORWARD and POSTROUTING chains.

Use of −j SET requires that ipset kernel support is provided, which, for standard kernels, is the case since Linux 2.6.39.

SNAT
This target is only valid in the **nat** table, in the **POSTROUTING** and **INPUT** chains, and user-defined chains which are only called from those chains. It specifies that the source address of the packet should be modified (and all future packets in this connection will also be mangled), and rules should cease being examined. It takes the following options:

−−**to−source** [*ipaddr*[*−ipaddr*]][**:***port*[*−port*]]
which can specify a single new source IP address, an inclusive range of IP addresses. Optionally a port range, if the rule also specifies one of the following protocols: **tcp**, **udp**, **dccp** or **sctp**. If no port range is specified, then source ports below 512 will be mapped to other ports below 512: those between 512 and 1023 inclusive will be mapped to ports below 1024, and other ports will be mapped to 1024 or above. Where possible, no port alteration will occur. In Kernels up to 2.6.10, you can add several −−to−source options. For those kernels, if you specify more than one source address, either via an address range or multiple −−to−source options, a simple round-robin (one after another in cycle) takes place between these addresses. Later Kernels (>= 2.6.11-rc1) don't have the ability to NAT to multiple ranges anymore.

−−**random**
If option −−**random** is used then port mapping will be randomized through a hash-based algorithm (kernel >= 2.6.21).

−−**random-fully**
If option −−**random-fully** is used then port mapping will be fully randomized through a PRNG (kernel >= 3.14).

−−**persistent**
Gives a client the same source-/destination-address for each connection. This supersedes the SAME target. Support for persistent mappings is available from 2.6.29-rc2.

Kernels prior to 2.6.36-rc1 don't have the ability to **SNAT** in the **INPUT** chain.

IPv6 support available since Linux kernels >= 3.7.

SNPT (IPv6-specific)
Provides stateless source IPv6-to-IPv6 Network Prefix Translation (as described by RFC 6296).

You have to use this target in the **mangle** table, not in the **nat** table. It takes the following options:

−−**src−pfx** [*prefix/length]*
Set source prefix that you want to translate and length

−−dst−pfx [*prefix/length]*
Set destination prefix that you want to use in the translation and length

You have to use the DNPT target to undo the translation. Example:

ip6tables −t mangle −I POSTROUTING −s fd00::/64 −o vboxnet0 −j SNPT −−src-pfx fd00::/64 −−dst-pfx 2001:e20:2000:40f::/64

ip6tables −t mangle −I PREROUTING −i wlan0 −d 2001:e20:2000:40f::/64 −j DNPT −−src-pfx 2001:e20:2000:40f::/64 −−dst-pfx fd00::/64

You may need to enable IPv6 neighbor proxy:

sysctl −w net.ipv6.conf.all.proxy_ndp=1

You also have to use the **NOTRACK** target to disable connection tracking for translated flows.

SYNPROXY

This target will process TCP three-way-handshake parallel in netfilter context to protect either local or backend system. This target requires connection tracking because sequence numbers need to be translated. The kernels ability to absorb SYNFLOOD was greatly improved starting with Linux 4.4, so this target should not be needed anymore to protect Linux servers.

−−mss *maximum segment size*
Maximum segment size announced to clients. This must match the backend.

−−wscale *window scale*
Window scale announced to clients. This must match the backend.

−−sack−perm
Pass client selective acknowledgement option to backend (will be disabled if not present).

−−timestamps
Pass client timestamp option to backend (will be disabled if not present, also needed for selective acknowledgement and window scaling).

Example:

Determine tcp options used by backend, from an external system

```
tcpdump -pni eth0 -c 1 'tcp[tcpflags] == (tcp-syn|tcp-ack)'
  port 80 &
telnet 192.0.2.42 80
18:57:24.693307 IP 192.0.2.42.80 > 192.0.2.43.48757:
  Flags [S.], seq 360414582, ack 788841994, win 14480,
  options [mss 1460,sackOK,
  TS val 1409056151 ecr 9690221,
  nop,wscale 9],
  length 0
```

Switch tcp_loose mode off, so conntrack will mark out−of−flow packets as state INVALID.

echo 0 > /proc/sys/net/netfilter/nf_conntrack_tcp_loose

Make SYN packets untracked

iptables −t raw −A PREROUTING −i eth0 −p tcp −−dport 80 −−syn −j CT −−notrack

Catch UNTRACKED (SYN packets) and INVALID (3WHS ACK packets) states and send them to SYNPROXY. This rule will respond to SYN packets with SYN+ACK syncookies, create ESTABLISHED for valid client response (3WHS ACK packets) and drop incorrect cookies. Flags combinations not expected during 3WHS will not match and continue (e.g. SYN+FIN, SYN+ACK).

iptables −A INPUT −i eth0 −p tcp −−dport 80 −m state −−state UNTRACKED,INVALID −j SYNPROXY

−−sack−perm −−timestamp −−mss 1460 −−wscale 9

Drop invalid packets, this will be out−of−flow packets that were not matched by SYNPROXY.

iptables −A INPUT −i eth0 −p tcp −−dport 80 −m state −−state INVALID −j DROP

TCPMSS

This target alters the MSS value of TCP SYN packets, to control the maximum size for that connection (usually limiting it to your outgoing interface's MTU minus 40 for IPv4 or 60 for IPv6, respectively). Of course, it can only be used in conjunction with **−p tcp**.

This target is used to overcome criminally braindead ISPs or servers which block "ICMP Fragmentation Needed" or "ICMPv6 Packet Too Big" packets. The symptoms of this problem are that everything works fine from your Linux firewall/router, but machines behind it can never exchange large packets:

1. Web browsers connect, then hang with no data received.
2. Small mail works fine, but large emails hang.
3. ssh works fine, but scp hangs after initial handshaking.

Workaround: activate this option and add a rule to your firewall configuration like:

iptables −t mangle −A FORWARD −p tcp −−tcp−flags SYN,RST SYN
−j TCPMSS −−clamp−mss−to−pmtu

−−set−mss *value*

Explicitly sets MSS option to specified value. If the MSS of the packet is already lower than *value*, it will **not** be increased (from Linux 2.6.25 onwards) to avoid more problems with hosts relying on a proper MSS.

−−clamp−mss−to−pmtu

Automatically clamp MSS value to (path_MTU − 40 for IPv4; −60 for IPv6). This may not function as desired where asymmetric routes with differing path MTU exist — the kernel uses the path MTU which it would use to send packets from itself to the source and destination IP addresses. Prior to Linux 2.6.25, only the path MTU to the destination IP address was considered by this option; subsequent kernels also consider the path MTU to the source IP address.

These options are mutually exclusive.

TCPOPTSTRIP

This target will strip TCP options off a TCP packet. (It will actually replace them by NO-OPs.) As such, you will need to add the **−p tcp** parameters.

−−strip−options *option*[**,***option*...]

Strip the given option(s). The options may be specified by TCP option number or by symbolic name. The list of recognized options can be obtained by calling iptables with **−j TCPOPTSTRIP −h**.

TEE

The **TEE** target will clone a packet and redirect this clone to another machine on the **local** network segment. In other words, the nexthop must be the target, or you will have to configure the nexthop to forward it further if so desired.

−−gateway *ipaddr*

Send the cloned packet to the host reachable at the given IP address. Use of 0.0.0.0 (for IPv4 packets) or :: (IPv6) is invalid.

To forward all incoming traffic on eth0 to an Network Layer logging box:

−t mangle −A PREROUTING −i eth0 −j TEE −−gateway 2001:db8::1

TOS

This module sets the Type of Service field in the IPv4 header (including the "precedence" bits) or the Priority field in the IPv6 header. Note that TOS shares the same bits as DSCP and ECN. The TOS target is only valid in the **mangle** table.

−−**set−tos** *value*[/*mask*]
Zeroes out the bits given by *mask* (see NOTE below) and XORs *value* into the TOS/Priority field. If *mask* is omitted, 0xFF is assumed.

−−**set−tos** *symbol*
You can specify a symbolic name when using the TOS target for IPv4. It implies a mask of 0xFF (see NOTE below). The list of recognized TOS names can be obtained by calling iptables with **−j TOS −h**.

The following mnemonics are available:

−−**and−tos** *bits*
Binary AND the TOS value with *bits*. (Mnemonic for −−**set−tos 0**/*invbits*, where *invbits* is the binary negation of *bits*. See NOTE below.)

−−**or−tos** *bits*
Binary OR the TOS value with *bits*. (Mnemonic for −−**set−tos** *bits*/*bits*. See NOTE below.)

−−**xor−tos** *bits*
Binary XOR the TOS value with *bits*. (Mnemonic for −−**set−tos** *bits*/**0**. See NOTE below.)

NOTE: In Linux kernels up to and including 2.6.38, with the exception of longterm releases 2.6.32 (>=.42), 2.6.33 (>=.15), and 2.6.35 (>=.14), there is a bug whereby IPv6 TOS mangling does not behave as documented and differs from the IPv4 version. The TOS mask indicates the bits one wants to zero out, so it needs to be inverted before applying it to the original TOS field. However, the aformentioned kernels forgo the inversion which breaks −−set−tos and its mnemonics.

TPROXY
This target is only valid in the **mangle** table, in the **PREROUTING** chain and user-defined chains which are only called from this chain. It redirects the packet to a local socket without changing the packet header in any way. It can also change the mark value which can then be used in advanced routing rules. It takes three options:

−−**on−port** *port*
This specifies a destination port to use. It is a required option, 0 means the new destination port is the same as the original. This is only valid if the rule also specifies **−p tcp** or **−p udp**.

−−**on−ip** *address*
This specifies a destination address to use. By default the address is the IP address of the incoming interface. This is only valid if the rule also specifies **−p tcp** or **−p udp**.

−−**tproxy−mark** *value*[/*mask*]
Marks packets with the given value/mask. The fwmark value set here can be used by advanced routing. (Required for transparent proxying to work: otherwise these packets will get forwarded, which is probably not what you want.)

TRACE
This target marks packets so that the kernel will log every rule which match the packets as those traverse the tables, chains, rules. It can only be used in the **raw** table.

With iptables-legacy, a logging backend, such as ip(6)t_LOG or nfnetlink_log, must be loaded for this to be visible. The packets are logged with the string prefix: "TRACE: tablename:chainname:type:rulenum " where type can be "rule" for plain rule, "return" for implicit rule at the end of a user defined chain and "policy" for the policy of the built in chains.

With iptables-nft, the target is translated into nftables' **meta nftrace** expression. Hence the kernel sends trace events via netlink to userspace where they may be displayed using **xtables-monitor --trace** command. For details, refer to **xtables-monitor**(8).

TTL (IPv4-specific)
This is used to modify the IPv4 TTL header field. The TTL field determines how many hops (routers) a packet can traverse until it's time to live is exceeded.

Setting or incrementing the TTL field can potentially be very dangerous, so it should be avoided at any cost. This target is only valid in **mangle** table.

Don't ever set or increment the value on packets that leave your local network!

−−ttl−set *value*
Set the TTL value to 'value'.

−−ttl−dec *value*
Decrement the TTL value 'value' times.

−−ttl−inc *value*
Increment the TTL value 'value' times.

ULOG (IPv4-specific)

This is the deprecated ipv4-only predecessor of the NFLOG target. It provides userspace logging of matching packets. When this target is set for a rule, the Linux kernel will multicast this packet through a *netlink* socket. One or more userspace processes may then subscribe to various multicast groups and receive the packets. Like LOG, this is a "non-terminating target", i.e. rule traversal continues at the next rule.

−−ulog−nlgroup *nlgroup*
This specifies the netlink group (1-32) to which the packet is sent. Default value is 1.

−−ulog−prefix *prefix*
Prefix log messages with the specified prefix; up to 32 characters long, and useful for distinguishing messages in the logs.

−−ulog−cprange *size*
Number of bytes to be copied to userspace. A value of 0 always copies the entire packet, regardless of its size. Default is 0.

−−ulog−qthreshold *size*
Number of packet to queue inside kernel. Setting this value to, e.g. 10 accumulates ten packets inside the kernel and transmits them as one netlink multipart message to userspace. Default is 1 (for backwards compatibility).