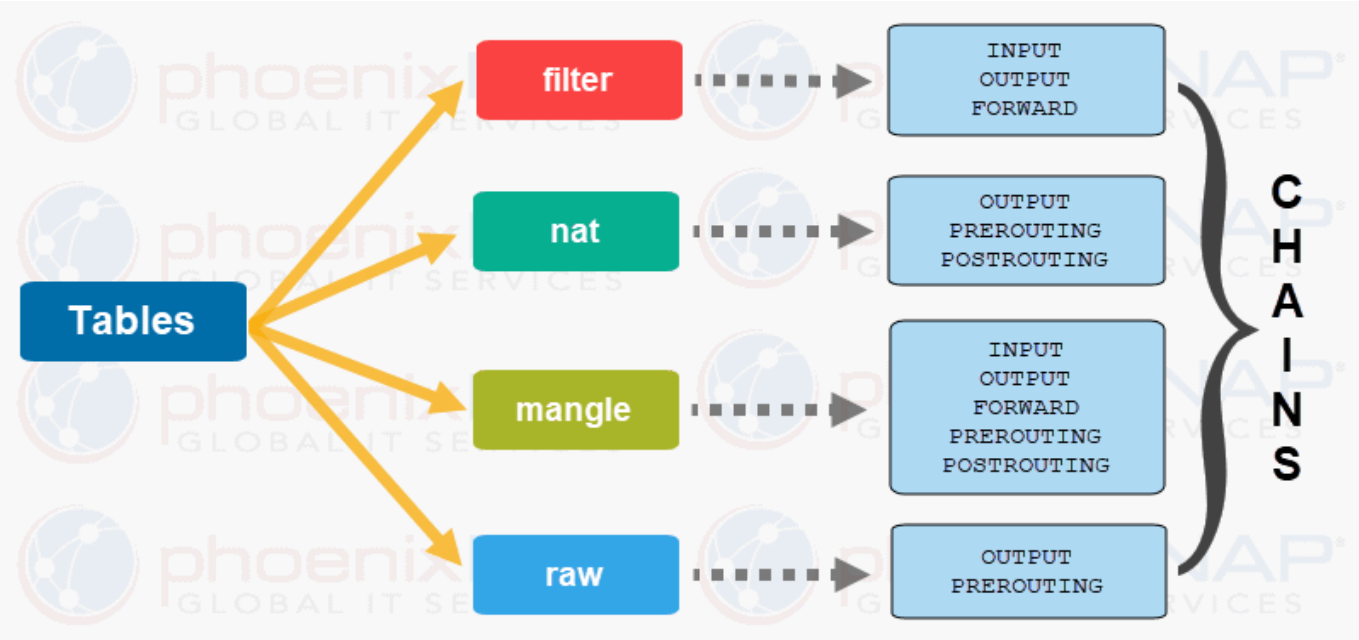
Iptables Linux



Commands

- Show actual config `iptables -nvL`
- Change polity `iptables -P <INPUT, OUTPUT, FORWARD> <DROP, ACCEPT, RETURN>`

ICMP

Allow ICMP from WAN interface

```
iptables -A INPUT -p icmp -j ACCEPT
```

Allow ICMP between LANs

```
iptables -t nat POSTROUTING -p icmp -o <OUTPUT interface> -s <source network IP
192.168.40.0/24> -d <destination network IP 192.168.30.0/24> -j MASQUERADE
```

NAT

Activate IP-forwarding in the kernel uncommenting the following line in `/etc/sysctl.conf`

```
net.ipv4.ip_forward=1
```

Make the nat rules:

```
iptables -t nat -A POSTROUTING -o ens18 -j MASQUERADE
iptables -A FORWARD -m state --state ESTABLISH,RELATED,NEW -f ACCEPT
```

Enable services on DMZ or LAN to be accesed from WAN

HTTP

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination
192.168.202.105:80
iptables -t nat -A POSTROUTING -p tcp -d 192.168.202.105 --dport 80 -j MASQUERADE
```

Defining Chain Rules

Defining a rule means appending it to the chain. To do this, you need to insert the -A option **(Append)** right after the iptables command, like so:

```
iptables -A
```

It will alert iptables that you are adding new rules to a chain. Then, you can combine the command with other options, such as:

- **i (interface)** — the network interface whose traffic you want to filter, such as eth0, lo, ppp0, etc.
- **p (protocol)** — the network protocol where your filtering process takes place. It can be either **tcp, udp, udplite, icmp, sctp, icmpv6,** and so on. Alternatively, you can type all to choose every protocol.
- **s (source)** — the address from which traffic comes from. You can add a hostname or IP address. – dport (destination port) — the destination port number of a protocol, such as 22 (SSH), 443 (https), etc.
- **j (target)** — the target name **(ACCEPT, DROP, RETURN)**. You need to insert this every time you make a new rule.

If you want to use all of them, you must write the command in this order:

```
iptables -A <chain> -i <interface> -p <protocol (tcp/udp) > -s <source> --dport <port
no.>  -j <target>
```

Once you understand the basic syntax, you can start configuring the firewall to give more security to your server. For this iptables tutorial, we are going to use the INPUT chain as an example.

Enabling Traffic on Localhost

To allow traffic on localhost, type this command:

```
iptables -A INPUT -i lo -j ACCEPT
```

For this iptables tutorial, we use lo or loopback interface. It is utilized for all communications on the localhost. The command above will make sure that the connections between a database and a web application on the same machine are working properly.

Enabling Connections on HTTP, SSH, and SSL Port

Next, we want http (port 80), https (port 443), and ssh (port 22) connections to work as usual. To do this, we need to specify the protocol (-p) and the corresponding port (–dport). You can execute these commands one by one:

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Filtering Packets Based on Source

Iptables allows you to filter packets based on an IP address or a range of IP addresses. You need to specify it after the -s option. For example, to accept packets from 192.168.1.3, the command would be:

```
iptables -A INPUT -s 192.168.1.3 -j ACCEPT
```

If you want to drop packets from a range of IP addresses, you have to use the -m option and iprange module. Then, specify the IP address range with –src-range. Remember, a hyphen

should separate the range of ip addresses without space, like this:

```
iptables -A INPUT -m iprange --src-range 192.168.1.100-192.168.1.200 -j DROP
```

Allow internet Access when filter chain is on DROP

Create the following rules to allow access to internet:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state NEW,ESTABLISHED -j ACCEPT
```

Deleting Rules

If you want to remove all rules and start with a clean slate, you can use the **-F** option **(flush)**:

```
iptables -F
```

This command erases all current rules. However, to delete a specific rule, you must use the -D option. First, you need to see all the available rules by entering the following command:

```
iptables -L --line-numbers
```

List rules of other chain:

```
iptables -t nat -nL --line
```

You will get a list of rules with numbers:

```
Chain INPUT (policy ACCEPT)

num  target     prot opt source               destination

1    ACCEPT     all -- 192.168.0.4          anywhere
2    ACCEPT     tcp -- anywhere             anywhere tcp dpt:https
3    ACCEPT     tcp -- anywhere             anywhere tcp dpt:http
4    ACCEPT     tcp -- anywhere             anywhere tcp dpt:ssh
```

To delete a rule, insert the corresponding chain and the number from the list. Let's say for this iptables tutorial, we want to get rid of rule number three of the INPUT chain. The command should be:

```
iptables -D INPUT 3
```

Delete rule from anther chain:

```
iptables -t nat -D POSTROUTING 1
```

Persisting Changes

Install the next package to save the active rules and persist the rules after reboot

```
apt install iptables-persistent
```

To show the active rules:

```
/sbin/iptables-save
```

Save the rules:

```
/sbin/iptables-save > /etc/iptables/rules.v4
```