iptables(8) - Linux man page

Name

iptables - administration tool for IPv4 packet filtering and NAT

Synopsis

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

Description

Iptables is used to set up, maintain, and inspect the tables of IP 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 specific action to perform. Only one of them can be specified on the command line unless otherwise specified below. For all the 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. As 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
```

-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*]

Zero the packet and byte counters in all chains. 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*, *icmp*, 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 (please note that specifying any name to be resolved with a remote query such as DNS is a really bad idea), a network IP address (with /mask), or a plain IP address. 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.

-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 *PKTS 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.

account

Account traffic for all hosts in defined network/netmask.

Features:

- long (one counter per protocol TCP/UDP/IMCP/Other) and short statistics

- one iptables rule for all hosts in network/netmask

- loading/saving counters (by reading/writting to procfs entries)

--aaddr *network/netmask*
: defines network/netmask for which make statistics.

--aname *name*
: defines name of list where statistics will be kept. If no is specified DEFAULT will be used.

--ashort
: table will colect only short statistics (only total counters without splitting it into protocols.

Example usage:

account traffic for/to 192.168.0.0/24 network into table mynetwork:

```
# iptables -A FORWARD -m account --aname mynetwork --aaddr 192.168.0.0/24
```

account traffic for/to WWW serwer for 192.168.0.0/24 network into table mywwwserver:

# iptables -A INPUT -p tcp --dport 80 -m account --aname mywwwserver --aaddr 192.168.0.0/24 --ashort

# iptables -A OUTPUT -p tcp --sport 80 -m account --aname mywwwserver --aaddr 192.168.0.0/24 --ashort

read counters:

# cat /proc/net/ipt_account/mynetwork # cat /proc/net/ipt_account/mywwwserver

set counters:

# echo "ip = 192.168.0.1 packets_src = 0" > /proc/net/ipt_account/mywwserver

Webpage: http://www.barbara.eu.org/~quaker/ipt_account/

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** FIXME

--src-type *type*
  Matches if the source address is of given type
--dst-type *type*
  Matches if the destination address is of given type

ah

This module matches the SPIs in Authentication header of IPsec packets.
--ahspi [!] *spi*[:*spi*]

childlevel

This is an experimental module. It matches on whether the packet is part of a master connection or one of its children (or grandchildren, etc). For instance, most packets are level 0. FTP data transfer is level 1.
--childlevel [!] *level*

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"
```

condition

This matches if a specific /proc filename is '0' or '1'.

--condition *[!] filename*

Match on boolean value stored in /proc/net/ipt_condition/filename file

connbytes

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

The counters are 64bit 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 transfered bytes per connection can also be viewed through /proc/net/ip_conntrack and accessed via ctnetlink

[**!**] **--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 TCP connections to a server per client IP address (or address block).

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

match if the number of existing tcp connections is (not) above n

--connlimit-mask *bits*

group hosts using mask

Examples:

# allow 2 telnet connections per client host

```
iptables -p tcp --syn --dport 23 -m connlimit --connlimit-above 2 -j REJECT
```

# you can also match the other way around:

```
iptables -p tcp --syn --dport 23 -m connlimit ! --connlimit-above 2 -j ACCEPT
```

# limit the nr 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
```

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).

connrate

This module matches the current transfer rate in a connection.

--connrate *[!] [from]:[to]*

Match against the current connection transfer rate being within 'from' and 'to' bytes per second. When the "!" argument is used before the range, the sense of the match is inverted.

conntrack

This module, when combined with connection tracking, allows access to more connection tracking information than the "state" match. (this module is present only if iptables was compiled under a kernel supporting this feature)

--ctstate *state*

Where state is a comma separated list of the connection states to match. Possible states are **INVALID** meaning that the packet is associated with no 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. **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.

--ctproto *proto*

Protocol to match (by number or name)

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

Match against original source address

--ctorigdst *[!] address[/mask]*
: Match against original destination address

--ctreplsrc *[!] address[/mask]*
: Match against reply source address

--ctrepldst *[!] address***[/***mask***]**
: Match against reply destination address

--ctstatus *[NONE|EXPECTED|SEEN_REPLY|ASSURED][,...]*
: Match against internal conntrack states

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

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 *DiffServ Class*
: Match the DiffServ class. This value may be any of the BE, EF, AFxx or CSx classes. It will then be converted into it's according numeric value.

dstlimit

This module allows you to limit the packet per second (pps) rate on a per destination IP or per destination port base. As opposed to the 'limit' match, every destination ip / destination port has it's own limit.
THIS MODULE IS DEPRECATED AND HAS BEEN REPLACED BY ''hashlimit''

--dstlimit *avg*
: Maximum average match rate (packets per second unless followed by /sec /minute /hour /day postfixes).

--dstlimit-mode *mode*
: The limiting hashmode. Is the specified limit per **dstip, dstip-dstport** tuple, **srcip-dstip** tuple, or per **srcipdstip-dstport** tuple.

--dstlimit-name *name*
: Name for /proc/net/ipt_dstlimit/* file entry

[*--dstlimit-burst* **burst***]*
: Number of packets to match in a burst. Default: 5

[*--dstlimit-htable-size* **size***]*
: Number of buckets in the hashtable

[*--dstlimit-htable-max* **max***]*
: Maximum number of entries in the hashtable

[*--dstlimit-htable-gcinterval* **interval***]*
: Interval between garbage collection runs of the hashtable (in miliseconds). Default is 1000 (1 second).

[*--dstlimit-htable-expire* **time**
: After which time are idle entries expired from hashtable (in miliseconds)? Default is 10000 (10 seconds).

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*]

fuzzy

This module matches a rate limit based on a fuzzy logic controller [FLC]

--lower-limit *number*
: Specifies the lower limit (in packets per second).

--upper-limit *number*
: Specifies the upper limit (in packets per second).

hashlimit

This patch adds a new match called 'hashlimit'. The idea is to have something like 'limit', but either per destination-ip or per (destip,destport) tuple.

It gives you the ability to express

'1000 packets per second for every host in 192.168.0.0/16'

'100 packets per second for every service of 192.168.1.1'

with a single iptables rule.

--hashlimit *rate*
    A rate just like the limit match

--hashlimit-burst *num*
    Burst value, just like limit match

--hashlimit-mode *destip* | *destip-destport*
    Limit per IP or per port

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

--hashlimit-htable-size *num*
    The number of buckets of the hash table

--hashlimit-htable-max *num*
    Maximum entries in the hash

--hashlimit-htable-expire *num*
    After how many miliseconds do hash entries expire

--hashlimit-htable-gcinterval *num*
    How many miliseconds 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 is loaded if '--protocol icmp' is specified. It provides the following option:

--icmp-type [!] *typename*
    This allows specification of the ICMP type, which can be a numeric ICMP type, or one of the ICMP type names shown by the command

```
iptables -p icmp -h
```

iprange

This matches on a given arbitrary range of IPv4 addresses

[!]*--src-range* **ip-ip**
    Match source IP in the specified range.

[!]*--dst-range* **ip-ip**

Match destination IP in the specified range.

ipv4options

Match on IPv4 header options like source routing, record route, timestamp and router-alert.

--ssrr

To match packets with the flag strict source routing.

--lsrr

To match packets with the flag loose source routing.

--no-srr

To match packets with no flag for source routing.

[**!**] **--rr**

To match packets with the RR flag.

[**!**] **--ts**

To match packets with the TS flag.

[**!**] **--ra**

To match packets with the router-alert option.

[**!**] **--any-opt**

To match a packet with at least one IP option, or no IP option at all if ! is chosen.

Examples:

$ iptables -A input -m ipv4options --rr -j DROP

will drop packets with the record-route flag.

$ iptables -A input -m ipv4options --ts -j DROP

will drop packets with the timestamp flag.

length

This module matches the length 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*

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).

mport

This module matches a set of source or destination ports. Up to 15 ports can be specified. It can only be used in conjunction with **-p tcp** or **-p udp**.

--source-ports *port*[,*port*[,*port*...]]
: Match if the source port is one of the given ports. The flag **--sports** is a convenient alias for this option.

--destination-ports *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*...]]
: Match if the both the source and destination ports are equal to each other and to one of the given ports.

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 *[!] 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.

--destination-ports *[!] 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.

nth

This module matches every 'n'th packet

--every *value*
: Match every 'value' packet

[--*counter* **num**]
: Use internal counter number 'num'. Default is '0'.

[--*start* **num**]
: Initialize the counter at the number 'num' insetad of '0'. Most between '0' and 'value'-1.

[--*packet* **num**]
: Match on 'num' packet. Most be between '0' and 'value'-1.

osf

The idea of passive OS fingerprint matching exists for quite a long time, but was created as extension fo OpenBSD pf only some weeks ago. Original idea was lurked in some OpenBSD mailing list (thanks grange@open...) and than adopted for Linux netfilter in form of this code.

Original fingerprint table was created by Michal Zalewski <lcamtuf@coredump.cx>.

This module compares some data(WS, MSS, options and it's order, ttl, df and others) from first SYN packet (actually from packets with SYN bit set) with dynamically loaded OS fingerprints.

--log 1/0

If present, OSF will log determined genres even if they don't match desired one.

> 0 - log all determined entries, 1 - only first one.
> In syslog you find something like this:
> ipt_osf: Windows [2000:SP3:Windows XP Pro SP1, 2000 SP3]: 11.22.33.55:4024 -> 11.22.33.44:139
>
> ipt_osf: Unknown: 16384:106:1:48:020405B401010402 44.33.22.11:1239 -> 11.22.33.44:80

--smart
: if present, OSF will use some smartness to determine remote OS. OSF will use initial TTL only if source of connection is in our local network.

--netlink
: If present, OSF will log all events also through netlink NETLINK_NFLOG groupt 1.

--genre *[!] string*
: Match a OS genre by passive fingerprinting

Example:

```
#iptables -I INPUT -j ACCEPT -p tcp -m osf --genre Linux --log 1 --smart
```

NOTE: -p tcp is obviously required as it is a TCP match.

Fingerprints can be loaded and read through /proc/sys/net/ipv4/osf file. One can flush all fingerprints with following command:

```
echo -en FLUSH > /proc/sys/net/ipv4/osf
```

Only one fingerprint per open/write/close.

Fingerprints 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. It is only valid in the **OUTPUT** chain, and even this some packets (such as ICMP ping responses) may have no owner, and hence never match.

--uid-owner *userid*
: Matches if the packet was created by a process with the given effective user id.

--gid-owner *groupid*
: Matches if the packet was created by a process with the given effective group id.

--pid-owner *processid*
: Matches if the packet was created by a process with the given process id.

--sid-owner *sessionid*
: Matches if the packet was created by a process in the given session group.

--cmd-owner *name*
: Matches if the packet was created by a process with the given command name. (this option is present only if iptables was compiled under a kernel supporting this feature)

NOTE: pid, sid and command matching are broken on SMP

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 it is yet unknown what the output device will be, then the packet won't match this option, unless

[!] **--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

psd

Attempt to detect TCP and UDP port scans. This match was derived from Solar Designer's scanlogd.

--psd-weight-threshold *threshold*
: Total weight of the latest TCP/UDP packets with different destination ports coming from the same host to be treated as port scan sequence.

--psd-delay-threshold *delay*
: Delay (in hundredths of second) for the packets with different destination ports coming from the same host to be treated as possible port scan subsequence.

--psd-lo-ports-weight *weight*
: Weight of the packet with privileged (<=1024) destination port.

--psd-hi-ports-weight *weight*
: Weight of the packet with non-priviliged destination port.

quota

Implements network quotas by decrementing a byte counter with each packet.

--quota *bytes*
: The quota in bytes.

KNOWN BUGS: this does not work on SMP systems.

random

This module randomly matches a certain percentage of all packets.

--average *percent*
: Matches the given percentage. If omitted, a probability of 50% is set.

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).

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.

--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).

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

--rttl

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 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
```

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

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

Each file in /proc/net/ipt_recent/ can be read from to see the current list or written two using the following commands to modify the list:

echo xx.xx.xx.xx > /proc/net/ipt_recent/DEFAULT
    to Add to the DEFAULT list

echo -xx.xx.xx.xx > /proc/net/ipt_recent/DEFAULT
    to Remove from the DEFAULT list

echo clear > /proc/net/ipt_recent/DEFAULT
    to empty the DEFAULT list.

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/ipt_recent/* files

debug=*0*
    Set to 1 to get lots of debugging info

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 modules macthes IP sets which can be defined by **ipset**(8).

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

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

```
iptables -A FORWARD -m set --set test src,dst
```

will match packets, for which (depending on the type of the set) the source address or port number of the packet can be found in the specified set. If there is a binding belonging to the mached set element or there is a default binding for the given set, then the rule will match the packet only if additionally (depending on the type of the set) the destination address or port number of the packet can be found in the set according to the binding.

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.

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 are loaded if '--protocol tcp' is specified. It provides the following options:

--source-port [!] *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 *port*:*port*. If the first port is omitted, "0" is assumed; if the last is omitted, "65535" is assumed. If the second port greater then the first they will be swapped. The flag **--sport** is a convenient alias for this option.

--destination-port [!] *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 is the flags which we should examine, written as a comma-separated list, and the second argument 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.

--mss *value*[:*value*]
Match TCP SYN or SYN/ACK packets with the specified MSS value (or range), which control the maximum packet size for that connection.

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 facultative.

--timestart *value*
    Match only if it is after 'value' (Inclusive, format: HH:MM ; default 00:00).

--timestop *value*
    Match only if it is before 'value' (Inclusive, format: HH:MM ; default 23:59).

--days *listofdays*
    Match only if today is one of the given days. (format: Mon,Tue,Wed,Thu,Fri,Sat,Sun ; default everyday)

--datestart *date*
    Match only if it is after 'date' (Inclusive, format: YYYY[:MM[:DD[:hh[:mm[:ss]]]]] ; h,m,s start from 0 ; default to 1970)

--datestop *date*
    Match only if it is before 'date' (Inclusive, format: YYYY[:MM[:DD[:hh[:mm[:ss]]]]] ; h,m,s start from 0 ; default to 2037)

tos

This module matches the 8 bits of Type of Service field in the IP header (ie. including the precedence bits).

--tos *tos*
    The argument is either a standard name, (use
    iptables -m tos -h
    to see the list), or a numeric value to match.

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 allows you to extract quantities of up to 4 bytes from a packet, AND them with specified masks, shift them by specified amounts and test whether the results are in any of a set of specified ranges. The specification of what to extract is general enough to skip over headers with lengths stored in the packet, as in IP or TCP header lengths.

Details and examples are in the kernel module source.

udp

These extensions are loaded if '--protocol udp' is specified. It provides the following options:

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

--destination-port [!] *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.

BALANCE

This allows you to DNAT connections in a round-robin way over a given range of destination addresses.

--to-destination *ipaddr-ipaddr*
    Address range to round-robin over.

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.

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
--set-mark mark[/mask]
  Set connection mark. If a mask is specified then only those bits set in the mask is modified.
--save-mark [--mask mask]
  Copy the netfilter packet mark value to the connection mark. If a mask is specified then only those bits are copied.
--restore-mark [--mask mask]
  Copy the connection mark value to the packet. If a mask is specified then only those bits are copied. This is only valid in the **mangle** table.

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.
  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.

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**.

IPMARK

Allows you to mark a received packet basing on its IP address. This can replace many mangle/mark entries with only one, if you use firewall based classifier.

This target is to be used inside the mangle table, in the PREROUTING, POSTROUTING or FORWARD hooks.

--addr *src/dst*

Use source or destination IP address.

--and-mask *mask*

Perform bitwise 'and' on the IP address and this mask.

--or-mask *mask*

Perform bitwise 'or' on the IP address and this mask.

The order of IP address bytes is reversed to meet "human order of bytes": 192.168.0.1 is 0xc0a80001. At first the 'and' operation is performed, then 'or'.

Examples:

We create a queue for each user, the queue number is adequate to the IP address of the user, e.g.: all packets going to/from 192.168.5.2 are directed to 1:0502 queue, 192.168.5.12 -> 1:050c etc.

We have one classifier rule:

```
tc filter add dev eth3 parent 1:0 protocol ip fw
```

Earlier we had many rules just like below:

```
iptables -t mangle -A POSTROUTING -o eth3 -d 192.168.5.2 -j MARK --set-mark
0x10502

iptables -t mangle -A POSTROUTING -o eth3 -d 192.168.5.3 -j MARK --set-mark
0x10503
```

Using IPMARK target we can replace all the mangle/mark rules with only one:

iptables -t mangle -A POSTROUTING -o eth3 -j IPMARK --addr=dst --and-mask=0xffff --or-mask=0x10000

On the routers with hundreds of users there should be significant load decrease (e.g. twice).

IPV4OPTSSTRIP

Strip all the IP options from a packet.

The target doesn't take any option, and therefore is extremly easy to use :

```
# iptables -t mangle -A PREROUTING -j IPV4OPTSSTRIP
```

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 is used to set the netfilter mark value associated with the packet. It is only valid in the **mangle** table. It can for example be used in conjunction with iproute2.

--set-mark *mark*

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**.

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.

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.

--queue-num *value*
: This specifies the QUEUE number to use. Valud queue numbers are 0 to 65535. The default value is 0.

It can only be used with Kernel versions 2.6.14 or later, since it requires
: the **nfnetlink_queue** kernel support.

NOTRACK

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

It can only be used in the
: **raw** table.

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). It takes one option:

--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**.

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.

--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

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** and there can be no more than six of them.

The bindings to follow must previously be defined in order to use
multilevel adding/deleting by the SET target.

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 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.

TARPIT

Captures and holds incoming TCP connections using no local per-connection resources. Connections are accepted, but immediately switched to the persist state (0 byte window), in which the remote side stops sending data and asks to continue every 60-240 seconds. Attempts to close the connection are ignored, forcing the remote side to time out the connection in 12-24 minutes.

This offers similar functionality to LaBrea <http://www.hackbusters.net/LaBrea/> but doesn't require dedicated hardware or IPs. Any TCP port that you would normally DROP or REJECT can instead become a tarpit.

To tarpit connections to TCP port 80 destined for the current machine:

```
iptables -A INPUT -p tcp -m tcp --dport 80 -j TARPIT
```

To significantly slow down Code Red/Nimda-style scans of unused address space, forward unused ip addresses to a Linux box not acting as a router (e.g. "ip route 10.0.0.0 255.0.0.0 ip.of.linux.box" on a Cisco), enable IP forwarding on the Linux box, and add:

```
iptables -A FORWARD -p tcp -j TARPIT

iptables -A FORWARD -j DROP
```

NOTE:

If you use the conntrack module while you are using TARPIT, you should also use the NOTRACK target, or the kernel will unnecessarily allocate resources for each TARPITted connection. To TARPIT incoming connections to the standard IRC port while using conntrack, you could:

    iptables -t raw -A PREROUTING -p tcp --dport 6667 -j NOTRACK

    iptables -A INPUT -p tcp --dport 6667 -j TARPIT

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). 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 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 set MSS option to specified value.
--clamp-mss-to-pmtu
    Automatically clamp MSS value to (path_MTU - 40).
These options are mutually exclusive.

TOS

This is used to set the 8-bit Type of Service field in the IP header. It is only valid in the **mangle** table.
--set-tos *tos*

You can use a numeric TOS values, or use

```
iptables -j TOS -h
```

to see the list of valid TOS names.

TRACE

This target has no options. It just turns on **packet tracing** for all packets that match this rule.

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).

XOR

Encrypt TCP and UDP traffic using a simple XOR encryption

--key *string*
  Set key to "string"

--block-size
  Set block size

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.

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, Jozsef Kadlecsik, Patrick McHardy, James Morris, 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), **ip6tables-1.4.7**(8), **ipq_destroy_handle**(3), **ipq_get_msgerr**(3), **ipq_perror**(3), **ipq_read**(3), **ipq_set_mode**(3), **ipq_set_verdict**(3), **iptables-restore-1.4.7**(8), **iptables-save-1.4.7**(8), **iptables-xml-1.4.7**(8), **iptables_selinux**(8), **iptstate**(8), **ipvsadm**(8), **mountd**(8), **rpc.statd**(8), **shorewall**(8), **shorewall-lite**(8), **shorewall.conf**(5), **shorewall6-lite**(8)