About.com Linux

By Juergen Haas, About.com

# Linux / Unix Command: *iptables*

Command Library [1]

**NAME**

iptables - administration tool for IPv4 packet filtering and NAT

**SYNOPSIS**

**iptables [-t table] -[ADC]** 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 (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 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 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. 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).

**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. 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, 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***
Name of an interface via which a packet is going to be 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.

**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 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.

**--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.

**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 coming from an Ethernet device and entering the **PREROUTING**, **FORWARD** or **INPUT** chains.

**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.

**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-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.

**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).

**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)

**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.

**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)

**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 [0-32].

**--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.

**pkttype**

This module matches the link-layer packet type.

**--pkt-type *[unicast|broadcast|multicast*]**

**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.

**ah**

This module matches the SPIs in AH header of IPSec packets.

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

**esp**

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

**--espspi [!] *spi*[:*spi*]**

**length**

This module matches the length of a packet against a specific value or range of values.

**--length *length*[:*length*]**

**ttl**

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

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

**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.

**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.

**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***

**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 or icmp-host-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).

**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 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.

**You can add several --to-source options. 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 adresses.

**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.

**You can add several --to-destination options. 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 adresses.

**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**.

**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 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**.

**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).

**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**.
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 -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.**

**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**.

**SEE ALSO**

**iptables-save**(8), **iptables-restore**(8), **ip6tables**(8), **ip6tables-save**(8), **ip6tables-restore(8).** 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/**.

---

**Important:** Use the *man* command (*% man*) to see how a command is used on your particular computer.

**>> Linux/Unix Command Library**

**>> Shell Command Library**