Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'ebtables.8' command

$ man ebtables.8

EBTABLES(8) System Manager's Manual EBTABLES(8)

NAME

ebtables - Ethernet bridge frame table administration (nft-based)

SYNOPSIS

ebtables [-t table] -[ACDI] chain rule specification [match exten?

sions] [watcher extensions] target

ebtables [-t table] -P chain ACCEPT | DROP | RETURN

ebtables [-t table] -F [chain]

ebtables [-t table] -Z [chain]

ebtables [-t table] -L [-Z] [chain] [[--Ln] | [--Lx]] [--Lc]

[--Lmac2]

ebtables [-t table] -N chain [-P ACCEPT | DROP | RETURN]

ebtables [-t table] -X [chain]

ebtables [-t table] -E old-chain-name new-chain-name

ebtables [-t table] --init-table

DESCRIPTION

This tool is deprecated in Red Hat Enterprise Linux. It is maintenance

only and will not receive new features. New setups should use nft(8).

Existing setups should migrate to nft(8) when possible. See

?https://red.ht/nft_your_tables? for details.

ebtables is an application program used to set up and maintain the ta?

bles of rules (inside the Linux kernel) that inspect Ethernet frames.

It is analogous to the iptables application, but less complicated, due

to the fact that the Ethernet protocol is much simpler than the IP pro?
tocol.

CHAINS

There are two ebtables tables with built-in chains in the Linux kernel. These tables are used to divide functionality into different sets of rules. Each set of rules is called a chain. Each chain is an ordered list of rules that can match Ethernet frames. If a rule matches an Eth? ernet frame, then a processing specification tells what to do with that matching frame. The processing specification is called a 'target'. How? ever, if the frame does not match the current rule in the chain, then the next rule in the chain is examined and so forth. The user can cre? ate new (user-defined) chains that can be used as the 'target' of a rule. User-defined chains are very useful to get better performance over the linear traversal of the rules and are also essential for structuring the filtering rules into well-organized and maintainable sets of rules.

TARGETS

A firewall rule specifies criteria for an Ethernet frame and a frame processing specification called a target. When a frame matches a rule, then the next action performed by the kernel is specified by the tar? get. The target can be one of these values: ACCEPT, DROP, CONTINUE, RETURN, an 'extension' (see below) or a jump to a user-defined chain. ACCEPT means to let the frame through. DROP means the frame has to be dropped. CONTINUE means the next rule has to be checked. This can be handy, f.e., to know how many frames pass a certain point in the chain, to log those frames or to apply multiple targets on a frame. RETURN means stop traversing this chain and resume at the next rule in the previous (calling) chain. For the extension targets please refer to the TARGET EXTENSIONS section of this man page.

TABLES

As stated earlier, there are two ebtables tables in the Linux kernel. The table names are filter and nat. Of these two tables, the filter table is the default table that the command operates on. If you are

working with the filter table, then you can drop the '-t filter' argu?
ment to the ebtables command. However, you will need to provide the -t
argument for nat table. Moreover, the -t argument must be the first
argument on the ebtables command line, if used.

-t, --table

filter is the default table and contains three built-in chains:
INPUT (for frames destined for the bridge itself, on the level
of the MAC destination address), OUTPUT (for locally-generated
or (b)routed frames) and FORWARD (for frames being forwarded by
the bridge).

nat is mostly used to change the mac addresses and contains
three built-in chains: PREROUTING (for altering frames as soon
as they come in), OUTPUT (for altering locally generated or
(b)routed frames before they are bridged) and POSTROUTING (for
altering frames as they are about to go out). A small note on
the naming of chains PREROUTING and POSTROUTING: it would be
more accurate to call them PREFORWARDING and POSTFORWARDING, but
for all those who come from the iptables world to ebtables it is
easier to have the same names. Note that you can change the name
(-E) if you don't like the default.

EBTABLES COMMAND LINE ARGUMENTS

After the initial ebtables '-t table' command line argument, the re?
maining arguments can be divided into several groups. These groups are
commands, miscellaneous commands, rule specifications, match exten?
sions, watcher extensions and target extensions.

COMMANDS

The ebtables command arguments specify the actions to perform on the
table defined with the -t argument. If you do not use the -t argument
to name a table, the commands apply to the default filter table. Only
one command may be used on the command line at a time, except when the
commands -L and -Z are combined or the commands -N and -P are combined.

-A, --append

Append a rule to the end of the selected chain.

-D, --delete

Delete the specified rule or rules from the selected chain. There are two ways to use this command. The first is by specify? ing an interval of rule numbers to delete (directly after -D). Syntax: start_nr[:end_nr] (use -L --Ln to list the rules with their rule number). When end_nr is omitted, all rules starting from start_nr are deleted. Using negative numbers is allowed, for more details about using negative numbers, see the -I com? mand. The second usage is by specifying the complete rule as it would have been specified when it was added. Only the first en? countered rule that is the same as this specified rule, in other words the matching rule with the lowest (positive) rule number, is deleted.

-C, --change-counters

Change the counters of the specified rule or rules from the se? lected chain. There are two ways to use this command. The first is by specifying an interval of rule numbers to do the changes on (directly after -C). Syntax: start_nr[:end_nr] (use -L --Ln to list the rules with their rule number). The details are the same as for the -D command. The second usage is by specifying the complete rule as it would have been specified when it was added. Only the counters of the first encountered rule that is the same as this specified rule, in other words the matching rule with the lowest (positive) rule number, are changed. In the first usage, the counters are specified directly after the interval specification, in the second usage directly after -C. First the packet counter is specified, then the byte counter. If the specified counters start with a '+', the counter values are added to the respective current counter values. If the speci? fied counters start with a '-', the counter values are decreased from the respective current counter values. No bounds checking is done. If the counters don't start with '+' or '-', the cur? rent counters are changed to the specified counters.

-I, --insert

Insert the specified rule into the selected chain at the speci? fied rule number. If the rule number is not specified, the rule is added at the head of the chain. If the current number of rules equals N, then the specified number can be between -N and N+1. For a positive number i, it holds that i and i-N-1 specify the same place in the chain where the rule should be inserted. The rule number 0 specifies the place past the last rule in the chain and using this number is therefore equivalent to using the -A command. Rule numbers structly smaller than 0 can be useful when more than one rule needs to be inserted in a chain.

-P, --policy

Set the policy for the chain to the given target. The policy can be ACCEPT, DROP or RETURN.

-F, --flush

Flush the selected chain. If no chain is selected, then every chain will be flushed. Flushing a chain does not change the pol? icy of the chain, however.

-Z, --zero

Set the counters of the selected chain to zero. If no chain is selected, all the counters are set to zero. The -Z command can be used in conjunction with the -L command. When both the -Z and -L commands are used together in this way, the rule counters are printed on the screen before they are set to zero.

-L, --list

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

The following options change the output of the -L command.

--Ln

Places the rule number in front of every rule. This option is incompatible with the --Lx option.

--Lc

Shows the counters at the end of each rule displayed by the -L

command. Both a frame counter (pcnt) and a byte counter (bcnt) are displayed. The frame counter shows how many frames have matched the specific rule, the byte counter shows the sum of the frame sizes of these matching frames. Using this option in com? bination with the --Lx option causes the counters to be written out in the '-c <pcnt> <bcnt>' option format.

--Lx

Changes the output so that it produces a set of ebtables com? mands that construct the contents of the chain, when specified. If no chain is specified, ebtables commands to construct the contents of the table are given, including commands for creating the user-defined chains (if any). You can use this set of com? mands in an ebtables boot or reload script. For example the output could be used at system startup. The --Lx option is in? compatible with the --Ln listing option. Using the --Lx option together with the --Lc option will cause the counters to be written out in the '-c <pcnt> <bcnt>' option format.

--Lmac2

Shows all MAC addresses with the same length, adding leading ze? roes if necessary. The default representation omits leading ze? roes in the addresses.

-N, --new-chain

Create a new user-defined chain with the given name. The number of user-defined chains is limited only by the number of possible chain names. A user-defined chain name has a maximum length of 31 characters. The standard policy of the user-defined chain is ACCEPT. The policy of the new chain can be initialized to a dif? ferent standard target by using the -P command together with the -N command. In this case, the chain name does not have to be specified for the -P command.

-X, --delete-chain

Delete the specified user-defined chain. There must be no re? maining references (jumps) to the specified chain, otherwise

ebtables will refuse to delete it. If no chain is specified, all user-defined chains that aren't referenced will be removed.

-E, --rename-chain

Rename the specified chain to a new name. Besides renaming a user-defined chain, you can rename a standard chain to a name that suits your taste. For example, if you like PREFORWARDING more than PREROUTING, then you can use the -E command to rename the PREROUTING chain. If you do rename one of the standard ebta? bles chain names, please be sure to mention this fact should you post a question on the ebtables mailing lists. It would be wise to use the standard name in your post. Renaming a standard ebta? bles chain in this fashion has no effect on the structure or functioning of the ebtables kernel table.

--init-table

Replace the current table data by the initial table data.

MISCELLANOUS COMMANDS

-v, --verbose

Verbose mode. For appending, insertion, deletion and replace? ment, this causes detailed information on the rule or rules to be printed. -v may be specified multiple times to possibly emit more detailed debug statements.

-V, --version

Show the version of the ebtables userspace program.

-h, --help [list of module names]

Give a brief description of the command syntax. Here you can also specify names of extensions and ebtables will try to write help about those extensions. E.g. ebtables -h snat log ip arp. Specify list_extensions to list all extensions supported by the userspace utility.

-j, --jump target

The target of the rule. This is one of the following values: AC? CEPT, DROP, CONTINUE, RETURN, a target extension (see TARGET EX? TENSIONS) or a user-defined chain name.

-M, --modprobe program

When talking to the kernel, use this program to try to automati?

cally load missing kernel modules.

--concurrent

Use a file lock to support concurrent scripts updating the ebta?

bles kernel tables.

RULE SPECIFICATIONS

The following command line arguments make up a rule specification (as used in the add and delete commands). A "!" option before the specifi? cation inverts the test for that specification. Apart from these stan? dard rule specifications there are some other command line arguments of interest. See both the MATCH EXTENSIONS and the WATCHER EXTENSIONS be? low.

-p, --protocol [!] protocol

The protocol that was responsible for creating the frame. This can be a hexadecimal number, above 0x0600, a name (e.g. ARP) or LENGTH. The protocol field of the Ethernet frame can be used to denote the length of the header (802.2/802.3 networks). When the value of that field is below or equals 0x0600, the value equals the size of the header and shouldn't be used as a proto? col number. Instead, all frames where the protocol field is used as the length field are assumed to be of the same 'protocol'. The protocol name used in ebtables for these frames is LENGTH. The file /etc/ethertypes can be used to show readable characters instead of hexadecimal numbers for the protocols. For example, 0x0800 will be represented by IPV4. The use of this file is not case sensitive. See that file for more information. The flag --proto is an alias for this option.

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

The interface (bridge port) via which a frame is received (this option is useful in the INPUT, FORWARD, PREROUTING and BROUTING chains). If the interface name ends with '+', then any interface name that begins with this name (disregarding '+') will match.

The flag --in-if is an alias for this option.

--logical-in [!] name

The (logical) bridge interface via which a frame is received (this option is useful in the INPUT, FORWARD, PREROUTING and BROUTING chains). If the interface name ends with '+', then any interface name that begins with this name (disregarding '+') will match.

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

The interface (bridge port) via which a frame is going to be sent (this option is useful in the OUTPUT, FORWARD and POSTROUT? ING chains). If the interface name ends with '+', then any in? terface name that begins with this name (disregarding '+') will match. The flag --out-if is an alias for this option.

--logical-out [!] name

The (logical) bridge interface via which a frame is going to be sent (this option is useful in the OUTPUT, FORWARD and POSTROUT? ING chains). If the interface name ends with '+', then any in? terface name that begins with this name (disregarding '+') will match.

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

The source MAC address. Both mask and address are written as 6 hexadecimal numbers separated by colons. Alternatively one can specify Unicast, Multicast, Broadcast or BGA (Bridge Group Ad? dress):

Unicast=00:00:00:00:00:00/01:00:00:00:00:00, Multi? cast=01:00:00:00:00:00/01:00:00:00:00:00, Broad? cast=ff:ff:ff:ff:ff:ff/ff:ff:ff:ff:ff:ff or BGA=01:80:c2:00:00:00/ff:ff:ff:ff:ff:ff. Note that a broadcast address will also match the multicast specification. The flag --src is an alias for this option.

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

The destination MAC address. See -s (above) for more details on MAC addresses. The flag --dst is an alias for this option.

-c, --set-counter pcnt bcnt

If used with -A or -I, then the packet and byte counters of the new rule will be set to pcnt, resp. bcnt. If used with the -C or -D commands, only rules with a packet and byte count equal to pcnt, resp. bcnt will match.

MATCH EXTENSIONS

Ebtables extensions are dynamically loaded into the userspace tool, there is therefore no need to explicitly load them with a -m option like is done in iptables. These extensions deal with functionality supported by kernel modules supplemental to the core ebtables code.

802_3

Specify 802.3 DSAP/SSAP fields or SNAP type. The protocol must be specified as LENGTH (see the option -p above).

--802_3-sap [!] sap

DSAP and SSAP are two one byte 802.3 fields. The bytes are al? ways equal, so only one byte (hexadecimal) is needed as an argu? ment.

--802_3-type [!] type

If the 802.3 DSAP and SSAP values are 0xaa then the SNAP type field must be consulted to determine the payload protocol. This is a two byte (hexadecimal) argument. Only 802.3 frames with DSAP/SSAP 0xaa are checked for type.

among

Match a MAC address or MAC/IP address pair versus a list of MAC ad? dresses and MAC/IP address pairs. A list entry has the following for? mat: xx:xx:xx:xx:xx:xx[=ip.ip.ip.ip][,]. Multiple list entries are sep? arated by a comma, specifying an IP address corresponding to the MAC address is optional. Multiple MAC/IP address pairs with the same MAC address but different IP address (and vice versa) can be specified. If the MAC address doesn't match any entry from the list, the frame doesn't match the rule (unless "!" was used).

--among-dst [!] list

Compare the MAC destination to the given list. If the Ethernet

frame has type IPv4 or ARP, then comparison with MAC/IP destina?
tion address pairs from the list is possible.

--among-src [!] list

Compare the MAC source to the given list. If the Ethernet frame has type IPv4 or ARP, then comparison with MAC/IP source address pairs from the list is possible.

--among-dst-file [!] file

Same as --among-dst but the list is read in from the specified file.

--among-src-file [!] file

Same as --among-src but the list is read in from the specified file.

arp

Specify (R)ARP fields. The protocol must be specified as ARP or RARP.

--arp-opcode [!] opcode

The (R)ARP opcode (decimal or a string, for more details see ebtables -h arp).

--arp-htype [!] hardware type

The hardware type, this can be a decimal or the string Ethernet (which sets type to 1). Most (R)ARP packets have Eternet as hardware type.

--arp-ptype [!] protocol type

The protocol type for which the (r)arp is used (hexadecimal or the string IPv4, denoting 0x0800). Most (R)ARP packets have protocol type IPv4.

--arp-ip-src [!] address[/mask]

The (R)ARP IP source address specification.

--arp-ip-dst [!] address[/mask]

The (R)ARP IP destination address specification.

--arp-mac-src [!] address[/mask]

The (R)ARP MAC source address specification.

--arp-mac-dst [!] address[/mask]

The (R)ARP MAC destination address specification.

[!] --arp-gratuitous

Checks for ARP gratuitous packets: checks equality of IPv4 source address and IPv4 destination address inside the ARP header.

ip

Specify IPv4 fields. The protocol must be specified as IPv4.

--ip-source [!] address[/mask]

The source IP address. The flag --ip-src is an alias for this option.

--ip-destination [!] address[/mask]

The destination IP address. The flag --ip-dst is an alias for this option.

--ip-tos [!] tos

The IP type of service, in hexadecimal numbers. IPv4.

--ip-protocol [!] protocol

The IP protocol. The flag --ip-proto is an alias for this op? tion.

--ip-source-port [!] port1[:port2]

The source port or port range for the IP protocols 6 (TCP), 17 (UDP), 33 (DCCP) or 132 (SCTP). The --ip-protocol option must be specified as TCP, UDP, DCCP or SCTP. If port1 is omitted, 0:port2 is used; if port2 is omitted but a colon is specified, port1:65535 is used. The flag --ip-sport is an alias for this option.

--ip-destination-port [!] port1[:port2]

The destination port or port range for ip protocols 6 (TCP), 17 (UDP), 33 (DCCP) or 132 (SCTP). The --ip-protocol option must be specified as TCP, UDP, DCCP or SCTP. If port1 is omitted, 0:port2 is used; if port2 is omitted but a colon is specified, port1:65535 is used. The flag --ip-dport is an alias for this option.

ip6

Specify IPv6 fields. The protocol must be specified as IPv6.

--ip6-source [!] address[/mask]

The source IPv6 address. The flag --ip6-src is an alias for this option.

--ip6-destination [!] address[/mask]

The destination IPv6 address. The flag --ip6-dst is an alias for this option.

--ip6-tclass [!] tclass

The IPv6 traffic class, in hexadecimal numbers.

--ip6-protocol [!] protocol

The IP protocol. The flag --ip6-proto is an alias for this op? tion.

--ip6-source-port [!] port1[:port2]

The source port or port range for the IPv6 protocols 6 (TCP), 17 (UDP), 33 (DCCP) or 132 (SCTP). The --ip6-protocol option must be specified as TCP, UDP, DCCP or SCTP. If port1 is omitted, 0:port2 is used; if port2 is omitted but a colon is specified, port1:65535 is used. The flag --ip6-sport is an alias for this option.

--ip6-destination-port [!] port1[:port2]

The destination port or port range for IPv6 protocols 6 (TCP), 17 (UDP), 33 (DCCP) or 132 (SCTP). The --ip6-protocol option must be specified as TCP, UDP, DCCP or SCTP. If port1 is omit? ted, 0:port2 is used; if port2 is omitted but a colon is speci? fied, port1:65535 is used. The flag --ip6-dport is an alias for this option.

--ip6-icmp-type [!] {type[:type]/code[:code]|typename}

Specify ipv6-icmp type and code to match. Ranges for both type and code are supported. Type and code are separated by a slash. Valid numbers for type and range are 0 to 255. To match a sin? gle type including all valid codes, symbolic names can be used instead of numbers. The list of known type names is shown by the command

ebtables --help ip6

This option is only valid for --ip6-prococol ipv6-icmp.

limit

This module matches at a limited rate using a token bucket filter. A rule using this extension will match until this limit is reached. It can be used with the --log watcher to give limited logging, for exam? ple. Its use is the same as the limit match of iptables.

--limit [value]

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.

mark_m

--mark [!] [value][/mask]

Matches frames with the given unsigned mark value. If a value and mask are specified, the logical AND of the mark value of the frame and the user-specified mask is taken before comparing it with the user-specified mark value. When only a mark value is specified, the packet only matches when the mark value of the frame equals the user-specified mark value. If only a mask is specified, the logical AND of the mark value of the frame and the user-specified mask is taken and the frame matches when the result of this logical AND is non-zero. Only specifying a mask is useful to match multiple mark values.

pkttype

--pkttype-type [!] type

Matches on the Ethernet "class" of the frame, which is deter? mined by the generic networking code. Possible values: broadcast (MAC destination is the broadcast address), multicast (MAC des? tination is a multicast address), host (MAC destination is the receiving network device), or otherhost (none of the above).

stp

Specify stp BPDU (bridge protocol data unit) fields. The destination address (-d) must be specified as the bridge group address (BGA). For all options for which a range of values can be specified, it holds that if the lower bound is omitted (but the colon is not), then the lowest possible lower bound for that option is used, while if the upper bound is omitted (but the colon again is not), the highest possible upper bound for that option is used.

--stp-type [!] type

The BPDU type (0-255), recognized non-numerical types are con? fig, denoting a configuration BPDU (=0), and tcn, denothing a topology change notification BPDU (=128).

--stp-flags [!] flag

The BPDU flag (0-255), recognized non-numerical flags are topol? ogy-change, denoting the topology change flag (=1), and topol? ogy-change-ack, denoting the topology change acknowledgement flag (=128).

--stp-root-prio [!] [prio][:prio]

The root priority (0-65535) range.

--stp-root-addr [!] [address][/mask]

The root mac address, see the option -s for more details.

--stp-root-cost [!] [cost][:cost]

The root path cost (0-4294967295) range.

--stp-sender-prio [!] [prio][:prio]

The BPDU's sender priority (0-65535) range.

--stp-sender-addr [!] [address][/mask]

The BPDU's sender mac address, see the option -s for more de? tails.

--stp-port [!] [port][:port]

The port identifier (0-65535) range.

--stp-msg-age [!] [age][:age]

The message age timer (0-65535) range.

--stp-max-age [!] [age][:age]

The max age timer (0-65535) range.

--stp-hello-time [!] [time][:time]

The hello time timer (0-65535) range.

--stp-forward-delay [!] [delay][:delay]

The forward delay timer (0-65535) range.

vlan

Specify 802.1Q Tag Control Information fields. The protocol must be specified as 802_1Q (0x8100).

--vlan-id [!] id

The VLAN identifier field (VID). Decimal number from 0 to 4095.

--vlan-prio [!] prio

The user priority field, a decimal number from 0 to 7. The VID should be set to 0 ("null VID") or unspecified (in the latter case the VID is deliberately set to 0).

--vlan-encap [!] type

The encapsulated Ethernet frame type/length. Specified as a hexadecimal number from 0x0000 to 0xFFFF or as a symbolic name from /etc/ethertypes.

WATCHER EXTENSIONS

Watchers only look at frames passing by, they don't modify them nor de? cide to accept the frames or not. These watchers only see the frame if the frame matches the rule, and they see it before the target is exe? cuted.

log

The log watcher writes descriptive data about a frame to the syslog.

--log

Log with the default loggin options: log-level= info, log-pre? fix="", no ip logging, no arp logging.

--log-level level

Defines the logging level. For the possible values, see ebtables -h log. The default level is info.

--log-prefix text

Defines the prefix text to be printed at the beginning of the

line with the logging information.

--log-ip

Will log the ip information when a frame made by the ip protocol matches the rule. The default is no ip information logging.

--log-ip6

Will log the ipv6 information when a frame made by the ipv6 pro? tocol matches the rule. The default is no ipv6 information log? ging.

--log-arp

Will log the (r)arp information when a frame made by the (r)arp protocols matches the rule. The default is no (r)arp information logging.

nflog

The nflog watcher passes the packet to the loaded logging backend in order to log the packet. This is usually used in combination with nfnetlink_log as logging backend, which will multicast the packet through a netlink socket to the specified multicast group. One or more userspace processes may subscribe to the group to receive the packets.

--nflog

Log with the default logging options

--nflog-group nlgroup

The netlink group (1 - 2^32-1) to which packets are (only appli? cable for nfnetlink_log). The default value is 1.

--nflog-prefix prefix

A prefix string to include in the log message, up to 30 charac? ters long, useful for distinguishing messages in the logs.

--nflog-range size

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

--nflog-threshold size

Number of packets to queue inside the kernel before sending them to userspace (only applicable for nfnetlink_log). Higher values

result in less overhead per packet, but increase delay until the packets reach userspace. The default value is 1.

ulog

The ulog watcher passes the packet to a userspace logging daemon using netlink multicast sockets. This differs from the log watcher in the sense that the complete packet is sent to userspace instead of a de? scriptive text and that netlink multicast sockets are used instead of the syslog. This watcher enables parsing of packets with userspace programs, the physical bridge in and out ports are also included in the netlink messages. The ulog watcher module accepts 2 parameters when the module is loaded into the kernel (e.g. with modprobe): nlbufsiz specifies how big the buffer for each netlink multicast group is. If you say nlbufsiz=8192, for example, up to eight kB of packets will get accumulated in the kernel until they are sent to userspace. It is not possible to allocate more than 128kB. Please also keep in mind that this buffer size is allocated for each nlgroup you are using, so the total kernel memory usage increases by that factor. The default is 4096. flushtimeout specifies after how many hundredths of a second the queue should be flushed, even if it is not full yet. The default is 10 (one tenth of a second).

--ulog

Use the default settings: ulog-prefix="", ulog-nlgroup=1, ulog-cprange=4096, ulog-qthreshold=1.

--ulog-prefix text

Defines the prefix included with the packets sent to userspace.

--ulog-nlgroup group

Defines which netlink group number to use (a number from 1 to 32). Make sure the netlink group numbers used for the iptables ULOG target differ from those used for the ebtables ulog watcher. The default group number is 1.

--ulog-cprange range

Defines the maximum copy range to userspace, for packets match? ing the rule. The default range is 0, which means the maximum

copy range is given by nlbufsiz. A maximum copy range larger than 128*1024 is meaningless as the packets sent to userspace have an upper size limit of 128*1024.

--ulog-qthreshold threshold

Queue at most threshold number of packets before sending them to userspace with a netlink socket. Note that packets can be sent to userspace before the queue is full, this happens when the ulog kernel timer goes off (the frequency of this timer depends on flushtimeout).

TARGET EXTENSIONS

arpreply

The arpreply target can be used in the PREROUTING chain of the nat ta? ble. If this target sees an ARP request it will automatically reply with an ARP reply. The used MAC address for the reply can be specified. The protocol must be specified as ARP. When the ARP message is not an ARP request or when the ARP request isn't for an IP address on an Eth? ernet network, it is ignored by this target (CONTINUE). When the ARP request is malformed, it is dropped (DROP).

--arpreply-mac address

Specifies the MAC address to reply with: the Ethernet source MAC and the ARP payload source MAC will be filled in with this ad? dress.

--arpreply-target target

Specifies the standard target. After sending the ARP reply, the rule still has to give a standard target so ebtables knows what to do with the ARP request. The default target is DROP.

dnat

The dnat target can only be used in the PREROUTING and OUTPUT chains of the nat table. It specifies that the destination MAC address has to be changed.

--to-destination address

Change the destination MAC address to the specified address. The flag --to-dst is an alias for this option.

--dnat-target target

Specifies the standard target. After doing the dnat, the rule still has to give a standard target so ebtables knows what to do with the dnated frame. The default target is ACCEPT. Making it CONTINUE could let you use multiple target extensions on the same frame. Making it DROP only makes sense in the BROUTING chain but using the redirect target is more logical there. RE? TURN is also allowed. Note that using RETURN in a base chain is not allowed (for obvious reasons).

mark

The mark target can be used in every chain of every table. It is possi? ble to use the marking of a frame/packet in both ebtables and iptables, if the bridge-nf code is compiled into the kernel. Both put the marking at the same place. This allows for a form of communication between ebtables and iptables.

--mark-set value

Mark the frame with the specified non-negative value.

--mark-or value

Or the frame with the specified non-negative value.

--mark-and value

And the frame with the specified non-negative value.

--mark-xor value

Xor the frame with the specified non-negative value.

--mark-target target

Specifies the standard target. After marking the frame, the rule still has to give a standard target so ebtables knows what to do. The default target is ACCEPT. Making it CONTINUE can let you do other things with the frame in subsequent rules of the chain.

redirect

The redirect target will change the MAC target address to that of the bridge device the frame arrived on. This target can only be used in the PREROUTING chain of the nat table. The MAC address of the bridge is

used as destination address."

--redirect-target target

Specifies the standard target. After doing the MAC redirect, the rule still has to give a standard target so ebtables knows what to do. The default target is ACCEPT. Making it CONTINUE could let you use multiple target extensions on the same frame. Making it DROP in the BROUTING chain will let the frames be routed. RE? TURN is also allowed. Note that using RETURN in a base chain is not allowed.

snat

The snat target can only be used in the POSTROUTING chain of the nat table. It specifies that the source MAC address has to be changed.

--to-source address

Changes the source MAC address to the specified address. The flag --to-src is an alias for this option.

--snat-target target

Specifies the standard target. After doing the snat, the rule still has to give a standard target so ebtables knows what to do. The default target is ACCEPT. Making it CONTINUE could let you use multiple target extensions on the same frame. Making it DROP doesn't make sense, but you could do that too. RETURN is also allowed. Note that using RETURN in a base chain is not al? lowed.

--snat-arp

Also change the hardware source address inside the arp header if the packet is an arp message and the hardware address length in the arp header is 6 bytes.

FILES

/etc/ethertypes

MAILINGLISTS

See http://netfilter.org/mailinglists.html

BUGS

The version of ebtables this man page ships with does not support the

broute table. Also there is no support for string match. Further, sup?

port for atomic-options (--atomic-file, --atomic-init, --atomic-save,

--atomic-commit) has not been implemented, although ebtables-save and

ebtables-restore might replace them entirely given the inherent atomic?

ity of nftables. Finally, this list is probably not complete.

SEE ALSO

xtables-nft(8), iptables(8), ip(8), nft(8)

See https://wiki.nftables.org

December 2011 EBTABLES(8)