Firewall Setup

Firewalls provide a security mechanism to control and monitor network traffic between different network segments, such as internal and external networks or network zones. They protect networks from unauthorized access, malicious traffic, and other security threats.

Linux offers built-in firewall capabilities to filter incoming and outgoing traffic based on predefined rules, protocols, ports, and other criteria. The goal of firewall implementation depends on organizational needs, ensuring confidentiality, integrity, and availability of resources.

History of Linux Firewalls

- **iptables** replaced older tools like `ipchains` and `ipfwadm` around 2000 with Linux 2.4 kernel.
- iptables offers a flexible command-line interface for filtering traffic by IP, port, protocol, etc.
- It helps defend against DoS attacks, port scans, and intrusions.
- Linux firewall functionality is implemented through the **Netfilter** kernel framework, with iptables used for rule configuration.

Firewall Tools in Linux

Tool	Description
iptables	Flexible, powerful rule-based firewall utility.
nftables	Modern replacement for iptables with improved syntax and performance, but not backward compatible.
UFW	"Uncomplicated Firewall" — simpler, user-friendly interface built on iptables.
FirewallD	Dynamic firewall manager supporting zones and complex rule sets.

---

iptables Components

Component	Description
Tables	Organize and categorize firewall rules.
Chains	Group rules applied to specific network traffic types.
Rules	Define matching criteria and actions for traffic.
Matches	Specify matching criteria (IP, port, protocol, etc.).
Targets	Define the action to perform (ACCEPT, DROP, REJECT, etc.)

Tables in iptables

Table Name	Purpose	Built-in Chains
filter	Filter network traffic (IP, ports, protocols)	INPUT, OUTPUT, FORWARD
nat	Modify source/destination IP addresses (NAT)	PREROUTING, POSTROUTING
mangle	Modify packet header fields	PREROUTING, OUTPUT, INPUT, FORWARD, POSTROUTING
raw	Special packet processing options	PREROUTING, OUTPUT

Chains in iptables

- **Built-in Chains:** Automatically created with tables.
  - *filter*: INPUT, OUTPUT, FORWARD
  - *nat*: PREROUTING, POSTROUTING
  - *mangle*: PREROUTING, OUTPUT, INPUT, FORWARD, POSTROUTING
- **User-defined Chains:** Custom chains to organize rules by specific criteria, e.g., traffic for a specific port or group of servers.

Rules and Targets

- Rules specify matching criteria and actions.
- Added to chains with `-A` option.
- Example: Allow incoming SSH (TCP port 22) traffic

```
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Common Targets

Target Name	Description
ACCEPT	Allow packet through
DROP	Silently drop packet
REJECT	Drop packet and notify source
LOG	Log packet details
SNAT	Source NAT - change source IP address
DNAT	Destination NAT - change destination IP address
MASQUERADE	Like SNAT, for dynamic IP addresses

Target Name	Description
REDIRECT	Redirect packets to another port or IP
MARK	Mark packets for advanced routing

Matches

Matches specify criteria to apply rules to specific packets or connections.

Match Name	Description
`-p` or `--protocol`	Protocol (tcp, udp, icmp)
`--dport`	Destination port
`--sport`	Source port
`-s` or `--source`	Source IP address
`-d` or `--destination`	Destination IP address
`-m state`	Connection state (NEW, ESTABLISHED, RELATED)
`-m multiport`	Multiple ports or port ranges
`-m tcp`	TCP-specific matching options
`-m udp`	UDP-specific matching options
`-m string`	Match packets containing a specific string
`-m limit`	Rate limiting matches
`-m conntrack`	Match connection tracking info
`-m mark`	Match Netfilter mark value
`-m mac`	Match MAC address
`-m iprange`	Match range of IP addresses

Example Rule Using Match Module

Allow incoming HTTP (TCP port 80) traffic:

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