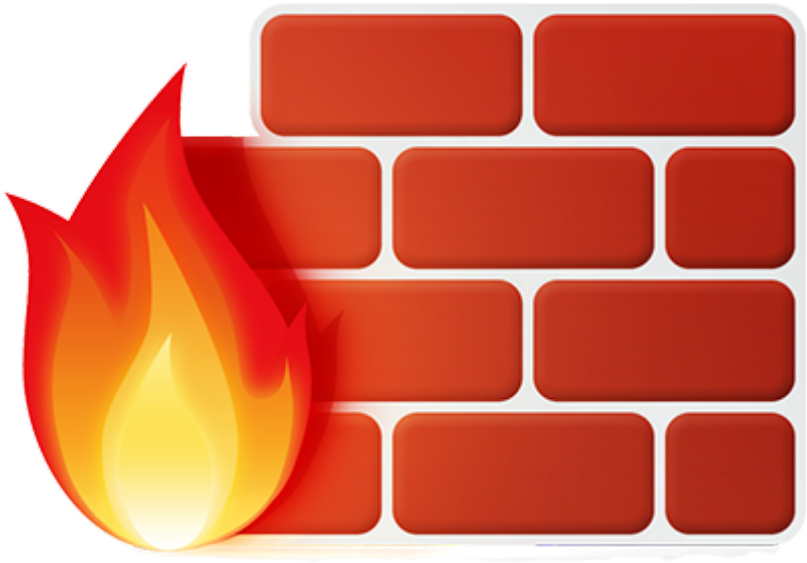
May 18, 2015

iptables (IPv4)



Notes on `iptables` including the most used command types and an example of some basic filter table rules.

> **Note:** Only IP version 4 examples are covered in this post.

Tables

The tables that are present in the system depends on the kernel configuration options and which modules are enabled.

The default table used is the filter table. `-t` specifies the table to be used when available and needed.

Filter Table

This is the first table and most commonly configured table.

It's comprised of the following chains:

- **Input** – Intended for incoming packets accessing local sockets.
- **Forward** – For packets that get routed through the server.
- **Output** – Is for locally-generated out-going packets.

NAT Table

When a packet forms a new connection this is the table that is consulted.

- **Prerouting** – Alters a packet as soon as it comes in.

- **Output** – Changes locally-generated packets before they are routed.
- **Postrouting** – Alters packets as they are about to be sent out.

Mangle Table

This table is made use of for specialised packet alteration.

- **Prerouting** – Alters a packet as soon as it comes in.
- **Output** – Changes locally-generated packets before they are routed.
- **Input** – Packets entering the host itself.
- **Forward** – Changes locally-generated packets before they are routed.
- **Postrouting** – Alters packets as they are about to be sent out.

Raw Table

Used for setting up exemptions from connection tracking in combination with the *NOTRACK* target.

- **Prerouting** – Deals with packets arriving via any network interface.
- **Output** – Changes locally-generated packets before they are routed.

Security Table

This table is used for Mandatory Access Control (MAC) networking rules, such as those enabled by the *SECMARK* and *CONNSECMARK* targets. It is called after the "Filter Table".

- **Input** – Intended for incoming packets accessing local sockets.
- **Output** – Is for locally-generated out-going packets.
- **Forward** – For packets that get routed through the server.

Viewing Rules

You can view the current rules in their respective tables with the `-L` parameter.

```
1    $ sudo iptables -L
```

Adding the `-v` parameter (verbose) provides additional information for each rule.

```
1    $ sudo iptables -L -v
```

Prints line numbers for each rule and `-n` prints IP addresses and ports where possible.

```
1    $ sudo iptables -vnL --line-numbers
```

Prints all rules in the selected chain (`INPUT`), all chain's rules are printed if none are passed.

```
1    $ sudo iptables -S INPUT
```

Check whether a rule matching the provided details exists or not with `-C` .

`-C` uses the same logic as the delete parameter to find matching entries, except it does not write or alter any rules.

```
1    $ sudo iptables -C INPUT -p tcp --dport 80 -j ACCEPT
```

Adding Rules

To *append* a rule to the end of a set of rules, use the `-A` parameter followed by the chain name and port details.

```
1    $ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

To instead *insert* a rule anywhere amongst a set of rules use the `-I` parameter with the chain name and port details.

```
1    $ sudo iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT
```

To *match* or add a comment alongside a rule append the following to the end of a rule entry:

```
1    -m comment --comment "this is an example comment - comment goes here"
```

Removing Rules

Delete a rule by line number with the `-D` parameter, then chain name, then number that you wish to delete.

```
1    $ sudo iptables -D INPUT 10
```

Deleting based off of what a rule is doing is also possible, again with `-D` and then the pre-existing port and rule details.

```
1    $ sudo iptables -D INPUT -s 127.0.0.1 -p tcp --dport 111 -j ACCEPT
```

To remove the entire active configuration you can use the `-F` parameter, which *flushes* all rules and tables.

```
```

```
1    $ sudo iptables -F
```

Adding a chain name deletes that chain, like so:

```
1    $ sudo iptables -F OUTPUT
```

Saving Rules

Rules created and inserted are lost upon shutting down the system. There a few different ways of saving the rules and loading them back in again.

Here's one of them.

Debian / Ubuntu

The package `iptables-persistent` once installed automatically loads rules saved in the `/etc/iptables.rules.v4` file.

```
```

```
$ sudo apt-get install iptables-persistent
$ sudo service iptables-persistent start
```

To save your rules in the above file run:

```
$ sudo iptables-save > /etc/iptables/rules.v4
```

You can confirm their entry after with:

```
$ sudo less /etc/iptables/rules.v4
```

These can also be loaded directly into your iptables firewall at any time using:

```
$ sudo iptables-restore < /etc/iptables/rules.v4
```

Standard Filter Table Rules

Local-Host / Loop-back Address Example

```
1   $ sudo iptables -I INPUT 1 -i lo -j ACCEPT -m comment --comment "allow input on localhost"
2   $ sudo iptables -I OUTPUT 1 -o lo -j ACCEPT -m comment --comment "allow output on localhost"
```

Maintain Current Connections

```
1   $ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT -m comment --cor
```

Drop All Remaining Incoming Packets

```
1   $ sudo iptables -A INPUT -j DROP -m comment --comment "Drop All Remaining Packets"
```

Other Standard Rules

These are in the format of a configuration script that you would load into `iptables`. So are missing the usual command line prefix.

Allows all loopback (`lo0`) traffic and drop all traffic to `127/8` that doesn't use `lo0`.

```
1    -A INPUT -i lo -j ACCEPT
2    -A INPUT ! -i lo -d 127.0.0.0/8 -j REJECT
```

Accepts all established inbound connections.

```
1    -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Allows all outbound traffic, you could modify this to only allow only certain traffic.

```
```

```
1    -A OUTPUT -j ACCEPT
```

Allows HTTP and HTTPS connections from anywhere (the normal ports for websites).

```
1    -A INPUT -p tcp --dport 80 -j ACCEPT
2    -A INPUT -p tcp --dport 443 -j ACCEPT
```

Allows SSH connections.

The `--dport` number is the same as in `/etc/ssh/sshd_config` .

```
1    -A INPUT -p tcp -m state --state NEW --dport 22 -j ACCEPT
```

> **Note:** The default SSH port is often changed from 22 to avoid unwanted activity. Or even better only allow access from certain IP's and use SSH keys.

Allows ping with the next code snippet, be aware that blocking other **types** of ICMP packets is considered a bad idea by some. Hence the `icmp-type 8`.

```
1    -A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
```

Reject all other inbound packets, aka deny by default unless explicitly allowed as part of other policies:

```
1    -A INPUT -j REJECT
2    -A FORWARD -j REJECT
```

Sources

- How To Set Up a Firewall Using iptables on Ubuntu 14.04

- Major.io iptables Posts
- Debian WIki - iptables

– Scarlz: @5car1z

#Firewall #Networking #IP COMMENTS

ALSO ON TRICKS OF THE TRADES

Ansible - Inventory Concepts (2) - Tricks of the Trades

2 comments • 3 years ago

Avatar Sanjay Upadhyay — nice and lucid explanation, thanks!

Ansible - Installing and Running (1)

4 comments • 3 years ago

Avatar C.J. Scarlett — Hi Pulkit,I hope you fixed this in the end but if not I ran into a similar problem after running through the post above today. I think it's due to updates and changes to

Docker - Data Volumes and Data Containers (4)

17 comments • 3 years ago

Avatar damnmaxims — So you are packing the absolute path folders in the tar archive as well, I am surprised these errors still remain in this article, the correct command for packing is

Debian 8 (Jessie) VPS Basic Checklist

2 comments • 3 years ago

Avatar C.J. Scarlett — Glad to hear this helped with the outcome, thanks!

Subscribe Add Disqus to your site Disqus' Privacy Policy DISQUS