

Configure and harden a host firewall on Linux with firewalld, ufw, and nftables

A default-deny inbound firewall, with safe SSH access, verification from another machine, and mappings to CIS and NIST guidance for Fedora, RHEL, Ubuntu, and Arch-based systems.

[FIREWALLD](#) · [UFW](#) · [NFTABLES](#) · [CIS- AND NIST-ALIGNED](#) · [HOW-TO GUIDE](#)

General educational guidance, not affiliated with or endorsed by the Center for Internet Security or NIST. On a remote host, allow SSH before you enable or tighten anything, and keep a second session open. Published 2026-09-07.

3 tools, one backend

firewalld, ufw, and nftables all program nftables

4 distributions covered

Fedora 44 · RHEL 9 and 10 · Ubuntu 26.04 · CachyOS

A host firewall decides which of your machine's network ports the rest of the world can reach. Without one, every service that listens on a network interface is reachable by any host that can route a packet to yours: other devices on the coffee-shop Wi-Fi, other tenants in a cloud network, or a compromised device on your LAN. A default-deny inbound firewall narrows that exposure to the ports you choose to publish.

CONTEXT Where a host firewall sits

Turn off what you do not need before you filter it

A service that is not running, or that listens only on `127.0.0.1`, cannot be attacked from the network no matter what the firewall says. Auditing what listens (Step 1) and turning off what you do not need is the primary control; the firewall covers what remains. This is NIST CM-7, and it is the reason Step 1 looks at listening sockets rather than firewall rules alone.

Deny inbound by default, allow outbound

Block everything inbound that you did not explicitly allow, and allow outbound. This maps to NIST SC-7 (boundary protection) and the CIS "Host Based Firewall" section.

A host firewall protects only the host it runs on

It protects the one host it runs on, and it is not an intrusion detection system (IDS). It complements a network firewall and separate SSH and DNS hardening; it does not replace them.

On a cloud VM, two firewalls are in the path

Your provider's security group or network ACL filters traffic before it reaches the instance. A port it blocks tests as closed no matter what the host firewall says, so a passing result in Step 3 does not by itself prove the host firewall is working. A port you open here stays unreachable until the security group opens it as well. Check both when a result surprises you.

Docker and rootful Podman can bypass the host firewall

Docker and rootful Podman manage their own firewall rules. When you publish a container's port (for example with `docker run -p 8080:80`), the runtime tells the kernel to send that port's traffic straight to the container, bypassing the firewalld or ufw rules that guard the host. The port is then reachable from the network even though your firewall never listed it.

To stay in control, publish container ports only to `127.0.0.1` (for example `-p 127.0.0.1:8080:80`). Where a port has to reach the network, filter it in Docker's own `DOCKER-USER` chain, which Docker's default iptables backend evaluates before its published-port rules and leaves in place across restarts. Either way, check what is reachable with `sudo ss -tulpn` (Step 1).

Run only one firewall manager

firewalld, ufw, and a hand-written nftables ruleset are three different front-ends that all write their rules into the same place in the kernel. Run two of them and the rules pile up on top of each other, which makes it hard to tell which rule is deciding what. Pick one manager per host and use only that one.

Lockout warning

READ THIS BEFORE YOU TOUCH A REMOTE HOST

On a remote machine, enabling a default-deny firewall before you allow SSH will cut off your session and leave you unable to reconnect. Every step that changes inbound filtering can lock you out if done out of order. Keep your current SSH session open, allow SSH **before** you enable or tighten the firewall (Step 2), test from a second terminal, and on a cloud VM find your provider's recovery console before you start.

MAP

Distribution cheat-sheet

The rule is the same everywhere: deny inbound by default, allow the handful of services you run, keep outbound open. Only the tool differs. Fedora and RHEL share tooling (firewalld, `firewall-cmd`), so they are grouped. The tool-specific steps are split into three tracks, one per tool: Track A (firewalld), Track B (ufw), and Track C (nftables). Follow the single track named in the last row of the table for your system.

	FEDORA 44 / RHEL 9, 10	UBUNTU 26.04 (DESKTOP/SERVER)	CACHYOS (ARCH-BASED)
Default firewall	firewalld	ufw	ufw
On by default?	yes, enabled and running	installed, inactive	yes, active after install
Kernel backend	nftables	nftables (via <code>iptables-nft</code>)	nftables (via <code>iptables-nft</code>)
Manage it with	firewall-cmd	ufw	ufw (or nftables / firewalld)
Follow	Track A	Track B (or Track C)	Track B (or Track C)

Fedora and RHEL enable firewalld by default, but the default zone differs

Fedora Workstation uses a permissive zone that opens a large port range; Fedora Server and RHEL do not. Track A deals with this directly.

Ubuntu installs ufw but leaves it inactive

A fresh Ubuntu host has no firewall filtering until you enable it. Do not enable it on a remote host before Step 2.

CachyOS enables ufw with a default-deny-inbound policy

A plain Arch install has no firewall at all. On CachyOS you are mostly allowing the services you run and verifying what is already in place.

They all share the same nftables backend

Underneath, all three tools store their rules in the kernel's nftables framework: firewalld and native nftables use it directly, and ufw reaches it through a compatibility layer called `iptables-nft` on both Ubuntu and CachyOS. Because they share that one backend, a hand-written nftables ruleset and ufw (or firewalld) can overwrite or clash with each other.

TOOLS What each tool is

firewalld

A zone-based front-end. Every network interface is assigned to a *zone*, and the zone decides what inbound traffic is allowed. It writes its rules through the nftables backend. It is the Fedora and RHEL default.

ufw ("Uncomplicated Firewall")

A rule-based front-end: you allow or deny services and ports with short commands and it generates the low-level rules for you. It is the Ubuntu and CachyOS default.

nftables

The native, in-kernel packet-filtering framework that both of the above write their rules into. You can also write an nftables ruleset by hand, which is the most transparent option and the one Arch provides a starter file for.

Choose the one your distribution already uses unless you have a reason not to.

```

      Network (Wi-Fi, cloud, LAN)
      |  inbound packets
      v
+-----+
| host firewall          | default-deny inbound:
| firewalld / ufw / nftables | established,related -> accept
| (one manager, nftables backend) | allowed ports      -> accept
+-----+-----+ everything else      -> drop
      | ssh tcp/22, https tcp/443
      v
      local services

loopback  iif "lo" (127.0.0.0/8, ::1) -> accept
outbound  output policy accept -> Network
```

The same default-deny inbound flow applies whichever tool you run: only allowed ports reach local services; loopback and outbound stay open.

PREP Prerequisites and conventions

- Administrative privileges (`sudo`) and basic familiarity with the terminal.
- On a **remote** host: an open SSH session you will keep, a second terminal to test with, and the location of your provider's recovery console.
- For Track C or the nftables checks: the `nftables` package (`nft`), present by default on Fedora, RHEL, and CachyOS and installable on Ubuntu with `sudo apt install nftables` .

• Keep a second terminal open

Do the work in your main SSH session, and keep a second terminal with a separate `ssh you@server` connection. Before you enable or tighten anything, confirm SSH is allowed (Step 2). After each change, open a *fresh* SSH connection in the second terminal to prove you can still get in. If a change breaks login, the original session is still connected, so you can undo it.

• Replace the placeholders

`you` is your login name, `server` is its address, and interface names such as `eth0` or `wlp2s0` should be replaced with the ones on your machine (list them with `ip -br link`).

• Reject or drop

A firewall refuses an unwanted packet by rejecting it or by dropping it. *Reject* sends back an error, so the sender learns the port is closed and gives up quickly. *Drop* stays silent, so the sender waits and retries, which slows down scanners but also slows down your own legitimate connection errors. `firewalld`'s default is reject; `ufw` and the nftables ruleset here drop. Both are fine; the note in each track explains how to switch.

• Runtime rules and permanent rules (`firewalld`)

`firewalld` keeps two configurations: the *runtime* rules currently in effect, and the *permanent* rules saved to disk that survive a reboot. A change is not permanent until you say so. This is easy to get wrong and is covered in Track A.

• If you do lock yourself out

A cloud provider gives you an out-of-band recovery console (labelled "Serial Console", "VNC console", or "Recovery mode", depending on the provider) that logs you in even when SSH is unreachable. Find yours before you start.

01 See what you have and what is listening

Two questions before you change anything: what firewall is already active, and what is listening on the network. Check the firewall state first:

```
# Fedora and RHEL (firewalld)
sudo firewall-cmd --state
sudo firewall-cmd --get-default-zone
sudo firewall-cmd --list-all

# Ubuntu and CachyOS (ufw)
sudo ufw status verbose
```

On Ubuntu this often prints `Status: inactive` , which means no `ufw` filtering yet. On CachyOS it usually prints `Status: active` with a default-deny incoming policy already in place.

Check what is listening. A port with nothing behind it is already closed; a port bound to `127.0.0.1` is not reachable from the network at all.

```
sudo ss -tulpn
```

An address of `127.0.0.1` or `::1` in the output is loopback-only and not exposed. An address of `0.0.0.0`, `*`, or `:::` is bound to every interface and *is* reachable from the network if the firewall lets it through. For anything in that second group that you do not need reachable, the better fix is to stop or reconfigure the service (NIST CM-7, least functionality) rather than only firewalling it.

02 Allow SSH before you tighten anything

On a remote host, do this **first**, before enabling ufw or changing firewalld zones. This is the step that prevents a lockout.

```
# Fedora and RHEL (firewalld): already running, so this takes effect on reload
sudo firewall-cmd --add-service=ssh --permanent
sudo firewall-cmd --reload

# Ubuntu (ufw is inactive): add the rule now, enable in Track B
sudo ufw limit OpenSSH

# CachyOS (ufw already active)
sudo ufw limit ssh
```

OpenSSH is an application profile ufw installs with `openssh-server`; `ssh` is the equivalent service name. Either opens TCP 22. `ufw limit` opens SSH and adds a light rate-limit on repeated connection attempts (explained in Track B); it still lets your own session through, so it is safe to run here. If you run SSH on a non-standard port, use that port number instead (for example `sudo ufw limit 2222/tcp`). Then follow the track for your distribution.

TRACK A firewalld: Fedora, RHEL, and anywhere firewalld runs

firewalld is already enabled and running on Fedora and RHEL. Your job is to put the active interface in a zone that denies inbound by default, allow only the services you run, and save it.

A1. Know your default zone, and the Fedora Workstation exception

```
sudo firewall-cmd --get-default-zone
sudo firewall-cmd --list-all
```

- RHEL 9 and 10 default to the `public` zone, which allows `ssh`, `dhcpv6-client`, and `cockpit` (the web console on TCP 9090).
- Fedora Server defaults to the `FedoraServer` zone, which allows the same three.
- Fedora Workstation defaults to the `FedoraWorkstation` zone, which is deliberately permissive. It allows `ssh`, `dhcpv6-client`, and `samba-client`, and it opens the entire `1025-65535` range on TCP and UDP so desktop apps work without per-app prompts. On a laptop that roams onto untrusted networks, that open high-port range is a real exposure: any application listening above port 1024 is reachable.

A representative `FedoraWorkstation` listing (an unmodified system shows the high-port range):

```
FedoraWorkstation (default, active)
target: default
interfaces: wlp2s0
services: dhcpv6-client samba-client ssh
ports: 1025-65535/udp 1025-65535/tcp
...
```

The `target: default` line decides what firewalld does with a packet that matches no rule (see A3).

A2. Choose a deny-by-default zone

On a server the default (`public` or `FedoraServer`) already denies inbound apart from its listed services, so you mainly need to prune that list (A4). On **Fedora Workstation**, close the open high-port range with either option below.

Option 1, switch the default zone to a strict one. The `public` zone denies the high ports and is a sensible desktop default:

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

`--set-default-zone` applies immediately and persists, no reload needed. After this, the `1025-65535` range is closed and the host keeps only the short list of services the `public` zone allows: `ssh` and `dhcpv6-client`, plus `mdns` on Fedora. Confirm the exact set with `sudo firewall-cmd --list-all`, and remove anything you do not need (A4). Some desktop sharing features will stop working until you allow their specific ports.

Option 2, keep the zone but remove the open range. If you rely on the Workstation zone's behavior but want the high ports closed:

```
sudo firewall-cmd --permanent --zone=FedoraWorkstation --remove-port=1025-65535/tcp
sudo firewall-cmd --permanent --zone=FedoraWorkstation --remove-port=1025-65535/udp
sudo firewall-cmd --reload
```

A3. What firewalld does with unmatched traffic

A firewalld zone with `target: default` **rejects** unsolicited inbound packets that match no rule: it replies with a "prohibited" error using ICMP (the network's control-message protocol), while still allowing ICMP itself. That already denies inbound by default. If you would rather drop silently, set the target to `DROP`:

```
sudo firewall-cmd --permanent --zone="$(sudo firewall-cmd --get-default-zone)" --set-target=DROP
sudo firewall-cmd --reload
```

This targets whatever zone is currently the default; substitute an explicit zone name if you manage a different one. `--set-target` is permanent-only, so it needs the `--reload` to take effect. Silent drop hides the host from casual scans but also delays legitimate error feedback; reject gives the sender faster feedback and both deny inbound.

CIS RHEL 9 v3.0.0 takes a stricter line at Level 2: its 4.1.8 asks for the default zone to be the built-in `drop` zone, which is a different mechanism from the target set here and drops everything the zone does not name. Use it if you are working to Level 2.

A4. Allow only what you run, remove the rest

List the current services, keep SSH, and remove anything you do not use. Cockpit is a common one to drop on a machine you never administer through the web console:

```
# Keep SSH (harmless to repeat)
sudo firewall-cmd --permanent --add-service=ssh

# Remove services you do not need (examples)
sudo firewall-cmd --permanent --remove-service=cockpit
sudo firewall-cmd --permanent --remove-service=samba-client

# Open a specific port you do need (example: a web server)
sudo firewall-cmd --permanent --add-service=https

sudo firewall-cmd --reload
```

Add services by name where firewalld knows them (`--add-service=https`), or by port where it does not (`--add-port=8080/tcp`). List the known service names with `firewall-cmd --get-services` .

A5. Runtime rules and permanent rules

This is the most common firewalld mistake. Commands **without** `--permanent` change only the running configuration and are lost on the next reload or reboot. Commands **with** `--permanent` change the saved configuration but do **not** take effect until `firewall-cmd --reload` . Pick one of these approaches:

- Make every change `--permanent` and finish with `--reload` (the approach used above).
- Experiment with runtime commands until it works, then snapshot the runtime into permanent with `sudo firewall-cmd --runtime-to-permanent` .

A6. Pin the zone to the interface (NetworkManager)

On NetworkManager-managed systems, bind the zone to the connection profile so a reconnect cannot reset it. Use the zone you settled on in A2: `public` if you took Option 1, `FedoraWorkstation` if you took Option 2.

```
sudo nmcli connection modify "<connection-name>" connection.zone <zone-name>
```

List connection names with `nmcli connection show` .

A7. Log what gets refused

```
sudo firewall-cmd --set-log-denied=unicast
```

This records refused unicast packets to the kernel log, which is the firewalld equivalent of `ufw logging on` in Track B. `all` also records broadcast and multicast, which is noisy on a busy network. The setting is runtime and permanent at once and reloads the firewall itself, so it needs no `--permanent` and no separate `--reload` . Turn it off again with `--set-log-denied=off` . This is CIS RHEL 9 v3.0.0 4.1.10 at Level 2, which accepts any value except `off` .

A8. Verify

```
sudo firewall-cmd --get-active-zones
sudo firewall-cmd --get-default-zone
sudo firewall-cmd --list-all
sudo nft list ruleset | grep -A3 'table inet firewalld' # the generated backend rules
```

Read `--get-active-zones` first: it names the zone your interface is in, which is the zone that decides what gets through. If that is not the zone you chose in A2, `--list-all` is reporting a zone nothing is using, because `--set-default-zone` moves only the interfaces that are following the default. Then you want the services list to contain only what you allow, and (on Workstation) no `1025-65535` port range. Reboot once and run these again: a change that was never made permanent looks correct until then.

TRACK B ufw: Ubuntu and CachyOS

ufw is the front-end on both. On CachyOS it is already active; on Ubuntu you will enable it. The commands are identical.

B1. Set the default policies

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
```

Together these deny inbound by default: nothing gets in unless a rule allows it, and everything outbound is permitted. Allowing outbound is the ordinary choice and what the rest of this guide assumes. CIS asks for `ufw default deny outgoing` at Level 2, which blocks everything the host tries to reach until you allow it: DNS, NTP, HTTP, HTTPS, and package updates at a minimum. Take that route only if you are working to Level 2 and can enumerate what the host needs to reach.

B2. Allow the services you run

You already allowed and rate-limited SSH in Step 2 with `ufw limit`. Add anything else you expose:

```
# Allow other services as needed (examples)
sudo ufw allow 443/tcp          # HTTPS
sudo ufw allow from 192.168.1.0/24 to any port 445 proto tcp # SMB, LAN only
```

The `limit` rule you used for SSH is a light brute-force rate limit, defined by ufw as: allow the connection normally, but deny an address that opens six or more connections in thirty seconds. Key-only SSH authentication is the stronger control; this rule mainly cuts scanner noise.

Keep a single SSH rule: ufw applies the first rule that matches a packet, so a second SSH rule under a different name (an `allow` stacked on top of the `limit`, say) can mask the limit.

Watch the output for `Skipping unsupported IPv6 'limit' rule`, which ufw prints where the kernel has no IPv6 `recent` match. ufw then adds no IPv6 rule at all for that port, so once you enable ufw the port is closed to IPv6 while IPv4 keeps the rate limit. On a host you reach over IPv6, use `sudo ufw allow OpenSSH` instead before you enable; ufw applies `allow` to both families.

B3. Enable ufw

On **Ubuntu**, enable it now that SSH is allowed. On **CachyOS** it is already active, so this just confirms it.

```
sudo ufw enable
```

- **Do not run this on a remote Ubuntu host until Step 2 has allowed SSH**

Enabling ufw with `deny incoming` and no SSH rule drops your session.

B4. Turn on logging

```
sudo ufw logging on
```

`low` (the default once logging is on) records blocked packets. Levels go up to `full`; `medium` is a reasonable middle ground: `sudo ufw logging medium`.

B5. IPv6

ufw filters IPv6 as well as IPv4 when `IPv6=yes` is set in `/etc/default/ufw`, which is the default. If you edited that file, reload ufw so it re-reads it (`sudo ufw disable && sudo ufw enable`). Leave IPv6 enabled unless you have a specific reason not to.

B6. Verify

```
sudo ufw status verbose
```

You want:

- `Status: active`.
- `Default: deny (incoming), allow (outgoing)`.
- A rule list containing SSH and only the services you meant to open.

The Default line ends with the routed default. On a host that does not pass traffic between interfaces, `disabled (routed)` or `deny (routed)` is what you want, and Ubuntu installs it as `deny`, so there is usually nothing to change; set it with `sudo ufw default deny routed` if it reads otherwise. Where it reads `disabled (routed)`, that is the kernel's forwarding state rather than the ufw policy, so confirm the policy itself with `grep -i routed /etc/default/ufw`; `DEFAULT_FORWARD_POLICY` is `DROP` as installed, which is the value CIS asks for. This is CIS Ubuntu 4.1.5.

`ufw status numbered` shows rule numbers if you need to delete one with `sudo ufw delete <number>`. Reboot once and run this again: it is the quickest proof the rules survive a restart.

TRACK C nftables: write the ruleset yourself

Writing nftables directly is the most transparent option: one file holds the whole ruleset, and no front-end translates it on your behalf. Arch, CachyOS and Ubuntu all install a starter `/etc/nftables.conf` to build on. **Do not run this alongside ufw or firewalld**; pick one. On CachyOS, disable ufw first (`sudo systemctl disable --now ufw`) if you switch to nftables.

- **Check this before you start: containers and virtual machines**

The ruleset below drops everything in the `forward` chain, which is right for a host that only receives traffic for itself and wrong for one that passes traffic through. Docker, rootful Podman, libvirt, and bridged virtual machines all rely on forwarding, and this ruleset cuts it, so containers lose their outbound network.

The file also opens with `flush ruleset`, which clears every table in the kernel, including the ones those runtimes install for themselves. Adding forward rules does not bring those back, and `nftables.service` reloads the file at every boot. On a host that runs containers or virtual machines, stay with the firewall manager that runtime already integrates with.

C1. A minimal, safe ruleset

Put this in `/etc/nftables.conf`. It denies inbound by default, keeps established connections working, allows loopback (and blocks spoofed loopback), permits ICMP (including IPv6 neighbor discovery, without which IPv6 breaks), lets a DHCPv6 reply through, and allows SSH:

```
#!/usr/sbin/nft -f
flush ruleset

table inet filter {
    chain input {
        type filter hook input priority 0; policy drop;

        ct state invalid drop
        ct state established,related accept

        iif "lo" accept
        iif != "lo" ip saddr 127.0.0.0/8 drop
        iif != "lo" ip6 saddr ::1 drop
        iif != "lo" ip daddr 127.0.0.0/8 drop
        iif != "lo" ip6 daddr ::1 drop

        ip protocol icmp accept
        meta l4proto ipv6-icmp accept

        ip6 daddr fe80::/64 udp dport 546 accept

        tcp dport 22 accept
    }

    chain forward {
        type filter hook forward priority 0; policy drop;
    }

    chain output {
        type filter hook output priority 0; policy accept;
    }
}
```

The two loopback pairs do different jobs, and the one ordering requirement is that `iif "lo" accept` comes first.

- The `saddr` pair is the anti-spoofing rule: it drops packets arriving on a real interface that claim to come from the loopback network. CIS RHEL 9 v2.0.0 checked for that rule at 4.3.4. No current CIS Linux benchmark covers a hand-written nftables ruleset, so that number is the closest thing to a citation this track has; the ALIGN section says why.
- The `daddr` pair drops packets from outside addressed to that network; the kernel normally discards those as martians anyway, so it is defense in depth.

The `udp dport 546` rule lets a DHCPv6 reply reach the host, which is the same traffic firewalld's default zones allow as `dhcpv6-client`. A network that hands out IPv6 addresses with SLAAC does not need it, because the ICMPv6 rule above already covers router advertisements, but a network running stateful DHCPv6 leaves this host without an IPv6 address if you drop the line.

Add a line per service you expose, for example `tcp dport 443 accept` for HTTPS.

Arch, CachyOS and Fedora install the `nft` binary at `/usr/bin/nft`, so change the first line to `#!/usr/bin/nft -f` there; on Fedora the `/usr/sbin` path still resolves through a symlink, so either form works. The shebang (the `#!` first line) is used only if you run the file directly rather than loading it with `nft -f`.

• Non-standard SSH port

If your SSH server does not listen on port 22, change `tcp dport 22 accept` to your port before you load the file. Loading it and rebooting with the wrong port locks you out, the same risk Step 2 flags for ufw.

• Rate-limiting SSH

This ruleset accepts SSH without the rate limit the ufw track adds with `ufw limit`, and key-only SSH authentication is the stronger control either way. To add a limit, nftables counts new connections per source address in a dynamic set; the nftables wiki page "Meters" gives the set declaration and the matching rule. Write a second rule keyed on `ip6 saddr` if the host answers SSH on IPv6 as well: a rule keyed on `ip saddr` matches IPv4 only, so a single-rule version never limits an IPv6 client.

• Logging drops

Dropped inbound traffic is silent here. To record it, add `log prefix "nftables input drop: " counter drop` as the last rule in the `input` chain: the rule logs and counts each unmatched packet and drops it in the same step, so nothing is left for the chain's `policy drop` to act on.

C2. Load and persist

```
# Check the syntax without touching the running ruleset
sudo nft -c -f /etc/nftables.conf

# Load it now (an error applies nothing; the load is atomic)
sudo nft -f /etc/nftables.conf

# Enable at boot
sudo systemctl enable --now nftables
```

The `nftables.service` loads `/etc/nftables.conf` at boot. It is disabled by default on Ubuntu and needs the `enable` above; on Arch and CachyOS the same command applies.

C3. Verify

```
sudo nft list ruleset
```

You want to see `policy drop` on the `input` and `forward` chains and your accept rules for loopback, established connections, ICMP, and SSH.

03 Test the firewall from another machine

Confirm from another machine on the same network, if you have one, that the ports you meant to close are closed:

```
# From a SECOND machine, replace with your host's address.
# An allowed port connects. A blocked port fails: fast with a "prohibited" error
# if the firewall rejects (firewalld's default), or by timing out if it drops
# silently (ufw and the nftables ruleset here, or firewalld set to DROP).
nc -vz server 22      # SSH: expect success
nc -vz server 9090    # Cockpit on RHEL/Fedora Server: expect it to fail
```

From the host itself, a quick sanity check that SSH survived and the default is deny:

```
# Fedora/RHEL
sudo firewall-cmd --list-all
# Ubuntu/CachyOS
sudo ufw status verbose
# nftables
sudo nft list ruleset
```

`nc -vz` tests TCP from wherever you ran it. A UDP port and a test from outside your own network are separate checks, and on a cloud VM the provider's security group answers first either way (see the note on page 1). Only close your fallback (the second terminal, the recovery console tab) once a fresh SSH login succeeds.

ALIGN CIS and NIST alignment

This guide follows the "Host Based Firewall" section of the CIS Linux Benchmarks and supports the boundary-protection and least-functionality controls of **NIST SP 800-53 Rev 5**. It covers the checks in the table below and does not cover every recommendation in those sections, so run the benchmark's own audit before you rely on the result. **NIST SP 800-41 Rev 1**, *Guidelines on Firewalls and Firewall Policy*, is the background reference and explicitly covers host-based firewalls. CIS recommendation numbers are version-specific, so verify against the exact benchmark you must meet.

CIS RHEL 9 Benchmark v3.0.0 (2026-08-28)

Section 4.1, *Configure firewalld*, running 4.1.1 to 4.1.11. v3.0.0 cut section 4 back to firewalld alone, so every RHEL 9 firewall number in v2.0.0 (2024-06-24) moved. Three checks are new, all at Level 2: 4.1.8 (default zone `drop`, see A3), 4.1.10 (log denied packets, see A7) and 4.1.11 (block ICMP timestamp requests and replies). 4.1.9 on conflicting utilities, at Level 1, restates v2.0.0's 4.1.2, scoped to firewalld.

CIS RHEL 10 Benchmark v1.0.1 (2025-09-30)

Section 4.1, *Configure firewalld*. RHEL 10 covers the firewalld path only, and 4.1.1 to 4.1.7 match RHEL 9 v3.0.0.

CIS Ubuntu 26.04 LTS Benchmark v1.0.0 (2026-08-26), and 24.04 LTS v2.0.0 (2026-05-28)

Section 4.1, *Configure Uncomplicated Firewall*. The column below is the 26.04 numbering, and 24.04 v2.0.0 carries the same 4.1.1 to 4.1.5; 24.04 LTS is supported to 2029 rather than replaced. Both editions cut this section back to the ufw path, so several cells read "no ufw check": the benchmark has no equivalent recommendation.

ACTION IN THIS GUIDE	CIS (RHEL PATH)	CIS (UBUNTU UFW PATH)	NIST 800-53 R5
Use exactly one firewall utility	RHEL 9 4.1.9	no ufw check	CM-7
Firewall service enabled and running	RHEL 9 4.1.3; RHEL 10 4.1.3	Ubuntu 4.1.2	SC-7
Default-deny inbound (zone target / default policy)	RHEL 9 4.1.4; RHEL 10 4.1.4; zone target	Ubuntu 4.1.3	SC-7, SC-7(5), SC-7(12)
Loopback traffic handled correctly	RHEL 9 4.1.5, 4.1.6; RHEL 10 4.1.5, 4.1.6	no ufw check	SC-7
Allow only needed services, drop the rest	RHEL 9 4.1.7; RHEL 10 4.1.7	no ufw check	CM-7, CM-7(1)
Log refused packets (A7 / B4 / the C1 note)	RHEL 9 4.1.10, Level 2	(site policy)	AU-2, AU-12
Consider outbound policy (this guide allows outbound; CIS asks for deny at Level 2)	(site policy)	Ubuntu 4.1.4, Level 2	AC-4, CM-7
Routed and forwarded default policy	(site policy)	Ubuntu 4.1.5	SC-7
nftables base chains and default-deny policy	gone in v3.0.0 (v2.0.0 4.3.1, 4.3.3)	no ufw check	SC-7
Audit listening services (least functionality)	(site policy)	(site policy)	CM-7

In NIST terms, SC-7 (Boundary Protection) is the parent control: its definition of a "managed interface" explicitly includes firewalls. Enhancement SC-7(5) is the deny-by-default, allow-by-exception rule the whole guide sets up, and SC-7(12), *Host-based Protection*, is the control that names a firewall running on the host it protects. CM-7 (Least Functionality) is the reason Step 1 audits listening services and the tracks remove services you do not use rather than leaving them running behind a closed port.

- **No current CIS benchmark covers Fedora, CachyOS, or Arch**

Fedora has no current CIS benchmark (the old Fedora 28 benchmark is archived); adapt the RHEL benchmark or the now-archived CIS Distribution Independent Linux Benchmark. CachyOS and Arch have no CIS benchmark at all, so their coverage here is by analogy rather than certification.

- **No current CIS benchmark covers a hand-written nftables ruleset**

Every current edition above covers one front-end and nothing else: RHEL 9 v3.0.0, RHEL 10 and AlmaLinux 10 check firewalld, and both Ubuntu editions check ufw. CIS RHEL 9 v2.0.0 was the last to carry checks for a hand-written nftables ruleset, and v3.0.0 removed them. Track C still meets the same outcomes, and the NIST controls in the last column still apply to it, but if you have to show a CIS result, run one of the front-end tracks instead.

HELP Troubleshooting

- **Locked out of a remote host after enabling the firewall**

Use the provider's serial or recovery console, then:

- On firewalld: `sudo firewall-cmd --add-service=ssh then --runtime-to-permanent` .
- On ufw: `sudo ufw limit OpenSSH` .
- On nftables: add `tcp dport 22 accept` to the input chain and reload.

That is the reason Step 2 comes before any tightening and the reason you keep a second terminal open.

- **A firewalld change did not take effect**

A `--permanent` change needs `firewall-cmd --reload` . A runtime change is lost on reload. Decide which you meant (see A5).

- **Two firewall managers are both active**

Running ufw or firewalld *and* a hand-written nftables ruleset produces overlapping rules and confusing behavior. Pick one. Check for strays: `systemctl is-enabled firewalld ufw nftables` .

- **IPv6 is not being filtered**

In ufw, confirm `IPv6=yes` in `/etc/default/ufw` . In nftables, use the `inet` family (as the ruleset here does) so one table covers IPv4 and IPv6. A host reachable over IPv6 with only IPv4 rules is a common and serious gap, and it is the reason B5 and the ruleset in C1 both keep IPv6 in scope.

- **A service you expected to reach is blocked**

Confirm it is both listening (`sudo ss -tulpn`) and allowed (the verify command for your track). A service bound only to `127.0.0.1` will never be reachable regardless of firewall rules.

- **nft -f reports a syntax error**

Nothing was applied; the load is atomic. Fix the reported line and reload. Your running rules are unchanged until a clean load succeeds.

ROLLBACK How to roll back

```
# firewalld, if you took A2 Option 1: go back to your original default zone
# (no reload needed; --set-default-zone applies at once and persists)
sudo firewall-cmd --set-default-zone=FedoraWorkstation # or your original zone
```

```
# firewalld, if you took A2 Option 2: put the high-port range back
sudo firewall-cmd --permanent --zone=FedoraWorkstation --add-port=1025-65535/tcp
sudo firewall-cmd --permanent --zone=FedoraWorkstation --add-port=1025-65535/udp
sudo firewall-cmd --reload
```

```
# ufw: disable, or reset all rules to the installed default
sudo ufw disable
sudo ufw reset # wipes rules; back to an inactive default-install ufw
```

```
# nftables: stop loading the custom ruleset
sudo systemctl disable --now nftables
sudo nft flush ruleset      # clears the loaded rules now; the disabled service will not reload them
```

On a remote host, roll back in the same careful order: make sure SSH is still reachable at each step, and keep the second terminal open until a fresh login works.

DONE What you achieved

- Inbound traffic is **denied by default**, and only the services you chose are reachable from the network.
- On Fedora Workstation, the permissive **1025-65535** port range is closed.
- Outbound traffic stays open, and established connections keep working.
- The firewall runs on the **nftables** backend through the tool your distribution provides.
- On Track A or Track B, your firewall covers the CIS "Host Based Firewall" checks listed in the table on page 12, apart from the Level 2 outbound policy this guide leaves open. Track C reaches the same outcomes and has no current CIS numbers to cite. All three support NIST SC-7 and CM-7, and you audited what listens instead of only firewalling it.

REF References

firewalld documentation and **firewall-cmd(1)** manual · firewalld.org/documentation

ufw(8) manual (Ubuntu) and Ubuntu Server firewall guide · [manpages.ubuntu.com](https://manpages.ubuntu.com/manpages/ubuntuserver/docs) · ubuntu.com/server/docs

nftables project wiki (including its "Meters" page) and Arch Wiki: nftables · [wiki.nftables.org](https://wiki.nftables.org/wiki.nftables.org) · wiki.archlinux.org/title/Nftables

Docker packet filtering and firewalls, and the DOCKER-USER chain · docs.docker.com/engine/network/packet-filtering-firewalls · docs.docker.com/engine/network/firewall-iptables

CIS Benchmarks (RHEL 9 v3.0.0, RHEL 10 v1.0.1, Ubuntu 26.04 v1.0.0, Ubuntu 24.04 v2.0.0, and RHEL 9 v2.0.0 for the nftables numbers) · cisecurity.org/cis-benchmarks

NIST SP 800-53 Rev 5, and SP 800-41 Rev 1: Guidelines on Firewalls and Firewall Policy · csrc.nist.gov