

Securing Linux

Practical Linux System Security and Hardening

LinuXpert Systems

No.5, Dr. Ambedkar St, Rathinapuram,

Nanganallur, Chennai – 600 061.

Phone: 98841 65649, 90808 01382

E-mail: baskar@linuxpert.in

TABLE OF CONTENTS

1. sudo 1

2. su 8

3. visudo 14

4. getenforce 19

5. setenforce 23

6. firewall-cmd 28

7. iptables 36

8. nft 45

1. sudo

sudo (**S**uper**U**ser **DO**) is a command that allows a permitted user to execute another command with the security privileges of another user, typically the **root** user (system administrator).

Instead of logging in as the root user, `sudo` temporarily grants elevated privileges to perform administrative tasks.

Syntax

```
sudo [OPTIONS] COMMAND [ARGUMENTS]
```

- `COMMAND` – The command to run with elevated privileges.
- `ARGUMENTS` – Any options or arguments for the command.

Why use `sudo` ?

Linux protects important system files from normal users. Tasks like installing software, modifying system configuration, or managing users require administrator privileges.

`sudo` provides these privileges temporarily without logging in as the root user.

Example:

```
sudo apt update
```

Runs `apt update` as the root user.

Common Examples

Install software

```
sudo apt install nginx
```

Installs the Nginx web server.

Update package lists

```
sudo apt update
```

Refreshes package information.

Upgrade installed packages

```
sudo apt upgrade
```

Updates installed software.

Edit a protected file

```
sudo nano /etc/hosts
```

Opens the `/etc/hosts` file with administrator permissions.

Restart a service

```
sudo systemctl restart ssh
```

Restarts the SSH service.

Create a directory in a protected location

```
sudo mkdir /opt/myapp
```

Creates a directory under `/opt`.

Copy files into system directories

```
sudo cp myprogram /usr/local/bin/
```

Copies a file into a system directory.

Delete a protected file

```
sudo rm /etc/test.conf
```

Removes a system file (use with caution).

How `sudo` Works

When you run:

```
sudo command
```

1. Linux checks whether your user is allowed to use `sudo`.
2. It asks for **your password** (not the root password, on most distributions).
3. If authentication succeeds, the command runs as root.
4. For a short period (typically 5–15 minutes, depending on configuration), you won't be asked for your password again.

Useful Options

Show version

```
sudo -V
```

Displays the installed `sudo` version.

Display help

```
sudo -h
```

Shows help information.

Run as another user

```
sudo -u username command
```

Example:

```
sudo -u postgres psql
```

Runs `psql` as the `postgres` user.

Start a root shell

```
sudo -i
```

Starts a login shell as the root user.

Execute a single shell command

```
sudo sh -c "echo Hello"
```

Runs the shell command with root privileges.

Clear cached credentials

```
sudo -k
```

Forces `sudo` to ask for your password again the next time it is used.

Validate credentials

```
sudo -v
```

Refreshes the authentication timestamp without running a command.

List allowed commands

```
sudo -l
```

Displays the commands your user is permitted to run via `sudo`.

Configuration

Permissions are controlled by the **sudoers** file.

Never edit it directly. Instead, use:

```
sudo visudo
```

This checks the file for syntax errors before saving.

Example entry:

```
bharath ALL=(ALL:ALL) ALL
```

Meaning:

- `bharath` → user
- `ALL` → any host
- `(ALL:ALL)` → can run commands as any user and group
- `ALL` → may run any command

Important Security Notes

- **Use `sudo` only when necessary.**
- Always verify commands before pressing Enter.
- Avoid running unknown scripts with `sudo`.
- `sudo rm -rf /` or similar destructive commands can permanently damage your system.

Difference Between `sudo` and `su`

Feature	`sudo`	`su`
Uses current user's password	Yes	No (uses target user's password, usually root)
Runs a single command	Yes	No (switches user)
Audit logging	Yes	Limited
Least-privilege access	Yes	No
Recommended for administration	Yes	Sometimes

Example:

```
sudo apt update
```

Runs only `apt update` as root.

```
su -
```

Switches to the root account until you exit.

Exit a Root Shell

If you started one with:

```
sudo -i
```

Exit it using:

```
exit
```

or press:

```
Ctrl + D
```

Summary

- `sudo` allows authorised users to run commands with elevated (usually root) privileges.
- It is the standard and safest way to perform administrative tasks on Linux.
- Permissions are managed through the `sudoers` configuration.
- Prefer `sudo` over logging in directly as the root user because it provides better security, accountability, and control.

2. su

The `su` (**Switch User** or **Substitute User**) command is used in Linux to switch from the current user account to another user account without logging out. It is most commonly used to become the **root** user to perform administrative tasks.

Syntax

```
su [options] [username]
```

If no username is specified, `su` switches to the **root** user.

Basic Examples

1. Switch to the root user

```
su
```

What happens:

- Prompts for the **root password**.
- If authentication succeeds, you become the root user.
- Your current shell changes to a root shell.

Example:

```
$ whoami
bharath

$ su
Password:

# whoami
root
```

Notice the prompt changes from `$` to `#`.

2. Switch to another user

```
su john
```

Example:

```
$ whoami
bharath

$ su john
Password:

$ whoami
john
```

You must know **john's password**.

---

3. Start a login shell

```
su -
```

or

```
su - root
```

This is the preferred way to become root.

It:

- Loads root's environment.
- Changes to root's home directory (`/root`).
- Reads root's login files such as `.profile` or `.bash_profile`.

Example:

```
$ pwd
/home/bharath

$ su -

# pwd
/root
```

4. Login as another user

```
su - john
```

This behaves as if **john** logged in directly.

Common Options

Option	Description
`-` or `-l`	Login shell (recommended)
`-c command`	Run a single command as another user
`-s shell`	Use a specified shell
`-m`, `-p`	Preserve current environment variables

Run a Single Command

```
su -c "ls /root"
```

Example:

```
su -c "systemctl status ssh"
```

After executing the command, it returns to the original user.

Specify a Different Shell

```
su -s /bin/bash john
```

Starts Bash for user **john**, regardless of their default shell.

Preserve Environment

```
su -p john
```

or

```
su -m john
```

```
Keeps most of the current user's environment variables.
```

Verify the Current User

Before:

```
$ whoami
bharath
```

Switch:

```
$ su
Password:
```

After:

```
# whoami
root
```

Check User Information

```
id
```

Example:

```
# id
uid=0(root) gid=0(root) groups=0(root)
```

Exit Back to the Previous User

```
exit
```

or press:

```
Ctrl + D
```

Example:

```
# exit
$
```

`su` vs `sudo`

Feature	`su`	`sudo`
Password required	Target user's password (usually root)	Current user's password
Switch user	Yes	Can run as another user
Login shell	Yes	No (unless specified)
Run one command	Yes (`su -c`)	Yes (default behaviour)
Security	Less granular	More granular and auditable

Example:

```
su -
```

becomes root until you exit.

```
sudo apt update
```

runs only that single command with elevated privileges.

Practical Examples

Become root

```
su -
```

Switch to user `alice`

```
su alice
```

Run a command as root

```
su -c "cat /etc/shadow"
```

Return to the previous user

```
exit
```

Typical Use Cases

- Performing system administration tasks as **root**.
- Testing applications under another user's account.
- Troubleshooting user-specific configuration issues.
- Managing files owned by another user.
- Running commands that require administrative privileges.

Best Practices

- Prefer `su -` over `su` when switching to root, as it provides a clean login environment.
- Use `sudo` for individual administrative commands instead of remaining logged in as root whenever possible.
- Exit the root shell (`exit`) as soon as administrative work is complete.
- Avoid sharing the root password with multiple users; `sudo` offers better accountability.

Summary

- `su` switches to another user account.
- `su` (without a username) attempts to switch to **root**.
- `su -` starts a full login shell with the target user's environment.
- `su username` switches to a specified user.
- `exit` or `Ctrl+D` returns to the previous user.
- For one-off administrative commands, `sudo` is generally the safer and more common choice.

3. visudo

The **`visudo`** command is used to **safely edit the `/etc/sudoers` file**, which controls **who can execute commands with `sudo` (superuser privileges)**.

Unlike editing the file directly with editors like `vi` or `nano`, `visudo` performs syntax checking before saving. This helps prevent configuration errors that could make `sudo` unusable.

Syntax

```
visudo
```

What `visudo` Does

- Opens the `/etc/sudoers` file in a text editor.
- Locks the file to prevent multiple administrators from editing it simultaneously.
- Checks the syntax before saving.
- Refuses to save the file if syntax errors are found (unless you explicitly choose to ignore them).

Basic Example

```
sudo visudo
```

This opens the `sudoers` file for editing.

Example entry:

```
john ALL=(ALL:ALL) ALL
```

Meaning:

- **john** → User name
- **ALL** → Any host
- **(ALL:ALL)** → Can run commands as any user and any group
- **ALL** → May execute any command

Common Options

1. Specify another sudoers file

```
sudo visudo -f /tmp/testsudoers
```

Edits an alternative sudoers file instead of `/etc/sudoers`.

---

2. Check syntax only

```
sudo visudo -c
```

Checks the syntax of the sudoers file without opening an editor.

Example output:

```
/etc/sudoers: parsed OK
```

---

3. Check multiple sudoers files

```
sudo visudo -c -f /etc/sudoers.d/admins
```

Verifies the specified file for syntax errors.

---

4. Select a different editor

```
sudo EDITOR=nano visudo
```

or

```
sudo VISUAL=nano visudo
```

Opens the file in `nano` instead of the default editor.

Example: Grant sudo access

Suppose you want user **alice** to have full sudo privileges.

Open:

```
sudo visudo
```

Add:

```
alice ALL=(ALL:ALL) ALL
```

Now Alice can execute:

```
sudo apt update
```

Example: Passwordless sudo

```
alice ALL=(ALL) NOPASSWD: ALL
```

Now Alice will not be prompted for a password when using `sudo`.

> Be careful with `NOPASSWD`, as it reduces security.

Example: Allow only one command

```
backup ALL=(root) /usr/bin/rsync
```

The `backup` user may execute only:

```
sudo rsync
```

Other commands are not permitted.

Include additional configuration files

The sudoers file often contains:

```
#includedir /etc/sudoers.d
```

Instead of modifying `/etc/sudoers` directly, you can create separate files under:

```
/etc/sudoers.d/
```

For example:

```
sudo visudo -f /etc/sudoers.d/developer
```

This is the recommended way to add custom sudo rules because it keeps the main configuration file clean and easier to maintain.

Important Sudoers Aliases

```
User_Alias
Runas_Alias
Host_Alias
Cmnd_Alias
```

Example:

```
User_Alias ADMINS = alice,bob
Cmnd_Alias SERVICES = /usr/bin/systemctl

ADMINS ALL=(ALL) SERVICES
```

This allows `alice` and `bob` to run `systemctl` via `sudo`.

Common `visudo` Options

Option	Description
`visudo`	Edit the default sudoers file
`visudo -c`	Check syntax only
`visudo -f file`	Edit an alternate sudoers file
`visudo -V`	Display version information
`EDITOR=nano visudo`	Use `nano` as the editor

Advantages of `visudo`

- Prevents syntax errors.
- Locks the file during editing.
- Verifies configuration before saving.
- Protects against accidentally breaking `sudo`.
- Supports editing alternate sudoers files.

Best Practices

- Always use `visudo` instead of editing `/etc/sudoers` directly.
- Prefer adding custom rules in `/etc/sudoers.d/` rather than modifying the main file.
- Use the principle of least privilege: grant only the commands users need.
- After making changes, verify the configuration with:

```
sudo visudo -c
```

Summary

Feature	Description
Command	`visudo`
Purpose	Safely edit the `sudoers` configuration
Default file	`/etc/sudoers`
Requires root?	Yes (typically run with `sudo`)
Main benefit	Syntax validation and file locking
Common use	Grant or restrict `sudo` privileges for users and groups

4. getenforce

`getenforce` is a Linux command used to **display the current mode of SELinux (Security-Enhanced Linux)**.

Syntax

```
getenforce
```

It takes **no arguments or options**.

What is SELinux ?

SELinux (Security-Enhanced Linux) is a Linux security framework that provides **mandatory access control (MAC)**. It restricts what users, processes, and applications can access, even if they have traditional Linux permissions.

Possible Outputs

1. Enforcing

```
$ getenforce
Enforcing
```

- SELinux is **enabled**.
- Security policies are **actively enforced**.
- Any action that violates SELinux policy is **blocked**.

---

2. Permissive

```
$ getenforce
Permissive
```

- SELinux is enabled.
- Policy violations are **logged**.
- Actions are **allowed**, making this mode useful for troubleshooting.

3. Disabled

```
$ getenforce
Disabled
```

- SELinux is completely disabled.
- No SELinux policies are loaded or enforced.

Examples

Check the current SELinux mode

```
getenforce
```

Example output:

```
Enforcing
```

Use in a shell script

```
if [ "$(getenforce)" = "Enforcing" ]; then
    echo "SELinux is enforcing security policies."
fi
```

Typical Use Cases

- Verify whether SELinux is enabled.
- Check if SELinux is causing permission issues.
- Confirm the system's security mode before deploying applications.
- Troubleshoot access denied errors.
- Use in automation scripts to verify security settings.

Related Commands

Command	Purpose
`sestatus`	Displays detailed SELinux status information.
`setenforce 0`	Switches SELinux to **Permissive** mode (temporarily).
`setenforce 1`	Switches SELinux to **Enforcing** mode (temporarily).
`cat /etc/selinux/config`	Shows the configured SELinux mode for the next boot.

Difference between `getenforce` and `sestatus`

`getenforce`	`sestatus`
Shows only the current SELinux mode	Shows detailed SELinux information
Very fast and simple	More comprehensive
Output: `Enforcing`, `Permissive`, or `Disabled`	Shows status, current mode, policy type, policy location, config file mode, and more

Example:

```
$ getenforce
Enforcing

$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
```

Notes

- `getenforce` **does not require root privileges**.
- It only reports the **current runtime SELinux mode**.
- To change the mode temporarily, use `setenforce`.
- Permanent changes require editing `/etc/selinux/config` and rebooting.

Summary

- **Command:** `getenforce`
- **Purpose:** Displays the current SELinux enforcement mode.
- **Outputs:** `Enforcing`, `Permissive`, or `Disabled`.
- **Requires root?** No.
- **Common use:** Quickly verify whether SELinux is actively enforcing security policies.

5. setenforce

`setenforce` is a Linux command used to **change the current mode of SELinux (Security-Enhanced Linux)** without rebooting the system.

SELinux is a Linux security module that provides **mandatory access control (MAC)**, enforcing security policies that restrict what users, processes, and applications can access.

Syntax

```
setenforce [0|1]
```

or

```
setenforce Enforcing
```

or

```
setenforce Permissive
```

Modes

Value	Mode	Description
`1`	Enforcing	SELinux actively enforces security policies and blocks unauthorized actions.
`0`	Permissive	SELinux does not block actions but logs policy violations.

Common Examples

Enable Enforcing Mode

```
sudo setenforce 1
```

or

```
sudo setenforce Enforcing
```

Example output:

```
(no output if successful)
```

Check the status:

```
getenforce
```

Output:

```
Enforcing
```

Switch to Permissive Mode

```
sudo setenforce 0
```

or

```
sudo setenforce Permissive
```

Verify:

```
getenforce
```

Output:

```
Permissive
```

Check Current SELinux Mode

```
getenforce
```

Possible outputs:

```
Enforcing
```

or

```
Permissive
```

or

```
Disabled
```

View Detailed SELinux Status

```
sestatus
```

Example output:

```
SELinux status:                 enabled
Current mode:                   enforcing
Mode from config file:          enforcing
Policy version:                 33
```

Make the Change Permanent

`setenforce` **only changes the mode until the next reboot**.
To make the change permanent, edit the SELinux configuration file:

```
sudo nano /etc/selinux/config
```

Modify:

```
SELINUX=enforcing
```

or

```
SELINUX=permissive
```

or

```
SELINUX=disabled
```

After saving the file, reboot the system.

Requirements

- Root privileges (`sudo` or root user)
- SELinux must be **enabled**
- Cannot enable SELinux if it is **disabled** in the configuration

If SELinux is disabled:

```
getenforce
```

Output:

```
Disabled
```

Then:

```
sudo setenforce 1
```

Output:

```
setenforce: SELinux is disabled
```

Practical Use Cases

1. Troubleshooting Application Issues

If an application is failing due to SELinux restrictions:

```
sudo setenforce 0
```

If the application starts working, it's likely being blocked by an SELinux policy.

---

2. Testing SELinux Policies

Administrators often use Permissive mode to:

- Test new SELinux policies
- Collect audit logs
- Identify denied operations before enforcing them

3. Restoring System Security

After testing:

```
sudo setenforce 1
```

This restores full SELinux protection without rebooting.

Related Commands

Command	Purpose
`getenforce`	Display the current SELinux mode
`sestatus`	Show detailed SELinux status
`setenforce`	Change SELinux mode temporarily
`restorecon`	Restore default SELinux file contexts
`chcon`	Change SELinux file security context
`semanage`	Manage SELinux policies, ports, and contexts
`ausearch`	Search SELinux audit logs
`audit2allow`	Generate SELinux policy rules from audit logs

Exit Status

- `0` – Command executed successfully.
- Non-zero – An error occurred (for example, insufficient privileges, invalid argument, or SELinux is disabled).

Summary

- **Command:** `setenforce`
- **Purpose:** Temporarily change the SELinux enforcement mode.
- **`setenforce 1`:** Enable **Enforcing** mode (blocks unauthorized actions).
- **`setenforce 0`:** Enable **Permissive** mode (logs violations but does not block them).
- **Effect:** Immediate, but **not persistent across reboots**.
- **Persistent configuration:** Edit `/etc/selinux/config` and reboot.

6. firewall-cmd

`firewall-cmd` is the **command-line utility used to manage `firewalld`**, the dynamic firewall management service available on many Linux distributions such as **RHEL, CentOS, Rocky Linux, AlmaLinux, Fedora**, and optionally on other distributions.

It allows you to **view, configure, and manage firewall rules** without manually editing firewall configuration files or directly using `iptables`/`nftables`.

Syntax

```
firewall-cmd [OPTIONS]
```

What does it do?

With `firewall-cmd`, you can:

- Open or close network ports
- Allow or block services
- Create firewall zones
- Configure trusted networks
- Allow specific IP addresses
- Enable port forwarding
- Reload firewall rules
- View firewall status

Check if firewalld is running

```
firewall-cmd --state
```

Example output

```
running
```

or

```
not running
```

Display the default zone

```
firewall-cmd --get-default-zone
```

Example

```
public
```

List all firewall settings

```
firewall-cmd --list-all
```

Example

```
public
  target: default
  services: ssh dhcpv6-client
  ports:
  protocols:
  masquerade: no
  forward-ports:
```

List open ports

```
firewall-cmd --list-ports
```

Example

```
80/tcp 443/tcp
```

List allowed services

```
firewall-cmd --list-services
```

Example

```
ssh dhcpv6-client http https
```

Open a port temporarily

Allow TCP port 8080:

```
sudo firewall-cmd --add-port=8080/tcp
```

Output

```
success
```

This rule disappears after reboot or firewall reload.

Open a port permanently

```
sudo firewall-cmd --permanent --add-port=8080/tcp
```

Reload firewall

```
sudo firewall-cmd --reload
```

Remove a port

Temporary

```
sudo firewall-cmd --remove-port=8080/tcp
```

Permanent

```
sudo firewall-cmd --permanent --remove-port=8080/tcp
sudo firewall-cmd --reload
```

Allow a service

Example: HTTP

```
sudo firewall-cmd --add-service=http
```

Permanent

```
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd –reload
```

Remove a service

```
sudo firewall-cmd --remove-service=http
```

Permanent

```
sudo firewall-cmd --permanent --remove-service=http
sudo firewall-cmd --reload
```

Reload firewall configuration

```
sudo firewall-cmd --reload
```

This reloads permanent rules without restarting the firewall service.

Show available zones

```
firewall-cmd --get-zones
```

Example

```
block dmz drop external home internal public trusted work
```

Show active zones

```
firewall-cmd --get-active-zones
```

Example :

```
public
  interfaces: eth0
```

Change the default zone

```
sudo firewall-cmd --set-default-zone=home
```

Assign an interface to a zone

```
sudo firewall-cmd --zone=public --change-interface=eth0
```

Allow an IP address

```
sudo firewall-cmd --permanent \
    --zone=public \
    --add-source=192.168.1.100
```

Reload

```
sudo firewall-cmd --reload
```

Block an IP address (Rich Rule)

```
sudo firewall-cmd --permanent \
    --add-rich-rule='rule family="ipv4" source address="192.168.1.100" reject'
```

Reload

```
sudo firewall-cmd –reload
```

Enable masquerading (NAT)

```
sudo firewall-cmd --add-masquerade
```

Permanent

```
sudo firewall-cmd --permanent --add-masquerade
sudo firewall-cmd --reload
```

Forward a port

Forward TCP port 80 to 8080

```
sudo firewall-cmd --permanent \
--add-forward-port=port=80:proto=tcp:toport=8080
```

Reload

```
sudo firewall-cmd --reload
```

Runtime vs Permanent Configuration

`firewalld` maintains two separate configurations:

Runtime	Permanent
Active immediately	Saved on disk
Lost after reboot or `firewall-cmd --reload`	Survives reboot
No `--permanent` option	Requires `--permanent`
Good for testing	Good for production

Example:

Temporary:

```
sudo firewall-cmd --add-service=http
```

Permanent:

```
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --reload
```

Common Options

Option	Description
`--state`	Shows firewall status
`--list-all`	Displays current configuration
`--list-services`	Lists allowed services
`--list-ports`	Lists open ports
`--add-port=`	Opens a port
`--remove-port=`	Closes a port
`--add-service=`	Allows a service
`--remove-service=`	Removes a service
`--reload`	Reloads permanent configuration
`--runtime-to-permanent`	Saves current runtime rules as permanent
`--get-zones`	Lists all zones
`--get-active-zones`	Shows active zones
`--get-default-zone`	Displays the default zone
`--set-default-zone=`	Changes the default zone

Practical Examples

Open HTTP and HTTPS permanently

```
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
sudo firewall-cmd --reload
```

Allow SSH

```
sudo firewall-cmd --permanent --add-service=ssh
sudo firewall-cmd –reload
```

Open MySQL port

```
sudo firewall-cmd --permanent --add-port=3306/tcp
sudo firewall-cmd --reload
```

Open a custom application port

```
sudo firewall-cmd --permanent --add-port=5000/tcp
sudo firewall-cmd --reload
```

Advantages of `firewall-cmd`

- Dynamic firewall management without restarting services.
- Supports **zones** to apply different security policies to different networks.
- Simpler and safer than editing `iptables` or `nftables` rules directly.
- Allows temporary (runtime) changes for testing and permanent changes for production.
- Integrates with both IPv4 and IPv6.

Summary

`firewall-cmd` is the primary command-line tool for managing the `firewalld` service. It enables administrators to control network access by opening or closing ports, allowing services, configuring zones, managing IP-based rules, and applying changes dynamically. Understanding the distinction between **runtime** and **permanent** configurations is essential for using `firewall-cmd` effectively in production environments.

7. iptables

`iptables` is a **command-line utility used to configure the Linux kernel firewall**. It allows you to **allow, block, redirect, or log network traffic** based on rules that you define.

It works with the **Netfilter** framework built into the Linux kernel.

Syntax

```
iptables [OPTIONS] CHAIN RULE-SPECIFICATION [TARGET]
```

or

```
iptables [COMMAND] [CHAIN] [OPTIONS]
```

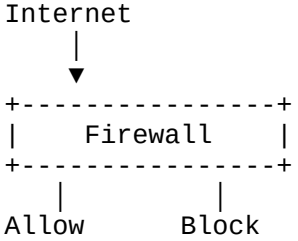
What is a Firewall ?

A firewall acts like a **security guard** for your computer.

It examines every incoming and outgoing network packet and decides whether to:

- Allow it
- Block it
- Log it
- Redirect it

Example:



Main Tables

`iptables` organizes rules into different **tables**.

Table	Purpose
filter	Filtering packets (most commonly used)
nat	Network Address Translation (port forwarding, masquerading)
mangle	Modify packet headers
raw	Configure packet tracking
security	SELinux security rules

The default table is **filter**.

Chains

Each table contains **chains**.

Chain	Description
INPUT	Incoming packets destined for this computer
OUTPUT	Packets generated by this computer
FORWARD	Packets passing through the computer (router)
PREROUTING	Packets before routing decision
POSTROUTING	Packets after routing decision

Example:

```
Internet
      |
      ▼
   INPUT ----> Local Computer

Local Computer
      |
      ▼
   OUTPUT ----> Internet

Internet ---> FORWARD ---> Another Network
```

Targets

When a packet matches a rule, a **target** tells iptables what action to take.

Common targets:

Target	Action
ACCEPT	Allow packet
DROP	Silently discard packet
REJECT	Reject packet and send an error response
LOG	Log packet information
DNAT	Destination NAT
SNAT	Source NAT
MASQUERADE	Dynamic source NAT

Common Commands

1. List firewall rules

```
iptables -L
```

Example:

```
Chain INPUT
target     prot opt source      destination
ACCEPT     all  --  anywhere    anywhere
DROP       tcp  --  anywhere    anywhere tcp dpt:ssh
```

---

2. Show rules with numbers

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

Example:

```
Chain INPUT
num target
1   ACCEPT
2   DROP

Useful when deleting rules.
```

3. Display detailed rules

```
iptables -L -v
```

Shows:

- packets matched
- bytes matched
- interfaces
- protocol

---

4. Append a rule

Allow SSH:

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

Explanation:

- `-A` → Append rule
- `INPUT` → Incoming traffic
- `-p tcp` → TCP protocol
- `--dport 22` → Port 22
- `-j ACCEPT` → Allow traffic

---

5. Block SSH

```
iptables -A INPUT -p tcp --dport 22 -j DROP
```

6. Delete a rule

Delete by number:

```
iptables -D INPUT 2
```

Delete by specification:

```
iptables -D INPUT -p tcp --dport 22 -j ACCEPT
```

---

7. Insert rule at the top

```
iptables -I INPUT 1 -p tcp --dport 22 -j ACCEPT
```

Inserts at rule number 1.

---

8. Replace a rule

```
iptables -R INPUT 2 -p tcp --dport 80 -j ACCEPT
```

---

9. Flush all rules

```
iptables -F
```

This removes all rules from the current table.

---

10. Set default policy

Allow everything:

```
iptables -P INPUT ACCEPT
```

Block everything:

```
iptables -P INPUT DROP
```

11. Save rules

On Debian/Ubuntu:

```
iptables-save > rules.v4
```

Restore:

```
iptables-restore < rules.v4
```

Useful Options

Option	Meaning
`-A`	Append rule
`-I`	Insert rule
`-D`	Delete rule
`-R`	Replace rule
`-L`	List rules
`-F`	Flush rules
`-P`	Set default policy
`-N`	Create new chain
`-X`	Delete user-defined chain
`-Z`	Reset packet/byte counters
`-t`	Select table
`-v`	Verbose output
`-n`	Show numeric addresses and ports (avoid DNS lookups)
`-j`	Jump to target
`-p`	Protocol
`-s`	Source address
`-d`	Destination address
`--sport`	Source port
`--dport`	Destination port

Practical Examples

Allow SSH

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

Allow HTTP

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

Allow HTTPS

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

Block a specific IP

```
iptables -A INPUT -s 192.168.1.100 -j DROP
```

Allow only a specific IP to SSH

```
iptables -A INPUT -p tcp -s 192.168.1.50 --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j DROP
```

Allow ping requests

```
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
```

Block ping requests

```
iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

Log dropped packets

```
iptables -A INPUT -j LOG --log-prefix "Blocked: "
```

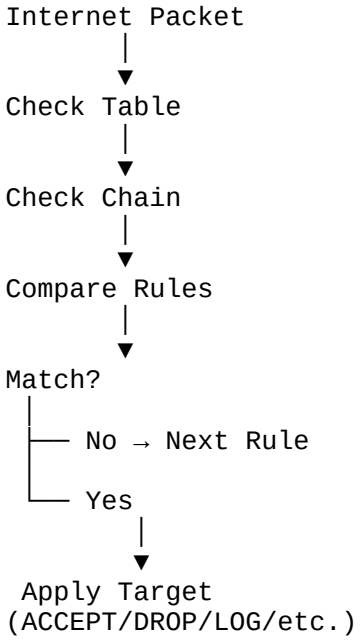
Viewing NAT Rules

```
iptables -t nat -L
```

View Rules in All Tables

```
iptables -t filter -L
iptables -t nat -L
iptables -t mangle -L
iptables -t raw -L
```

Typical Workflow



Important Notes

- Most `iptables` operations require **root** privileges (or `sudo`).
- Rule order matters: packets are evaluated **top to bottom**, and the first matching rule is applied.
- Be cautious when modifying firewall rules over SSH—you can accidentally block your own connection.
- Many modern Linux distributions now use **`nftables`** as the underlying firewall framework. On systems such as recent Ubuntu, Debian, Fedora, and RHEL versions, the `iptables` command may actually use an **`iptables-nft` compatibility layer** that translates rules into `nftables`. Legacy `iptables` is still widely encountered and remains important to understand for system administration and certifications like RHCSA/RHCE.

Summary

Feature	Description
Purpose	Configure the Linux firewall
Works with	Netfilter kernel framework
Default table	`filter`
Main chains	`INPUT`, `OUTPUT`, `FORWARD`, `PREROUTING`, `POSTROUTING`
Common actions	`ACCEPT`, `DROP`, `REJECT`, `LOG`
List rules	`iptables -L`
Add rule	`iptables -A`
Delete rule	`iptables -D`
Flush rules	`iptables -F`
Default policy	`iptables -P`

For Linux administration exams (RHCSA/RHCE, LPIC, CompTIA Linux+), `iptables` is considered a foundational firewall tool, even though `nftables` is the recommended modern replacement on many current distributions.

8. nft

The `nft` command is the command-line utility used to configure and manage **nftables**, the modern Linux packet filtering and firewall framework. It replaces older firewall tools such as **iptables**, **ip6tables**, **arptables**, and **ebtables** with a unified interface.

- **Command:** `nft`
- **Package:** `nftables`
- **Configuration file:** `/etc/nftables.conf`
- **Purpose:** Create and manage firewall rules, NAT, packet filtering, and traffic classification.

Syntax

```
nft [options] [command]
```

Common Options

Option	Description
`-h`, `--help`	Display help information
`-v`	Show version
`-f <file>`	Load rules from a file
`-i`	Interactive mode
`-c`	Check configuration without applying it
`-a`	Display rule handles
`-j`	Output in JSON format

Basic Examples

Display the installed version

```
nft --version
```

Example output:

```
nftables v1.0.9
```

List all firewall rules

```
sudo nft list ruleset
```

Example:

```
table inet filter {
    chain input {
        type filter hook input priority 0;
        policy accept;
    }
}
```

List available tables

```
sudo nft list tables
```

Example:

```
table inet filter
table ip nat
```

List chains in a table

```
sudo nft list table inet filter
```

Create a table

```
sudo nft add table inet filter
```

Explanation:

- Creates a table named **filter**
- Uses the **inet** family (IPv4 + IPv6)

Create a chain

```
sudo nft add chain inet filter input \
'{ type filter hook input priority 0; policy accept; }'
```

This creates the **input** chain.

Allow SSH

```
sudo nft add rule inet filter input tcp dport 22 accept
```

Meaning:

- Match TCP packets
- Destination port 22
- Accept them

Block Telnet

```
sudo nft add rule inet filter input tcp dport 23 drop
```

Allow HTTP and HTTPS

```
sudo nft add rule inet filter input tcp dport {80,443} accept
```

Delete a rule

First display handles:

```
sudo nft -a list ruleset
```

Example:

```
handle 12
```

Delete:

```
sudo nft delete rule inet filter input handle 12
```

Flush all rules

```
sudo nft flush ruleset
```

Removes every firewall rule.

Load rules from a configuration file

```
sudo nft -f /etc/nftables.conf
```

Check configuration syntax

```
sudo nft -c -f /etc/nftables.conf
```

No changes are applied—only syntax is validated.

Common Rule Examples

Allow established connections

```
sudo nft add rule inet filter input ct state established,related accept
```

Drop invalid packets

```
sudo nft add rule inet filter input ct state invalid drop
```

Allow loopback interface

```
sudo nft add rule inet filter input iif lo accept
```

Allow ICMP (ping)

```
sudo nft add rule inet filter input ip protocol icmp accept
```

Allow a subnet

```
sudo nft add rule inet filter input ip saddr 192.168.1.0/24 accept
```

Block an IP address

```
sudo nft add rule inet filter input ip saddr 10.10.10.5 drop
```

Save Rules

Many Linux distributions load rules from:

```
/etc/nftables.conf
```

To save the current ruleset:

```
sudo nft list ruleset > /etc/nftables.conf
```

View Current Rules

```
sudo nft list ruleset
```

or

```
sudo nft list table inet filter
```

Common `nft` Commands

Command	Purpose
`nft list ruleset`	Show all firewall rules
`nft list tables`	Show tables
`nft add table`	Create a table
`nft add chain`	Create a chain
`nft add rule`	Add a firewall rule
`nft delete rule`	Remove a rule
`nft flush ruleset`	Remove all rules
`nft -f file`	Load rules from a file
`nft -c -f file`	Check configuration syntax
`nft --version`	Show version

`nft` vs `iptables`

Feature	nftables (`nft`)	iptables
Introduced	Modern replacement	Legacy firewall
IPv4 & IPv6	Unified (`inet` family)	Separate tools (`iptables`/`ip6tables`)
Rule syntax	Simpler and more expressive	More verbose
Performance	Better with large rule sets	Slower as rules grow
Sets and maps	Built-in	Limited support
Recommended	Yes	Legacy systems only

When to Use `nft`

Use `nft` when you need to:

- Configure a Linux firewall.
- Allow or block network traffic.
- Perform Network Address Translation (NAT).
- Filter packets based on IP addresses, ports, protocols, or interfaces.
- Replace legacy `iptables` rules with the modern `nftables` framework.
- Manage both IPv4 and IPv6 firewall rules from a single tool.

`nft` is the recommended firewall management utility on most modern Linux distributions, including recent versions of Debian, Ubuntu, Fedora, RHEL, and openSUSE.