iptables(8) - Linux man page

Name

iptables - administration tool for IPv4 packet filtering and NAT

Synopsis

iptables [**-t** *table*] {**-A**|**-D**} *chain rule-specification*

iptables [**-t** *table*] **-I** *chain* [*rulenum*] *rule-specification*

iptables [**-t** *table*] **-R** *chain rulenum rule-specification*

iptables [**-t** *table*] **-D** *chain rulenum*

iptables [**-t** *table*] **-S** [*chain* [*rulenum*]]

iptables [**-t** *table*] {**-F**|**-L**|**-Z**} [*chain* [*rulenum*]] [*options...*]

iptables [**-t** *table*] **-N** *chain*

iptables [**-t** *table*] **-X** [*chain*]

iptables [**-t** *table*] **-P** *chain target*

iptables [**-t** *table*] **-E** *old-chain-name new-chain-name*

rule-specification = [*matches...*] [*target*]

match = **-m** *matchname* [*per-match-options*]

target = **-j** *targetname* [*per-target-options*]

Description

Iptables is used to set up, maintain, and inspect the tables of IPv4 packet filter rules in the Linux kernel. Several different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

Each chain is a list of rules which can match a set of packets. Each rule specifies what to do with a packet that matches. This is called a 'target', which may be a jump to a user-defined chain in the same table.

Targets

A firewall rule specifies criteria for a packet and a target. If the packet does not match, the next rule in the chain is the examined; if it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain or one of the special values **ACCEPT**, **DROP**, **QUEUE** or **RETURN**.

ACCEPT means to let the packet through. **DROP** means to drop the packet on the floor. **QUEUE** means to pass the packet to userspace. (How the packet can be received by a userspace process differs by the particular queue handler. 2.4.x and 2.6.x kernels up to 2.6.13 include the **ip_queue** queue handler. Kernels 2.6.14 and later additionally include the **nfnetlink_queue** queue handler. Packets with a target of QUEUE will be sent to queue number '0' in this case. Please also see the **NFQUEUE** target as described later in this man page.) **RETURN** means stop traversing this chain and resume at the next rule in the previous (calling) chain. If the end of a built-in chain is reached or a rule in a built-in chain with target **RETURN** is matched, the target specified by the chain policy determines the fate of the packet.

Tables

There are currently three independent tables (which tables are present at any time depends on the kernel configuration options and which modules are present).

-t, **--table** *table*

This option specifies the packet matching table which the command should operate on. If the kernel is configured with automatic module loading, an attempt will be made to load the appropriate module for that table if it is not already there.

The tables are as follows:

filter:

This is the default table (if no -t option is passed). It contains the built-in chains **INPUT** (for packets destined to local sockets), **FORWARD** (for packets being routed through the box), and **OUTPUT** (for locally-generated packets).

nat:

This table is consulted when a packet that creates a new connection is encountered. It consists of three built-ins: **PREROUTING** (for altering packets as soon as they come in), **OUTPUT** (for altering locally-generated packets before routing), and **POSTROUTING** (for altering packets as they are about to go out).

mangle:

This table is used for specialized packet alteration. Until kernel 2.4.17 it had two built-in chains: **PREROUTING** (for altering incoming packets before routing) and **OUTPUT** (for altering locally-generated packets before routing). Since kernel 2.4.18, three other built-in chains are also supported: **INPUT** (for packets coming into the box itself), **FORWARD** (for altering packets being routed through the box), and **POSTROUTING** (for altering packets as they are about to go out).

raw:

This table is used mainly for configuring exemptions from connection tracking in combination with the NOTRACK target. It registers at the netfilter hooks with higher priority and is thus called before ip_conntrack, or any other IP tables. It provides the following built-in chains: **PREROUTING** (for packets arriving via any network interface) **OUTPUT** (for packets generated by local processes)

Options

The options that are recognized by **iptables** can be divided into several different groups.

COMMANDS

These options specify the desired action to perform. Only one of them can be specified on the command line unless otherwise stated below. For long versions of the command and option names, you need to use only enough letters to ensure that **iptables** can differentiate it from all other options.

-A, **--append** *chain rule-specification*

Append one or more rules to the end of the selected chain. When the source and/or destination names resolve to more than one address, a rule will be added for each possible address combination.

-D, **--delete** *chain rule-specification*

-D, **--delete** *chain rulenum*

Delete one or more rules from the selected chain. There are two versions of this command: the rule can be specified as a number in the chain (starting at 1 for the first rule) or a rule to match.

-I, **--insert** *chain* [*rulenum*] *rule-specification*

Insert one or more rules in the selected chain as the given rule number. So, if the rule number is 1, the rule or rules are inserted at the head of the chain. This is also the default if no rule number is specified.

-R, **--replace** *chain rulenum rule-specification*

Replace a rule in the selected chain. If the source and/or destination names resolve to multiple addresses, the command will fail. Rules are numbered starting at 1.

-L, **--list** [*chain*]

List all rules in the selected chain. If no chain is selected, all chains are listed. Like every other iptables command, it applies to the specified table (filter is the default), so NAT rules get listed by

```
iptables -t nat -n -L
```

Please note that it is often used with the **-n** option, in order to avoid long reverse DNS lookups. It is legal to specify the **-Z** (zero) option as well, in which case the **chain**(s) will be atomically listed and zeroed. The exact output is affected by the other arguments given. The exact rules are suppressed until you use

```
iptables -L -v
```

-S, **--list-rules** [*chain*]

Print all rules in the selected chain. If no chain is selected, all chains are printed like iptables-save. Like every other iptables command, it applies to the specified table (filter is the default).

-F, **--flush** [*chain*]

Flush the selected chain (all the chains in the table if none is given). This is equivalent to deleting all the rules one by one.

-Z, **--zero** [*chain* [*rulenum*]]

Zero the packet and byte counters in all chains, or only the given chain, or only the given rule in a chain. It is legal to specify the **-L**, **--list** (list) option as well, to see the counters immediately before they are cleared. (See above.)

-N, **--new-chain** *chain*

Create a new user-defined chain by the given name. There must be no target of that name already.

-X, **--delete-chain** [*chain*]

Delete the optional user-defined chain specified. There must be no references to the chain. If there are, you must delete or replace the referring rules before the chain can be deleted. The chain must be empty, i.e. not contain any rules. If no argument is given, it will attempt to delete every non-builtin chain in the table.

-P, **--policy** *chain target*

Set the policy for the chain to the given target. See the section **TARGETS** for the legal targets. Only built-in (non-user-defined) chains can have policies, and neither built-in nor user-defined chains can be policy targets.

-E, **--rename-chain** *old-chain new-chain*

Rename the user specified chain to the user supplied name. This is cosmetic, and has no effect on the structure of the table.

-h

Help. Give a (currently very brief) description of the command syntax.

PARAMETERS

The following parameters make up a rule specification (as used in the add, delete, insert, replace and append commands).

[**!**] **-p**, **--protocol** *protocol*

The protocol of the rule or of the packet to check. The specified protocol can be one of **tcp**, **udp**, **udplite**, **icmp**, **esp**, **ah**, **sctp** or **all**, or it can be a numeric value, representing one of these protocols or a different one. A protocol name from /etc/protocols is also allowed. A "!" argument before the protocol inverts the test. The number zero is equivalent to **all**. Protocol **all** will match with all protocols and is taken as default when this option is omitted.

[**!**] **-s**, **--source** *address*[*/mask*][**,**...]

Source specification. *Address* can be either a network name, a hostname, a network IP address (with */mask*), or a plain IP address. Hostnames will be resolved once only, before the rule is submitted to the kernel. Please note that specifying any name to be resolved with a remote query such as DNS is a really bad idea. The *mask* can be either a network mask or a plain number, specifying the number of 1's at the left side of the network mask. Thus, a mask of *24* is equivalent to *255.255.255.0*. A "!" argument before the address specification inverts the sense of the address. The flag **--src** is an alias for this option. Multiple addresses can be specified, but this will **expand to multiple rules** (when adding with -A), or will cause multiple rules to be deleted (with -D).

[**!**] **-d**, **--destination** *address*[*/mask*][**,**...]

Destination specification. See the description of the **-s** (source) flag for a detailed description of the syntax. The flag **--dst** is an alias for this option.

-j, **--jump** *target*

This specifies the target of the rule; i.e., what to do if the packet matches it. The target can be a user-defined chain (other than the one this rule is in), one of the special builtin targets which decide the fate of the packet immediately, or an extension (see **EXTENSIONS** below). If this option is omitted in a rule (and **-g** is not used), then matching the rule will have no effect on the packet's fate, but the counters on the rule will be incremented.

-g, **--goto** *chain*

This specifies that the processing should continue in a user specified chain. Unlike the --jump option return will not continue processing in this chain but instead in the chain that called us via --jump.

[**!**] **-i**, **--in-interface** *name*

Name of an interface via which a packet was received (only for packets entering the **INPUT**, **FORWARD** and **PREROUTING** chains). When the "!" argument is used before the interface

name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, any interface name will match.

[**!**] **-o**, **--out-interface** *name*

Name of an interface via which a packet is going to be sent (for packets entering the **FORWARD**, **OUTPUT** and **POSTROUTING** chains). When the "!" argument is used before the interface name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, any interface name will match.

[**!**] **-f**, **--fragment**

This means that the rule only refers to second and further fragments of fragmented packets. Since there is no way to tell the source or destination ports of such a packet (or ICMP type), such a packet will not match any rules which specify them. When the "!" argument precedes the "-f" flag, the rule will only match head fragments, or unfragmented packets.

-c, **--set-counters** *packets bytes*

This enables the administrator to initialize the packet and byte counters of a rule (during **INSERT**, **APPEND**, **REPLACE** operations).

OTHER OPTIONS

The following additional options can be specified:

-v, **--verbose**

Verbose output. This option makes the list command show the interface name, the rule options (if any), and the TOS masks. The packet and byte counters are also listed, with the suffix 'K', 'M' or 'G' for 1000, 1,000,000 and 1,000,000,000 multipliers respectively (but see the **-x** flag to change this). For appending, insertion, deletion and replacement, this causes detailed information on the rule or rules to be printed.

-n, **--numeric**

Numeric output. IP addresses and port numbers will be printed in numeric format. By default, the program will try to display them as host names, network names, or services (whenever applicable).

-x, **--exact**

Expand numbers. Display the exact value of the packet and byte counters, instead of only the rounded number in K's (multiples of 1000) M's (multiples of 1000K) or G's (multiples of 1000M). This option is only relevant for the **-L** command.

--line-numbers

When listing rules, add line numbers to the beginning of each rule, corresponding to that rule's position in the chain.

--modprobe=*command*

When adding or inserting rules into a chain, use *command* to load any necessary modules (targets, match extensions, etc).

Match Extensions

iptables can use extended packet matching modules. These are loaded in two ways: implicitly, when **-p** or **--protocol** is specified, or 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 following are included in the base package, and most of these can be preceded by a "**!**" to invert the sense of the match.

addrtype

This module matches packets based on their **address type.** Address types are used within the kernel 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

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

[**!**] **--ahspi** *spi*[**:***spi*]

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

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 -s 192.168.0.0/16 -m comment --comment "A privatized IP block"

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 ...

connlimit

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

[**!**] **--connlimit-above** *n*

Match if the number of existing connections is (not) 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.

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-above 2 -j ACCEPT

limit the number of parallel HTTP requests to 16 per class C sized 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

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*

[**!**] **--ctorigdstport** *port*

[**!**] **--ctreplsrcport** *port*

[**!**] **--ctrepldstport** *port*

Match against original/reply source/destination port (TCP/UDP/etc.) or GRE key.

[**!**] **--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

meaning that the packet is associated with no known connection

NEW

meaning that the packet has started a new connection, or otherwise associated with a connection which has not seen packets in both directions, and

ESTABLISHED

meaning that the packet is associated with a connection which has seen packets in both directions,

RELATED

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

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.

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 DCP option set.

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.

ecn

This allows you to match the ECN bits of the IPv4 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 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*]

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":

matching on source host

> "1000 packets per second for every host in 192.168.0.0/16"

matching on source prot

> "100 packets per second for every service of 192.168.1.1"

matching on subnet

> "10000 packets per minute for every /28 subnet in 10.0.0.0/8"

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 as a number, with an optional time quantum suffix; the default is 3/hour.

--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.

--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.

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.

icmp

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

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.

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 (unless the '!' flag is used). It can be used in combination with the **LOG** target to give limited logging, for example.

--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).

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 **-p tcp** or **-p udp**.

[**!**] **--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.

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.

[**!**] **--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 packets entering the **FORWARD**, **OUTPUT** and **POSTROUTING** chains). If the interface name ends in a "+", then any interface which begins with this name will match. Note that in the **nat** and **mangle OUTPUT** chains one cannot match on the bridge output port, however one can in the **filter OUTPUT** chain. If the packet won't leave by a bridge device or if it is yet unknown what the output device will be, then the packet won't match this option, unless '!' is used.

[**!**] **--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 modules 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.

--strict

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

[**!**] **--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.

--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.

--rateest1 *name*

Name of the first rate estimator.

--rateest2 *name*

Name of the second rate estimator (if difference is to be calculated).

--rateest-delta

Compare **difference**(s) to given **rate**(s)

--rateest1-bps *value*

--rateest2-bps *value*

Compare bytes per second.

--rateest1-pps *value*

--rateest2-pps *value*

Compare packets per second.

[**!**] **--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.

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

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).

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.

[**!**] **--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.

--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

Steve's ipt_recent website (http://snowman.net/projects/ipt_recent/) also has some examples of usage.

/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.

sctp

[**!**] **--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

chunk type available flags

DATA U B E 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.

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. As standard kernels do not ship this currently, the ipset or Xtables-addons package needs to be installed.

socket

This matches if an open socket can be found by doing a socket lookup on the packet.

--transparent

Ignore non-transparent sockets.

state

This module, when combined with connection tracking, 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. Possible states are **INVALID** meaning that the packet could not be identified for some reason which includes running out of memory and ICMP errors which don't correspond to any known connection, **ESTABLISHED** meaning that the packet is associated with a connection which has seen packets in both directions, **NEW** meaning that the packet has started a new connection, or otherwise associated with a connection which has not seen packets in both directions, and **RELATED** meaning that the packet is starting a new connection, but is associated with an existing connection, such as an FTP data transfer, or an ICMP error.

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 from 0 to 1 for a packet to be randomly matched. It works only with the **random** mode.

--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 modules 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 from which it starts looking for any matching. If not passed, default is the packet size.

[**!**] **--string** *pattern*

Matches the given pattern.

[**!**] **--hex-string** *pattern*

Matches the given pattern in hex notation.

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. If the first port is greater than the second one they will be swapped. 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.

time

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

--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.).

--utc

Interpret the times given for **--datestart**, **--datestop**, **--timestart** and **--timestop** to be UTC.

--localtz

Interpret the times given for **--datestart**, **--datestop**, **--timestart** and **--timestop** to be local kernel time. (Default)

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.)

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

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.

unclean

This module takes no options, but attempts to match packets which seem malformed or unusual. This is regarded as experimental.

Target Extensions

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

CHECKSUM

This target allows to selectively work 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

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.

--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 only 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.

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 one type of option:

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

which can specify a single new destination IP address, an inclusive range of IP addresses, and optionally, a port range (which is only valid if the rule also specifies **-p tcp** or **-p udp**). 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.

DSCP

This target allows to alter 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

This target allows to selectively work 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**.

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 header fields) via the kernel log (where it can be read with *dmesg* or **syslogd**(8)). 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 (numeric or see **syslog.conf**(5)).

--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 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. The target can only be used in the **mangle** table. It can, for example, be used in conjunction with routing based on fwmark (needs iproute2). 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). It takes one option:

--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 **-p tcp** or **-p udp**.

--random

Randomize source port mapping If option **--random** is used then port mapping will be randomized (kernel >= 2.6.21).

MIRROR

This is an experimental demonstration target which inverts the source and destination fields in the IP header and retransmits the packet. It is only valid in the **INPUT**, **FORWARD** and **PREROUTING** chains, and user-defined chains which are only called from those chains. Note that the outgoing packets are **NOT** seen by any packet filtering chains, connection tracking or NAT, to avoid loops and other problems.

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.

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 (1 - 2^32-1) 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*

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 is an extension of the QUEUE target. As opposed to QUEUE, it allows you to put a packet into any specific queue, identified by its 16-bit queue number. It can only be used with Kernel versions 2.6.14 or later, since it requires the **nfnetlink_queue** kernel support. The **queue-balance** option was added in Linux 2.6.31.

--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.

NOTRACK

This target disables connection tracking for all packets matching that rule.

It 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 127.0.0.1 address).

--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 **-p tcp** or **-p udp**.

--random

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

REJECT

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 (**port-unreachable** is the default). 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).

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

SAME

Similar to SNAT/DNAT depending on chain: it takes a range of addresses ('--to 1.2.3.4-1.2.3.7') and gives a client the same source-/destination-address for each connection.

N.B.: The DNAT target's **--persistent** option replaced the SAME target.

--to *ipaddr*[-*ipaddr*]

> Addresses to map source to. May be specified more than once for multiple ranges.

--nodst

> Don't use the destination-ip in the calculations when selecting the new source-ip

--random

> Port mapping will be forcibly randomized to avoid attacks based on port prediction (kernel >= 2.6.21).

SECMARK

This is used to set the security mark value associated with the packet for use by security subsystems such as SELinux. It is only valid in the **mangle** table. The mark is 32 bits wide.

--selctx *security_context*

SET

This modules 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 sets

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

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

where flags are **src** and/or **dst** specifications and there can be no more than six of them.

Use of -j SET requires that ipset kernel support is provided. As standard kernels do not ship this currently, the ipset or Xtables-addons package needs to be installed.

SNAT

This target is only valid in the **nat** table, in the **POSTROUTING** chain. 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 one type of option:

--to-source *ipaddr*[**-***ipaddr*][**:***port*[**-***port*]]

which can specify a single new source IP address, an inclusive range of IP addresses, and optionally, a port range (which is only valid if the rule also specifies **-p tcp** or **-p udp**). 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

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 (kernel >= 2.6.21).

--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.

TCPMSS

This target allows to alter 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**. It is only valid in the **mangle**

table.
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**.

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* 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. 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*.)

--or-tos *bits*

Binary OR the TOS value with *bits*. (Mnemonic for **--set-tos** *bits*/*bits*.)

--xor-tos *bits*

Binary XOR the TOS value with *bits*. (Mnemonic for **--set-tos** *bits*/**0**.)

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 packes so that the kernel will log every rule which match the packets as those traverse the tables, chains, rules. (The ipt_LOG or ip6t_LOG module is required for the logging.) 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.
It can only be used in the **raw** table.

TTL

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.

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

--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

This target 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).

Diagnostics

Various error messages are printed to standard error. The exit code is 0 for correct functioning. Errors which appear to be caused by invalid or abused command line parameters cause an exit code of 2, and other errors cause an exit code of 1.

Bugs

Bugs? What's this? ;-) Well, you might want to have a look at http://bugzilla.netfilter.org/

Compatibility With Ipchains

This **iptables** is very similar to ipchains by Rusty Russell. The main difference is that the chains **INPUT** and **OUTPUT** are only traversed for packets coming into the local host and originating from the local host respectively. Hence every packet only passes through one of the three chains (except loopback traffic, which involves both INPUT and OUTPUT chains); previously a forwarded packet would pass through all three.

The other main difference is that **-i** refers to the input interface; **-o** refers to the output interface, and both are available for packets entering the **FORWARD** chain.

The various forms of NAT have been separated out; **iptables** is a pure packet filter when using the default 'filter' table, with optional extension modules. This should simplify much of the previous confusion over the combination of IP masquerading and packet filtering seen previously. So the following options are handled differently:

-j MASQ
-M -S
-M -L

There are several other changes in iptables.

See Also

iptables-save(8), **iptables-restore**(8), **ip6tables**(8), **ip6tables-save**(8), **ip6tables-restore**(8), **libipq**(3).

The packet-filtering-HOWTO details iptables usage for packet filtering, the NAT-HOWTO details NAT, the netfilter-extensions-HOWTO details the extensions that are not in the standard distribution, and the netfilter-hacking-HOWTO details the netfilter internals.
See **http://www.netfilter.org/**.

Authors

Rusty Russell originally wrote iptables, in early consultation with Michael Neuling.

Marc Boucher made Rusty abandon ipnatctl by lobbying for a generic packet selection framework in iptables, then wrote the mangle table, the owner match, the mark stuff, and ran around doing cool stuff everywhere.

James Morris wrote the TOS target, and tos match.

Jozsef Kadlecsik wrote the REJECT target.

Harald Welte wrote the ULOG and NFQUEUE target, the new libiptc, as well as the TTL, DSCP, ECN matches and targets.

The Netfilter Core Team is: Marc Boucher, Martin Josefsson, Yasuyuki Kozakai, Jozsef Kadlecsik, Patrick McHardy, James Morris, Pablo Neira Ayuso, Harald Welte and Rusty Russell.

Man page originally written by Herve Eychenne <rv@wallfire.org>.

Referenced By

arptables(8), **brctl**(8), **collectd.conf**(5), **ebtables**(8), **ferm**(1), **firehol**(1), **firehol.conf**(5), **fwsnort**(8), **ipq_destroy_handle**(3), **ipq_get_msgerr**(3), **ipq_perror**(3), **ipq_read**(3), **ipq_set_mode**(3), **ipq_set_verdict**(3), **iptables-xml**(8), **iptstate**(8), **ipvsadm**(8), **mountd**(8), **rpc.statd**(8), **shorewall**(8), **shorewall.conf**(5)