



ANCHOR CYBER SECURITY LLC

Linux & Server Hardening Field Reference

Baseline auditing, file integrity monitoring, and intrusion prevention — what to actually check, independent of which tool does the checking.

```
$ harden --baseline-audit
```

Security Baseline Audit

What an actual baseline audit checks — the goal is a CIS-aligned configuration review, not a vendor tool's default scan output.

- ❑ Kernel parameters — ASLR enabled, core dumps restricted, IP forwarding disabled unless needed
- ❑ SSH configuration — root login disabled, password auth disabled in favor of keys, non-default port considered
- ❑ File permissions — no world-writable files outside expected paths (`/tmp` , `/var/tmp`), SUID/SGID binaries reviewed against a known-good list
- ❑ Open ports & listening services — every listener accounted for, nothing running that isn't needed
- ❑ Account & password policy — password aging enforced, no unused accounts with shell access, no shared accounts
- ❑ Sudo configuration — least privilege, no blanket `NOPASSWD: ALL` entries without a real reason
- ❑ Unattended upgrades / automatic patching — actually configured, not just assumed

Common miss: a clean scan today means nothing six months from now. A baseline audit without a re-audit cadence is a snapshot, not a program.

```
$ harden --file-integrity
```

File Integrity Monitoring

The goal: know the moment a system binary or config file changes, without so much noise that the alerts get ignored.

WHAT TO ACTUALLY BASELINE

- ❑ `/etc` — system-wide configuration; changes here are rarely accidental
- ❑ `/bin`, `/sbin`, `/usr/bin`, `/usr/sbin` — core system binaries
- ❑ `/boot` — kernel and boot configuration
- ❑ Cron directories and systemd unit files — a common persistence target

The part that actually matters: automate re-baselining after every routine patch cycle. A FIM tool that fires a false positive on every `apt upgrade` gets disabled by the client within a month — that's not a tooling problem, it's a process problem, and it's on whoever set it up to solve.

```
$ harden --intrusion-prevention
```

Automated Intrusion Prevention

Local, log-driven defense against brute-force and credential-stuffing attempts.

LOG SOURCES TO WATCH

- ❑ `/var/log/auth.log` (Debian/Ubuntu) or `/var/log/secure` (RHEL/Fedora)
- ❑ Web application login endpoints, if internet-facing
- ❑ Any other service accepting authenticated connections from outside the LAN

BEFORE YOU ENABLE ANYTHING

- ❑ Allowlist known-good IPs first — office IP, VPN range, admin workstations
- ❑ Confirm the firewall backend in use — `firewalld` vs. `nftables` / `iptables` directly
- ❑ Tune ban thresholds after a few days live — defaults are often wrong for a given environment

The mistake we see most: enabling brute-force protection before allowlisting admin access. The single most common way a hardening pass locks a client out of their own server.

```
$ harden --quick-wins
```

A Baseline You Can Run Today

No tool install required — built-in commands that surface real findings in minutes.

```
# unexpected accounts with UID 0 (root-equivalent)
awk -F: '($3 == 0) {print}' /etc/passwd

# what's actually listening on the network
ss -tulnp

# recent successful logins — who, when, from where
last -a | head -20

# SSH config: confirm root login and password auth are off
grep -E "^(PermitRootLogin|PasswordAuthentication)" /etc/ssh/sshd_config
```

Sources: CIS Benchmarks · Anchor hardening engagement notes

```
$ echo $?
0
```