anoBytes | TECHNOLOGY SOLUTIONS

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 2026-07-20.

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

Least functionality comes first

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 why Step 1 looks at listening sockets rather than firewall rules alone.

Default-deny inbound is the core posture

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 is not a network firewall or an IDS

It protects the one host it runs on. It complements a network firewall and separate SSH and DNS hardening; it does not replace them.

Container runtimes can open ports around it

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`) or restrict them with the runtime's own network settings, and check what is actually 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 actually 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 posture is the same everywhere: deny inbound by default, allow the handful of services you actually 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 `iptables-nft`)	nftables
Manage it with	`firewall-cmd`	`ufw`	`ufw (or nftables / firewalld)`
Follow	Track A	Track B (or Track C)	Track B

Fedora and RHEL ship firewalld enabled, 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 ships ufw installed but inactive

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

CachyOS ships ufw active with a default-deny-inbound policy

This is unlike vanilla Arch, which ships 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`. Because they share that one backend, a hand-written nftables ruleset and ufw (or firewalld) can overwrite or clash with each other, so run only one of them per host.

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.

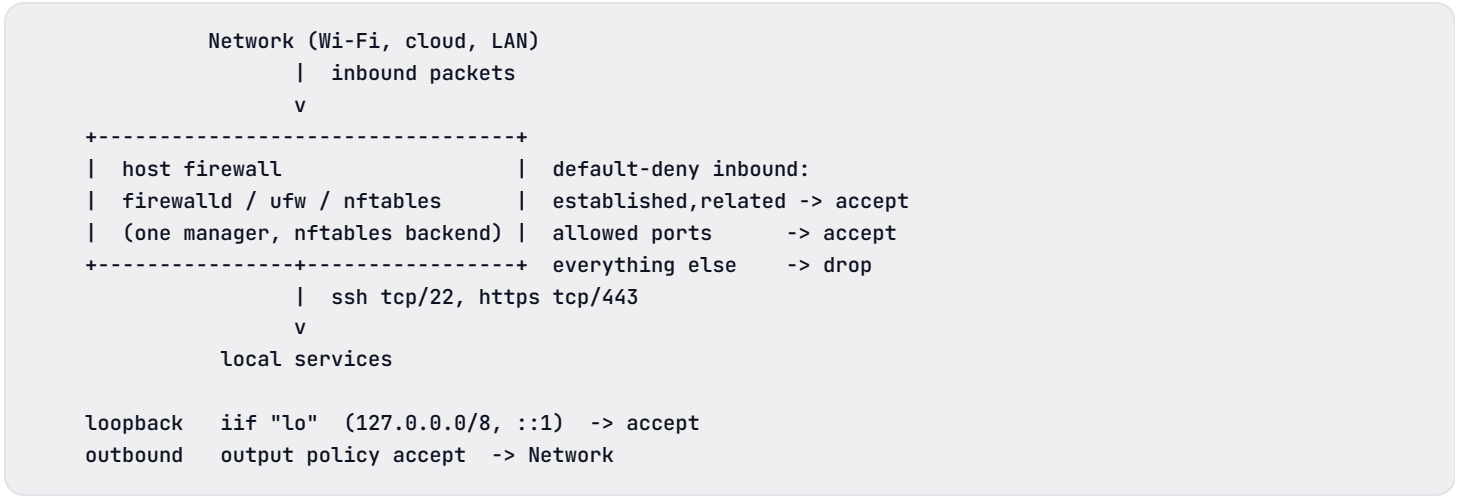
ufw ("Uncomplicated Firewall")

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 ultimately use. You can also write an nftables ruleset by hand, which is the most transparent option and the one Arch ships a starter file for.

Choose the one your distribution already uses unless you have a reason not to, and run only that one.



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` , `wlp2s0` , or `tailscale0` should be replaced with the ones on your machine (list them with `ip -br link`).

- **reject vs drop**

  A firewall can refuse an unwanted packet two ways. *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 vs permanent (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 actually 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 ships when `openssh-server` is installed; `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

```
firewall-cmd --get-default-zone
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 is the key to how unmatched traffic is treated (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 in one of two ways.

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 `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. Understand how unmatched traffic is treated

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 is already a

deny-by-default posture. If you would rather drop silently, set the target to `DROP`:

```
sudo firewall-cmd --permanent --zone="$(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 is the friendlier default and both deny inbound.

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 vs permanent, in practice

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`. Two ways to keep things straight: make every change `--permanent` and finish with `--reload` (the approach used above), or 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:

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

List connection names with `nmcli connection show`.

A7. Verify

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

You want the default zone to be the strict one, the services list to contain only what you allow, and (on Workstation) no `1025-65535` port range.

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
```

This is the deny-by-default posture: nothing inbound unless a rule allows it, everything outbound permitted.

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.

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; a host reachable over IPv6 with only IPv4 rules is a common gap.

B6. Verify

```
sudo ufw status verbose
```

You want `Status: active`, `Default: deny (incoming), allow (outgoing)`, and a rule list containing SSH and only the services you meant to open. `ufw status numbered` shows rule numbers if you need to delete one with `sudo ufw delete <number>`.

TRACK C nftables: the native option

Writing nftables directly is the most transparent option: one file holds the whole ruleset, and no front-end translates it on your behalf. It suits Arch and CachyOS (which ship a starter `/etc/nftables.conf`) and anyone on Ubuntu who prefers

native rules to ufw. **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.

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), 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

        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 order matters only in 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, which is what CIS checks for (RHEL 9 4.3.4). 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. Add a line per service you expose, for example `tcp dport 443 accept` for HTTPS. On Arch and CachyOS the `nft` binary is at `/usr/bin/nft`, so change the first line to `#!/usr/bin/nft -f`; the shebang (the `#!` first line) only matters if you run the file directly rather than loading it with `nft -f`.

Three things to adjust or add for this ruleset:

- **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`. Key-only SSH authentication is the stronger control; to also slow repeated attempts, add a per-source-IP `meter` rule for new SSH connections.

- **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

```
# Load it now (nft reports any syntax error and applies nothing)
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 Prove it from the outside

Rules that look right can still be wrong. Confirm the posture from another machine if you have one on the same network:

```
# 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, if you removed it: expect it to fail
```

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

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

Only close your fallback (the second terminal, the recovery console tab) once a fresh SSH login succeeds.

ALIGN CIS and NIST alignment

This guide implements the **"Host Based Firewall"** section of the CIS Linux Benchmarks and the boundary-protection and least-functionality controls of **NIST SP 800-53 Rev 5**. 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, and CIS restructured this section between recent releases, so verify against the exact benchmark you must meet.

CIS RHEL 9 Benchmark v2.0.0 (2024-06-24)

Section 4, *Host Based Firewall*, with subsections for firewalld and nftables.

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

Section 4.1, *Configure firewalld*. RHEL 10 covers the firewalld path only.

CIS Ubuntu 24.04 LTS Benchmark v2.0.0 (2026-05-28)

Section 4, *Host Based Firewall*, subsection 4.1, *Configure Uncomplicated Firewall*. v2.0.0 restructured this section to the ufw path only, and removed the nftables and iptables subsections and several standalone ufw checks that v1.0.0 had. The numbers below are v2.0.0; if you must meet the older v1.0.0 (2024-08-26), its numbering differs and is shown in parentheses where it helps.

Fedora and Arch-based systems

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.

ACTION IN THIS GUIDE	CIS (RHEL FIREWALLD PATH)	CIS (UBUNTU UFW PATH)	NIST 800-53 R5
Use exactly one firewall utility	RHEL 9 4.1.2	dropped in v2.0.0 (v1.0.0 4.1.1)	CM-7
Firewall service enabled and running	RHEL 10 4.1.3	Ubuntu 4.1.2 (v1.0.0 4.2.3)	SC-7
Default-deny inbound (zone target / default policy)	RHEL 10 4.1.4; zone `target`	Ubuntu 4.1.3 (v1.0.0 4.2.7)	SC-7, SC-7(5), SC-7(12)
Loopback traffic handled correctly	RHEL 9 4.2.2 (firewalld), 4.3.4 (nftables); RHEL 10 4.1.5, 4.1.6	dropped in v2.0.0 (v1.0.0 4.2.4)	SC-7
Allow only needed services, drop the rest	RHEL 9 4.2.1; RHEL 10 4.1.7	dropped in v2.0.0 (v1.0.0 4.2.6)	CM-7, CM-7(1)
Consider outbound policy	(site policy)	Ubuntu 4.1.4 (v1.0.0 4.2.5)	AC-4
Routed and forwarded default policy	(site policy)	Ubuntu 4.1.5	SC-7
nftables base chains and default-deny policy	RHEL 9 4.3.1, 4.3.3	none in v2.0.0 (v1.0.0 4.3.5, 4.3.8)	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 posture the whole guide sets up, and SC-7(12), *Host-based Protection*, is the direct hook for a firewall that runs 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.

HELP Troubleshooting

- **Locked out of a remote host after enabling the firewall**
  Use the provider's serial or recovery console. 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. This is why Step 2 comes before any tightening and why 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.

- **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: revert to the shipped default zone and reload
sudo firewall-cmd --set-default-zone=FedoraWorkstation   # or your original zone
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 ships, with exactly one manager in charge.

- The posture maps to the CIS "Host Based Firewall" controls and to NIST SC-7 and CM-7, and you audited what actually 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` · `ubuntu.com/server/docs`

nftables project wiki and Arch Wiki: nftables · `wiki.nftables.org` · `wiki.archlinux.org/title/Nftables`

CIS Benchmarks (RHEL 9 v2.0.0, RHEL 10 v1.0.1, Ubuntu 24.04 v2.0.0) · `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`