

**NAME**

arptables – ARP table administration (nft-based)

**SYNOPSIS**

```
arptables [-t table] [-[AD] chain rule-specification [options]
arptables [-t table] [-[RI] chain rulenum rule-specification [options]
arptables [-t table] [-D chain rulenum [options]
arptables [-t table] [-[LFZ] [chain] [options]
arptables [-t table] [-[NX] chain
arptables [-t table] [-E old-chain-name new-chain-name
arptables [-t table] [-P chain target [options]
```

**DESCRIPTION**

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

**CHAINS**

The kernel table is 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 ARP frames. If a rule matches an ARP 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 ARP 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**

There is only one ARP table in the Linux kernel. The table is **filter**. You can drop the '-t filter' argument to the arptables command. The -t argument must be the first argument on the arptables command line, if used.

**-t, --table**

**filter**, is the only table and contains two built-in chains: **INPUT** (for frames destined for the host) and **OUTPUT** (for locally-generated frames).

**ARPTABLES COMMAND LINE ARGUMENTS**

After the initial arptables 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 arptables 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 the **-Z** command, 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.

**-R, --replace**

Replaces 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 1 and *N*. *i* specifies the place in the chain where the rule should be replaced.

**-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.

**-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 **arptables** 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 **arptables** chain names, please be sure to mention this fact should you post a question on the **arptables** mailing lists. It would be wise to use the standard name in your post. Renaming a standard **arptables** chain in this fashion has no effect on the structure or function of the **arptables** kernel table.

**MISCELLANEOUS COMMANDS****-V, --version**

Show the version of the **arptables** userspace program.

**-h, --help**

Give a brief description of the command syntax.

**-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.

**-c, --set-counters** *PKTS BYTES*

This enables the administrator to initialize the packet and byte counters of a rule (during **INSERT**, **APPEND**, **REPLACE** operations).

**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.

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

The Source IP specification.

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

The Destination IP specification.

**--source-mac** [!] *address[/mask]*

The source mac address. Both mask and address are written as 6 hexadecimal numbers separated by colons.

**--destination-mac** [!] *address[/mask]*

The destination mac address. Both mask and address are written as 6 hexadecimal numbers separated by colons.

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

The interface via which a frame is received (for the **INPUT** chain). The flag **--in-if** is an alias for this option.

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

The interface via which a frame is going to be sent (for the **OUTPUT** chain). The flag **--out-if** is an alias for this option.

**-l, --h-length** *length[/mask]*

The hardware length (nr of bytes)

**--opcode** *code[/mask]*

The operation code (2 bytes). Available values are: **1=Request 2=Reply 3=Request\_Reverse 4=Reply\_Reverse 5=DRARP\_Request 6=DRARP\_Reply 7=DRARP\_Error 8=InARP\_Request 9=ARP\_NAK**.

**--h-type** *type[/mask]*

The hardware type (2 bytes, hexadecimal). Available values are: **1=Ethernet**.

**--proto-type** *type[/mask]*

The protocol type (2 bytes). Available values are: **0x800=IPv4**.

**TARGET-EXTENSIONS**

**arptables** 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.

**mangle****--mangle-ip-s** **IP address**

Mangles Source IP Address to given value.

**--mangle-ip-d IP address**

Mangles Destination IP Address to given value.

**--mangle-mac-s MAC address**

Mangles Source MAC Address to given value.

**--mangle-mac-d MAC address**

Mangles Destination MAC Address to given value.

**--mangle-target target**

Target of ARP mangle operation (**DROP**, **CONTINUE** or **ACCEPT** -- default is **ACCEPT**).

**CLASSIFY**

This module allows you to set the `skb->priority` value (and thus classify the packet into a specific CBQ class).

**--set-class major:minor**

Set the major and minor class value. The values are always interpreted as hexadecimal even if no 0x prefix is given.

**MARK**

This module allows you to set the `skb->mark` value (and thus classify the packet by the mark in u32)

**--set-mark mark**

Set the mark value. The values are always interpreted as hexadecimal even if no 0x prefix is given

**--and-mark mark**

Binary AND the mark with bits.

**--or-mark mark**

Binary OR the mark with bits.

**NOTES**

In this nft-based version of **arptables**, support for **FORWARD** chain has not been implemented. Since ARP packets are "forwarded" only by Linux bridges, the same may be achieved using **FORWARD** chain in **ebtables**.

**MAILINGLISTS**

See <http://netfilter.org/maillinglists.html>

**SEE ALSO**

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

See <https://wiki.nftables.org>