NAME

ebtables (v.2.0) - Ethernet bridge frame table administration

SYNOPSIS

ebtables [**-t table**] -[**ADI**] **chain rule-specification** [**match-extensions**] [**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**] [**--Lc**]] | [**--Lx**]] [**--Lmac2**]
ebtables [**-t table**] **-N chain**
ebtables [**-t table**] **-X** [**chain**]
ebtables [**-t table**] **-E old-chain-name new-chain-name**
ebtables [**-t table**] **--init-table**
ebtables [**-t table**] [**--atomic-file file**] **--atomic-commit**
ebtables [**-t table**] [**--atomic-file file**] **--atomic-init**
ebtables [**-t table**] [**--atomic-file file**] **--atomic-save**

DESCRIPTION

ebtables is a user space tool, it is used to set up and maintain the tables of Ethernet frame rules in the Linux kernel. These rules inspect the Ethernet frames which they see. **ebtables** is analogous to the **iptables** user space tool, but **ebtables** is less complicated.

CHAINS

There are three Ethernet frame tables with built-in chains in the Linux kernel. The kernel 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 Ethernet frame, then a processing specification tells what to do with that matching frame. The processing specification is called a 'target'. However, 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 create new (user-defined) chains which can be used as the 'target' of a rule.

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 target. The target can be one of these values: *ACCEPT*, *DROP*, *CONTINUE*, *RETURN*, an 'extension' (see below) or 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 to know how many frames pass a certain point in the chain or to log those frames. *RETURN* means stop traversing this chain and resume at the next rule in the previous (calling) chain. For the extension targets please see the **TARGET EXTENSIONS** section of this man page.

TABLES

As stated earlier, there are three Ethernet frame tables in the Linux kernel. The tables are **filter**, **nat** and **broute**. Of these three tables, the filter table is the default table that the **ebtables** command operates on. If you are working with the filter table, then you can drop the '-t filter' argument to the ebtables command. However, you will need to provide the -t argument for the other two tables. 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), **OUTPUT** (for locally-generated frames) and **FORWARD** (for frames being bridged).

nat, is 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 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 POSTROUTING and PREROUTING: 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.

broute, is used to make a brouter, it has one built-in chain: **BROUTING**. The targets **DROP** and **ACCEPT** have special meaning in the broute table. **DROP** actually means the frame has to be routed, while **ACCEPT** means the frame has to be bridged. The **BROUTING** chain is traversed very early. It is only traversed by frames entering on a bridge enslaved NIC that is in forwarding state. Normally those frames would be bridged, but you can decide otherwise here. The **redirect** target is very handy here.

EBTABLES COMMAND LINE ARGUMENTS

After the initial ebtables -t, table command line argument, the remaining arguments can be divided into several different groups. These groups are commands, miscellaneous commands, rule-specifications, match-extensions, and watcher-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. With the exception of both the **-Z** and **--atomic-file** commands, only one command may be used on the command line at a time.

-A, --append

Append a rule to the end of the selected chain.

-D, --delete

Delete the specified rule from the selected chain. There are two ways to use this command. The first is by specifying an interval of rule numbers to delete, syntax: start_nr[:end_nr]. Using negative numbers is allowed, for more details about using negative numbers, see the -I command. The second usage is by specifying the complete rule as it would have been specified when it was added.

-I, --insert

Insert the specified rule into the selected chain at the specified rule number. 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 number 0 specifies the place past the last rule in the chain and using this number is therefore equivalent with using the -A command.

-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 the chain does not change the policy 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 three options change the output of the **-L** list command:

--Ln
Places the rule number in front of every rule.

--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.

--Lx

The output of the **--Lx** option may be used to create a set of **ebtables** commands. You may use this set of commands in an **ebtables** boot or reload script. For example the output could be used at system startup. The **--Lx** option is incompatible with both of the other **--Ln** and **--Lc** chain listing options.

--Lmac2
Shows all MAC addresses with the same length, adding leading zeroes if necessary. The default representation omits zeroes in the addresses when they are not needed.

All necessary **ebtables** commands for making the current list of user-defined chains in the kernel and any commands issued by the user to rename the standard **ebtables** chains will be listed, when no chain name is supplied for the **-L** command while using the **--Lx** option.

-N, --new-chain

Create a new user-defined chain with the given name. The number of user-defined chains is unlimited. A user-defined chain name has maximum length of 31 characters.

-X, --delete-chain

Delete the specified user-defined chain. There must be no remaining references 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 may rename a standard chain name to a name that suits your taste. For example, if you like PREBRIDGING more than PREROUTING, then you can use the -E command to rename the PREROUTING chain. If you do rename one of the standard **ebtables** 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 **ebtables** chain in this fashion has no effect on the structure or function of the **ebtables** kernel table.

--init-table

Replace the current table data by the initial table data.

--atomic-init

Copy the kernel's initial data of the table to the specified file. This can be used as the first action, after which rules are added to the file. The file can be specified using the **--atomic-file** command or through the *EBTABLES_ATOMIC_FILE* environment variable.

--atomic-save

Copy the kernel's current data of the table to the specified file. This can be used as the first action, after which rules are added to the file. The file can be specified using the **--atomic-file** command or through the *EBTABLES_ATOMIC_FILE* environment variable.

--atomic-commit

Replace the kernel table data with the data contained in the specified file. This is a useful command that allows you to load all your rules of a certain table into the kernel at once, saving the kernel a lot of precious time and allowing atomic updates of the tables. The file which contains the table data is

constructed by using either the **--atomic-init** or the **--atomic-save** command to generate a starting file. After that, using the **--atomic-file** command when constructing rules or setting the *EBTABLES_ATOMIC_FILE* environment variable allows you to extend the file and build the complete table before committing it to the kernel.

--atomic-file -Z

The counters stored in a file with, say, **--atomic-init** can be optionally zeroed by supplying the **-Z** command. You may also zero the counters by setting the *EBTABLES_ATOMIC_FILE* environment variable.

MISCELLANOUS COMMANDS

-V, --version

Show the version of the ebtables user space program.

-h, --help

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: **ACCEPT**, **DROP**, **CONTINUE**, **RETURN**, a target extension (see **TARGET EXTENSIONS**) or a user-defined chain name.

--atomic-file file

Let the command operate on the specified file. The data of the table to operate on will be extracted from the file and the result of the operation will be saved back into the file. If specified, this option should come before the command specification. An alternative that should be preferred, is setting the *EBTABLES_ATOMIC_FILE* environment variable.

-M, --modprobe program

When talking to the kernel, use this program to try to automatically load missing kernel modules.

RULE-SPECIFICATIONS

The following command line arguments make up a rule specification (as used in the add and delete commands). A "!" option before the specification inverts the test for that specification. Apart from these standard rule specifications there are some other command line arguments of interest. See both the **MATCH-EXTENSIONS** and the **WATCHER-EXTENSION(S)** below.

-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 protocol 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 via which a frame is received (for the **INPUT**, **FORWARD**, **PREROUTING** and **BROUTING** chains). The flag **--in-if** is an alias for this option.

--logical-in [!] *name*

The (logical) bridge interface via which a frame is received (for the **INPUT**, **FORWARD**, **PREROUTING** and **BROUTING** chains).

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

The interface via which a frame is going to be sent (for the **OUTPUT**, **FORWARD** and **POSTROUTING** chains). 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 (for the **OUTPUT**, **FORWARD** and **POSTROUTING** chains).

-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 Address). **Unicast** = 00:00:00:00:00:00/01:00:00:00:00:00, **Multicast** = 01:00:00:00:00:00/01:00:00:00:00:00, **Broadcast** = 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. The flag **--dst** is an alias for this option.

MATCH-EXTENSIONS

ebtables extensions are precompiled into the userspace tool. So there is no need to explicitly load them with a -m option like in **iptables**. However, these extensions deal with functionality supported by supplemental kernel modules.

802.3

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

--802_3-sap [!] *sap*

DSAP and SSAP are two one byte 802.3 fields. The bytes are always equal, so only one byte (hexadecimal) is needed as an argument.

--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 addresses and MAC/IP address pairs. A list entry has the following format: xx:xx:xx:xx:xx:xx[=ip.ip.ip.ip][,]. Multiple list entries are separated 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 destination 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.

arp

Specify 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". This is normally Ethernet (value 1).

--arp-ptype [!] *protocol type*

The protocol type for which the (r)arp is used (hexadecimal or the string "IPv4"). This is normally IPv4 (0x0800).

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

The ARP IP source address specification.

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

The ARP IP destination address specification.

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

The ARP MAC source address specification.

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

The ARP MAC destination address specification.

ip

Specify ip 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 option.

--ip-source-port [!] *port*[:*port*]

The source port or port range for the ip protocols 6 (TCP) and 17 (UDP). If the first port is omitted, "0" is assumed; if the last is omitted, "65535" is assumed. The flag **--ip-sport** is an alias for this option.

--ip-destination-port [!] *port*[:*port*]

The destination port or port range for ip protocols 6 (TCP) and 17 (UDP). The flag **--ip-dport** is an alias for this option.

limit

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 watcher to give limited logging, for example. The usage/implementation is completely similar to that of the iptables limit match.

--limit *rate*

Maximum average matching rate: specified as a number, with an optional

--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 mark value and mask is 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. If only a mask is specified (start with '/') the logical AND of the mark value of the frame and the user-specified mark is taken and the result is compared with zero.

pkttype

--pkttype-type [!] *type*

Matches on the Ethernet "class" of the frame, which is determined by the generic networking code. Possible values: broadcast (MAC destination is broadcast address), multicast (MAC destination is multicast address), host (MAC destination is the receiving network device) or other host (none of the above).

stp

Specify stp BPDU (bridge protocol data unit) fields. The destination address must be specified as the bridge group address (BGA).

--stp-type [!] *type*

The BPDU type (0-255), special recognized types: **config**: configuration BPDU (=0) and **tcn**: topology change notification BPDU (=128).

--stp-flags [!] *flag*

The BPDU flag (0-255), special recognized flags: **topology-change**: the topology change flag (=1) **topology-change-ack**: 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 details.

--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. Decimal number from 0 to 7. The VID should be set to 0 ("null VID") or unspecified (for this case the VID is deliberately set to 0).

--vlan-encap [!] *type*

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

WATCHER-EXTENSION(S)

Watchers are things that only look at frames passing by. These watchers only see the frame if the frame matches the rule.

log

The fact that the log module is a watcher lets us log stuff while giving a target by choice. Note that the log module therefore is not a target.

--log

Use this if you won't specify any other log options, so if you want to use the default settings: log-prefix="", no arp logging, no ip logging, log-level=info.

--log-level *level*

defines the logging level. For the possible values: ebtables -h log. The default level is *info*.

--log-prefix *text*

defines the prefix to be printed before 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-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.

TARGET EXTENSIONS

arpreply

The **arpreply** target can be used in the **PREROUTING** chain of the **nat** table. 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. When the arp message is not an arp request, it is ignored by this target.

--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 address.

--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. The default target is DROP.

dnat

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

--to-destination *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. 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. RETURN is also allowed. Note that using RETURN in a base chain is not allowed.

mark

The mark target can be used in every chain of every table. It is possible to use the marking of a frame/packet in both ebtables and iptables, if the br-nf code is compiled into the kernel. Both put the marking at the same place. So, you can consider this fact as a feature, or as something to watch out for.

--set-mark *value*

Mark the frame with the specified unsigned 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 other 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 **BROUTING** chain of the **broute** table and the **PREROUTING** chain of the **nat** table.

--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. RETURN 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*

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 allowed.

Time: 21:18:29 GMT, November 09, 2003