NAME

iptables − IP packet filter administration

SYNOPSIS

iptables -[ADC] chain rule-specification [options]
iptables -[RI] chain rulenum rule-specification [options]
iptables -D chain rulenum [options]
iptables -[LFZ] [chain] [options]
iptables -[NX] chain
iptables -P chain target [options]
iptables -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 (if supported by the kernel). *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 current three independent tables (which tables are present at any time depends on the kernel configuration options and which modules are present).

-t, --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. It contains the built-in chains INPUT (for packets coming into the box itself), 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. It has two built-in chains: PREROUTING (for altering incoming packets before routing) and OUTPUT (for altering locally-generated packets before routing).

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
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
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.

-R, --replace
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.

-I, --insert
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.

-L, --list
List all rules in the selected chain. If no chain is selected, all chains are listed. 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.

-F, --flush
Flush the selected chain. This is equivalent to deleting all the rules one by one.

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

-X, --delete-chain
Delete the specified user-defined chain. 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. If no argument is given, it will attempt to delete every non-builtin chain in the table.

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

-E, --rename-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 hostname, a network name, 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 a convenient 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, then matching the rule will have no effect on the packet's fate, but the counters on the rule will be incremented.

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

Optional name of an interface via which a packet is received (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, the string "+" is assumed, which will match with any interface name.

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

Optional 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, the string "+" is assumed, which will match with any interface name.

[!] -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 administrater 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 address, 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.

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 an alias for this option.

--destination-port [!] [*port[:port]*]
Destination port or port range specification. The flag **--dport** is an 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 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 SYN**. If the "!" flag precedes the "--syn", the sense of the option is inverted.

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

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.

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

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 entering the **PREROUTING**, **FORWARD** or **INPUT** chains for packets coming from an ethernet device.

limit

This module matches at a limited rate using a token bucket filter: it can be used in combination with the **LOG** target to give limited logging. A rule using this extension will match until this limit is reached (unless the '!' flag is used).

--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*

The 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.

multiport

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-port [*port[,port]*]

Match if the source port is one of the given ports.

--destination-port [*port[,port]*]

Match if the destination port is one of the given 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.

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 mark before the comparison).

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.

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 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.

unclean

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

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.

TARGET EXTENSIONS

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

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)).

--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.

MARK

This is used to set the netfilter mark value associated with the packet. It is only valid in the **mangle** table.

--set-mark *mark*

REJECT

This is used to send back an error packet in response to the matched packet: otherwise it is equivalent to **DROP**. This target is only valid in the **INPUT**, **FORWARD** and **OUTPUT** chains, and user-defined chains which are only called from those chains. Several options control 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**or **icmp-host-prohibited**, which return the appropriate ICMP error message (port-unreachable is the default). The option **echo-reply** is also allowed; it can only be used for rules which specify an ICMP ping packet, and generates a ping reply. Finally, the option **tcp-reset** can be used on rules which only match the TCP protocol: this causes a TCP RST packet to be sent back. This is mainly useful for blocking *ident* probes which

frequently occur when sending mail to broken mail hosts (which won't accept your mail otherwise).

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.

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.

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 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.

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 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.

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 with if the rule also specifies **-p tcp** or **-p udp**).

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 alters the destination IP address to send the packet to the machine itself (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 or ports to use: without this, the destination port is never altered. This is only valid with if the rule also specifies **-p tcp** or **-p udp**).

EXTRA EXTENSIONS

The following extensions are not included by default in the standard distribution.

ttl

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

--ttl *ttl* Matches the given TTL value.

TTL

This target is used to modify the time to live field in the IP header. It is only valid in the **mangle** table.

--ttl-set *ttl*
Set the TTL to the given value.

--ttl-dec *ttl*
Decrement the TTL by the given value.

--ttl-inc *ttl*
Increment the TTL by the given value.

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.

--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 fro 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 multpart 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

Check is not implemented (yet).

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; 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

The iptables-HOWTO, which details more iptables usage, the NAT-HOWTO, which details NAT, and the netfilter-hacking-HOWTO which details the internals.

AUTHORS

Rusty Russell 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 target, TTL match+target and libipulog.

The Netfilter Core Team is: Marc Boucher, James Morris, Harald Welte and Rusty Russell.