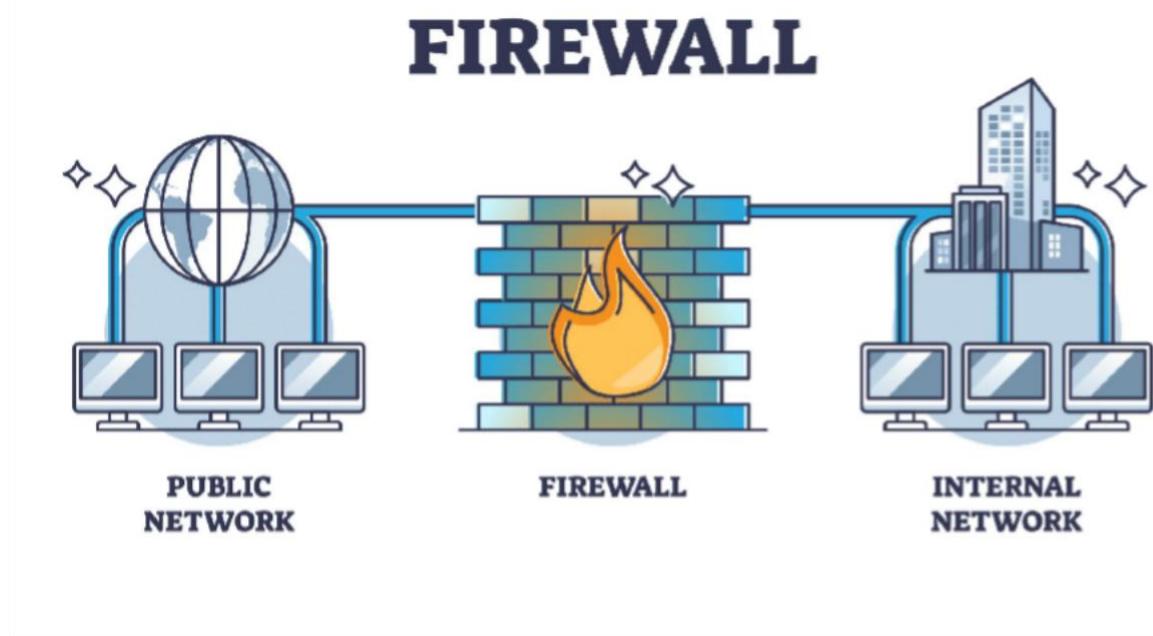


Setup and Use a Firewall on Windows/Linux



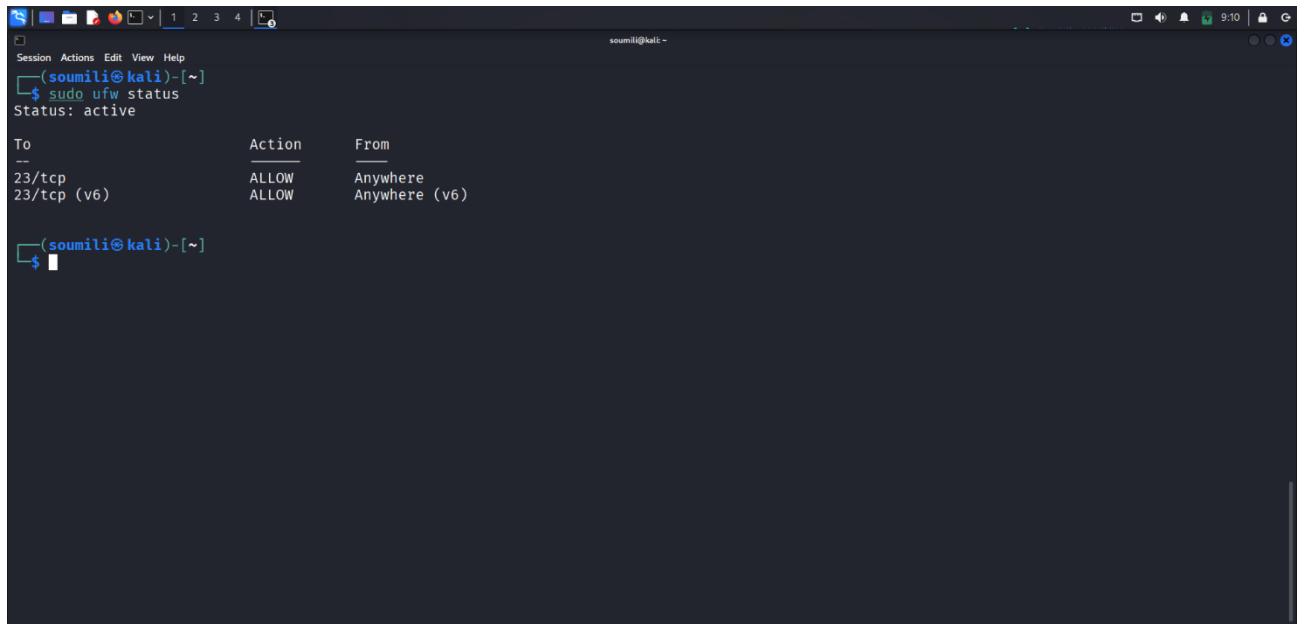
We setup and use firewall on Linux.

Open Firewall Configuration Tool

On Linux (UFW):

Open firewall configuration tool (**Windows Firewall or terminal for UFW**)

Open a terminal and run “**sudo ufw status**” to check if it's active.
If not, enable it with “**sudo ufw enable**”.



The screenshot shows a terminal window on a Kali Linux desktop environment. The terminal title is "(soumili㉿kali)-[~]". The user has run the command `sudo ufw status`. The output indicates that UFW is active and shows two rules:

To	Action	From
--	ALLOW	Anywhere
23/tcp	ALLOW	Anywhere (v6)
23/tcp (v6)		

At the bottom of the terminal, there is a prompt: `[soumili㉿kali)-[~]`.

List Current Firewall Rules

On Linux (UFW):

Run “**sudo ufw status verbose**” in the terminal. This lists active rules, including allowed/denied ports and directions (e.g., “22/tcp ALLOW IN Anywhere”).

First status 22/tcp Deny



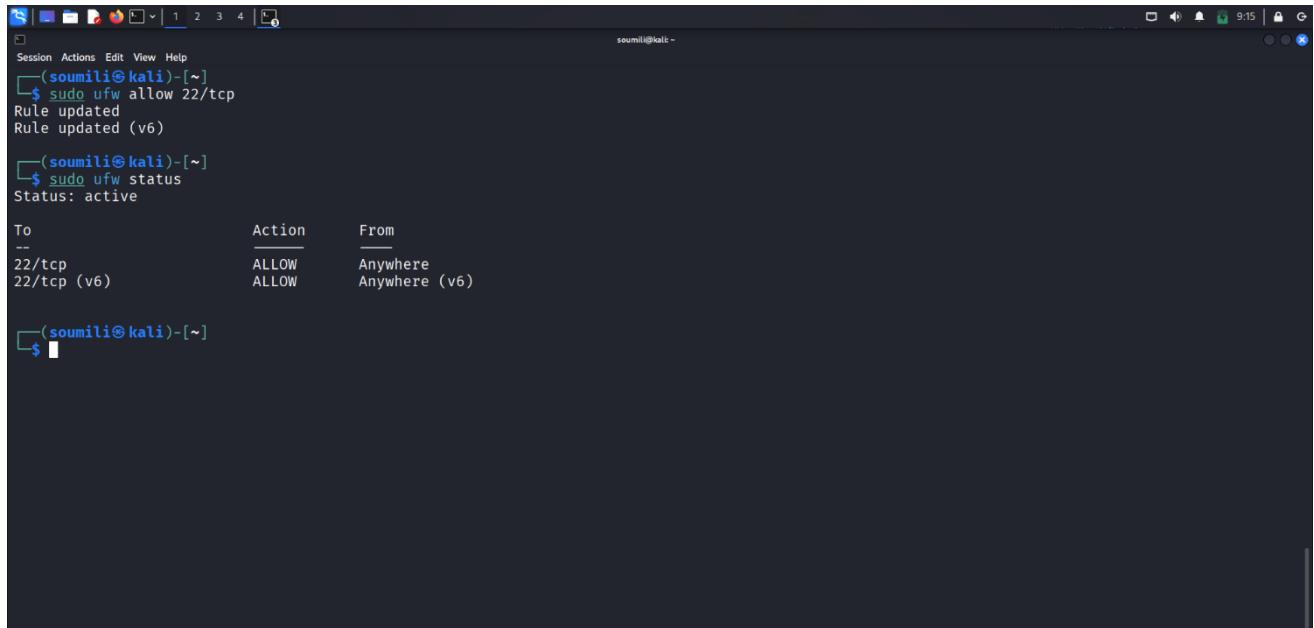
A screenshot of a terminal window on a Kali Linux desktop. The title bar shows the session name and number. The terminal prompt is `soumili@kali:~$`. The user runs the command `sudo ufw status`, which outputs:

```
Session Actions Edit View Help
└─(soumili㉿kali)-[~]
$ sudo ufw status
Status: active

To                         Action      From
--                         DENY       Anywhere
22/tcp (v6)                DENY       Anywhere (v6)

└─(soumili㉿kali)-[~]
$
```

Then Allow 22/tcp



A screenshot of a terminal window on a Kali Linux desktop. The title bar shows the session name and number. The terminal prompt is `soumili@kali:~$`. The user runs the command `sudo ufw allow 22/tcp`, followed by `sudo ufw status`, which outputs:

```
Session Actions Edit View Help
└─(soumili㉿kali)-[~]
$ sudo ufw allow 22/tcp
Rule updated
Rule updated (v6)

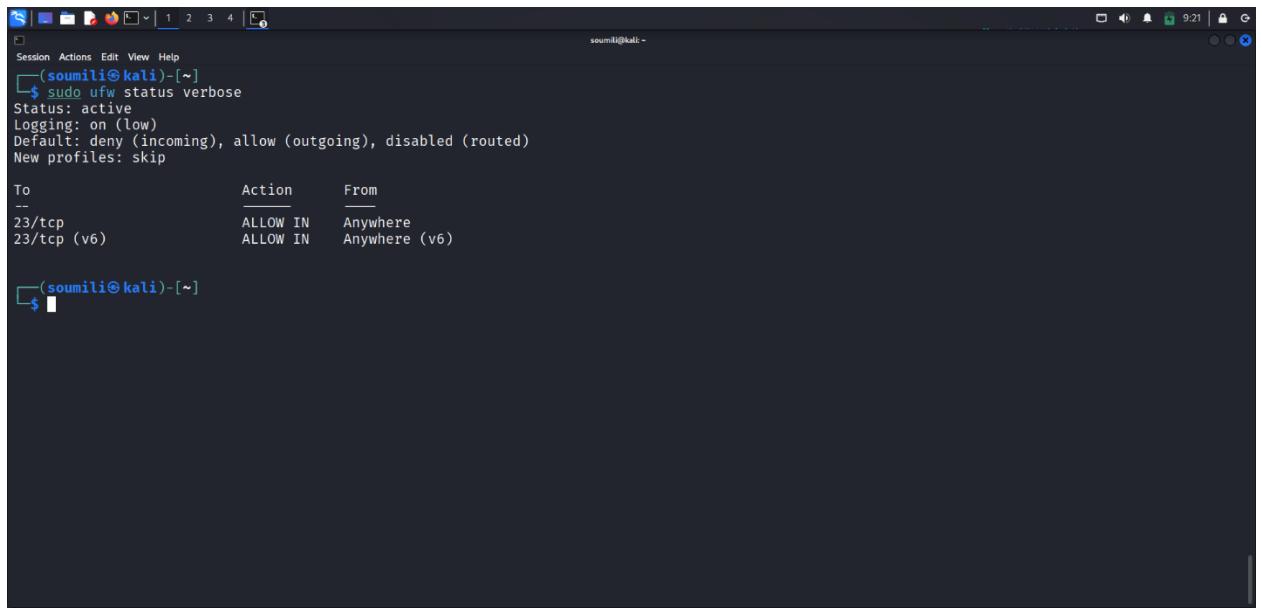
└─(soumili㉿kali)-[~]
$ sudo ufw status
Status: active

To                         Action      From
--                         ALLOW      Anywhere
22/tcp (v6)                ALLOW      Anywhere (v6)

└─(soumili㉿kali)-[~]
$
```

Add a Rule to Block Inbound Traffic on a Specific Port (e.g., 23 for Telnet)

On Linux (UFW):

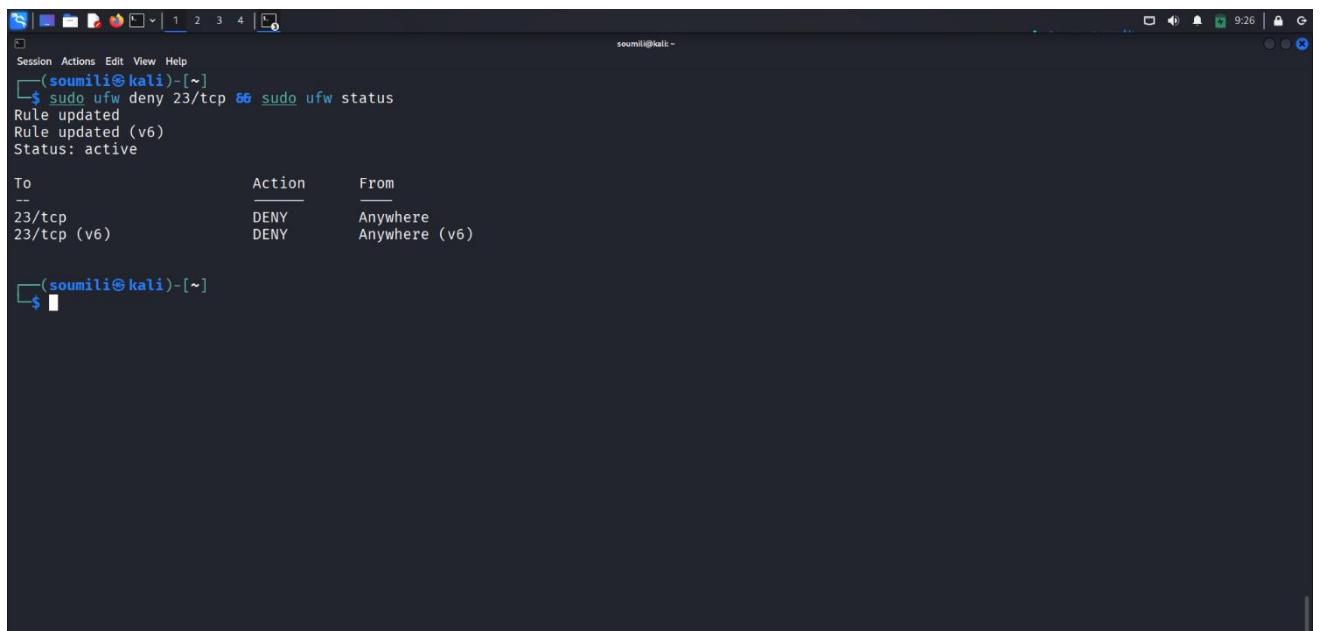


```
Session Actions Edit View Help
└─(soumili㉿kali)-[~]
$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To           Action      From
--          ALLOW IN    Anywhere
23/tcp       ALLOW IN    Anywhere (v6)

└─(soumili㉿kali)-[~]
$ █
```

Run “**sudo ufw deny 23/tcp**” (this blocks inbound on port 23).
To confirm, run “**sudo ufw status**”.



```
Session Actions Edit View Help
└─(soumili㉿kali)-[~]
$ sudo ufw deny 23/tcp
66 sudo ufw status
Rule updated
Rule updated (v6)
Status: active

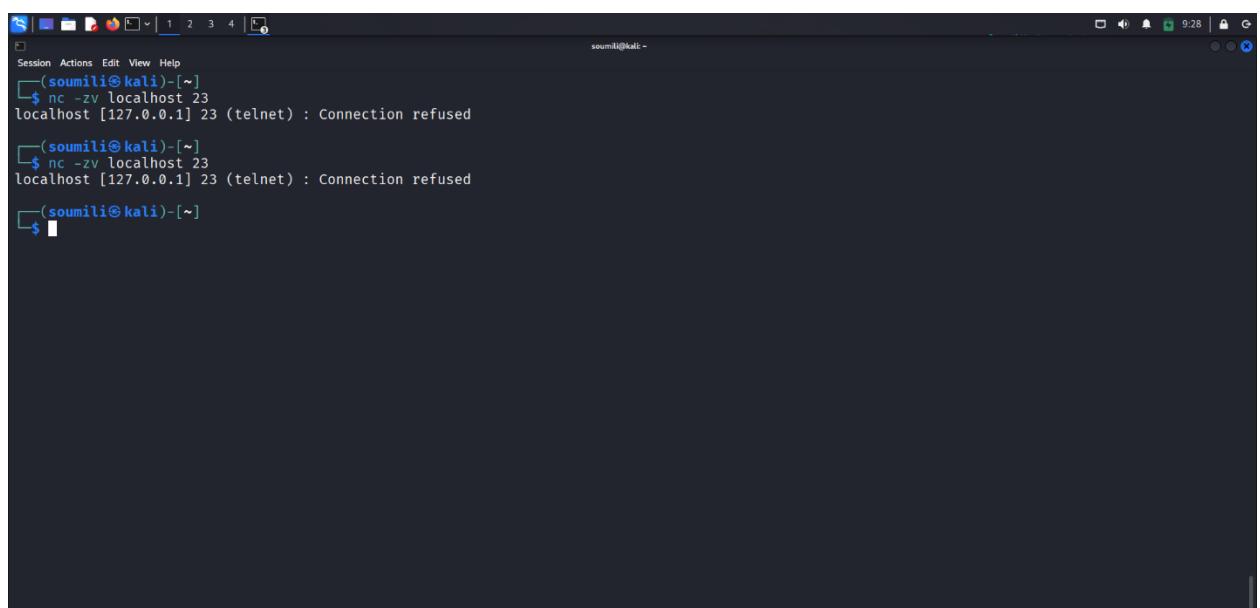
To           Action      From
--          DENY        Anywhere
23/tcp       DENY        Anywhere (v6)

└─(soumili㉿kali)-[~]
$ █
```

Test the Rule by Attempting to Connect to That Port Locally or Remotely

- Local Test (on the same machine):

Using Netcat on Linux & Try to connect.



```
soumili@kali:~$ nc -zv localhost 23
localhost [127.0.0.1] 23 (telnet) : Connection refused
soumili@kali:~$ nc -zv localhost 23
localhost [127.0.0.1] 23 (telnet) : Connection refused
soumili@kali:~$
```

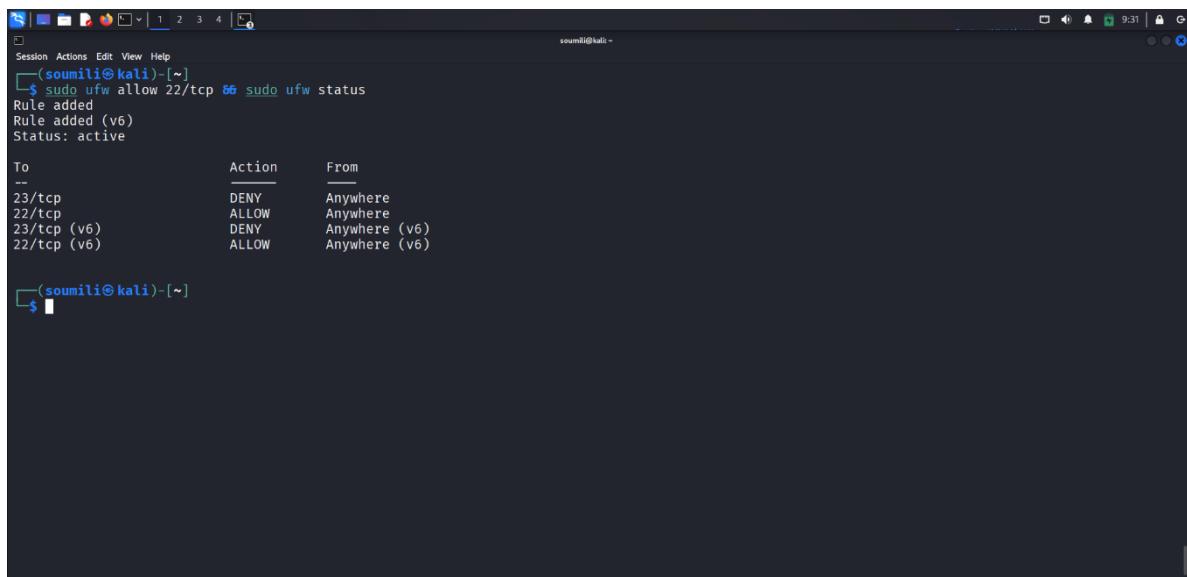
Reports:

Expected result: Connection should fail (e.g., "Connection refused" or timeout), confirming the block.

Add Rule to Allow SSH (Port 22) If on Linux

On Linux (UFW):

Run “**sudo ufw allow 22/tcp**” (allows inbound SSH). Confirm with “**sudo ufw status**”.

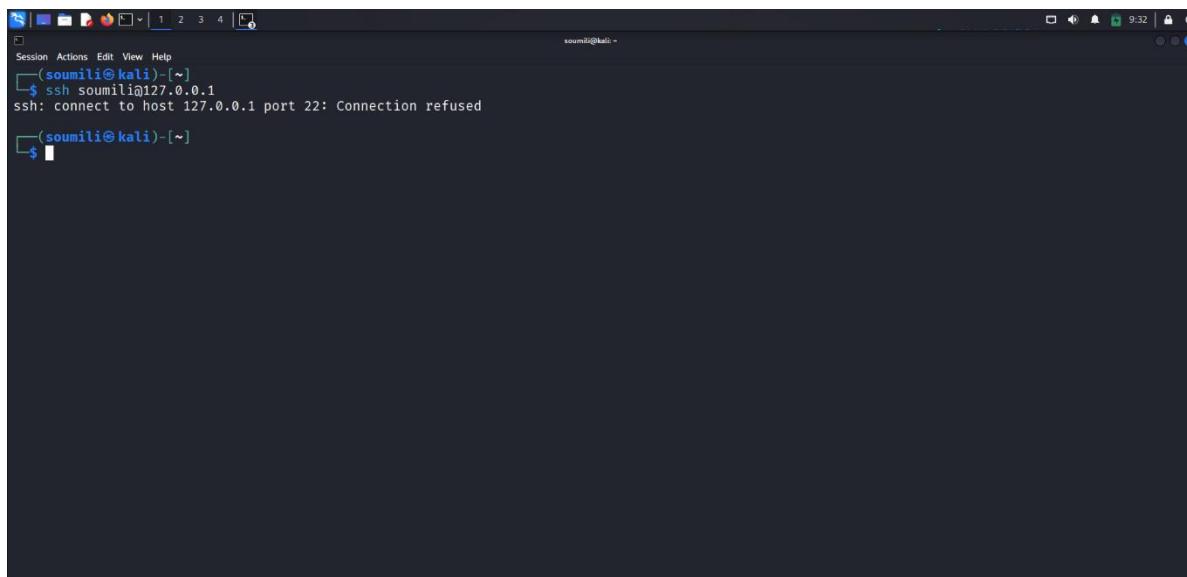


The screenshot shows a terminal window titled "soumili@kali ~". The user runs the command "sudo ufw allow 22/tcp" followed by "sudo ufw status". The output shows the following rules:

To	Action	From
--	DENY	Anywhere
22/tcp	ALLOW	Anywhere
23/tcp (v6)	DENY	Anywhere (v6)
22/tcp (v6)	ALLOW	Anywhere (v6)

Test by connecting via SSH from another machine: ssh user@<your-IP>

Result:



The screenshot shows a terminal window titled "soumili@kali ~". The user runs the command "ssh soumili@127.0.0.1". The output shows the connection attempt failed with the message "ssh: connect to host 127.0.0.1 port 22: Connection refused".

Connection Refused.

Remove the Test Block Rule to Restore Original State

On Linux (UFW):

Run “**sudo ufw delete deny 23/tcp**”. Confirm with “**sudo ufw status**”.

The screenshot shows a terminal window on a Kali Linux desktop environment. The terminal session starts with the command `sudo ufw delete deny 23/tcp`, followed by `sudo ufw status`. The output shows that the rule has been deleted, and the current status is active. The terminal also displays the default configuration for port 22/tcp.

```
(soumili㉿kali)-[~]
$ sudo ufw delete deny 23/tcp
$ sudo ufw status
Rule deleted
Rule deleted (v6)
Status: active

To                         Action      From
--                         ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere (v6)

(soumili㉿kali)-[~]
$
```

Document Commands or GUI Steps Used

Create a log file (**e.g., text document**) and record each step, including exact commands/GUI actions, timestamps, and outcomes.

Example:

Step 1: Opened firewall.cpl on Windows.

Step 2: Added inbound rule to block port 23 via GUI.

Step 3: Tested with nc -zv localhost 23; result: Connection refused.

This documentation helps track changes and troubleshoot.

Summarize How Firewall Filters Traffic

Firewalls act as a barrier between your system and the network, controlling inbound and outbound traffic based on rules. They filter packets by criteria like source/destination IP, port, protocol (e.g., TCP/UDP), and state (e.g., new vs. established connections). For example:

- **Inbound Filtering:** Blocks unwanted incoming connections (e.g., denying port 23 prevents Telnet access).
- **Outbound Filtering:** Restricts what your system can send out (less common but useful for malware prevention).
- **Default Behavior:** Most firewalls deny all inbound by default and allow outbound, requiring explicit "allow" rules for services like SSH.
- **Stateful Inspection:** Tracks connection states, allowing responses to outbound requests while blocking unsolicited inbound traffic. This layered approach enhances security by reducing attack surfaces, but over-restrictive rules can break functionality.

Thank You