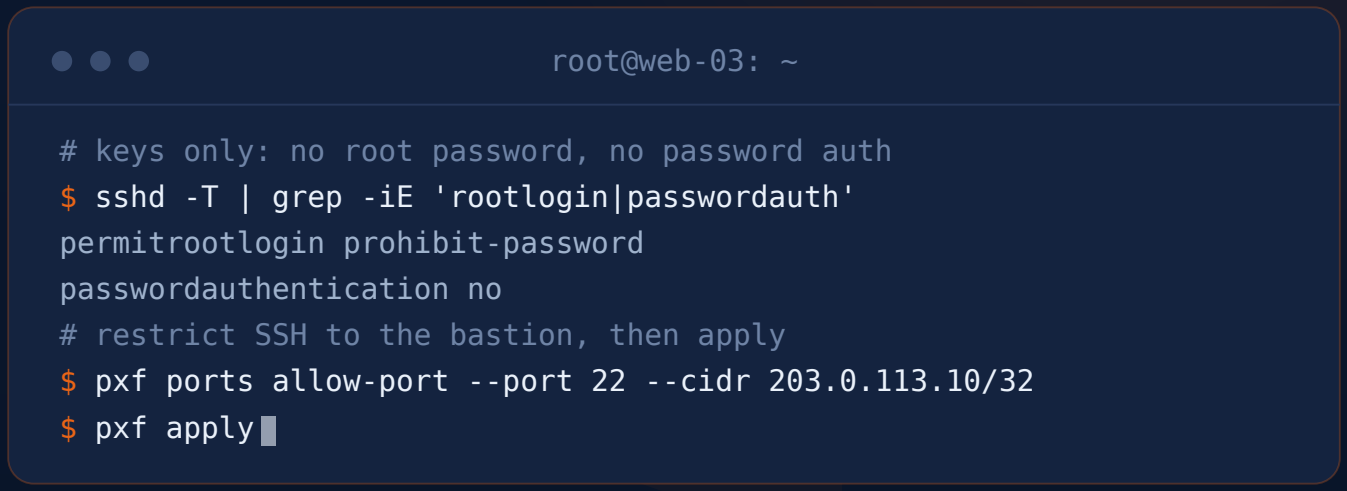
CentralHost

SERVER HARDENING GUIDE

How to Harden Your cPanel Server

18 ordered steps · a copy-paste checklist · real-world examples

A field guide for hosting providers and sysadmins securing cPanel servers — one box or a fleet of two hundred.



www.centralhost.sh

2026 Edition

Contents

Introduction

A freshly installed cPanel server is not a secure server.

Default configurations prioritize compatibility over security: the SSH port is wide open, firewall rules are permissive, plain-text panel ports are listening, and the WAF is either off or running a minimal ruleset. On one server that's a manageable afternoon of cleanup. On a fleet of 20, 50, or 200 — where new servers are provisioned constantly, configurations drift, and team members come and go — staying hardened stops being a task and becomes an operational discipline.

This guide is that discipline written down: **18 concrete steps, ordered by impact**, from OS-level SSH configuration through firewall, WAF, panel hardening, email authentication, backups, and auditing. Every step is actionable, tested in production, and designed not to break client sites. At the end you'll find a **copy-paste checklist** to run against any server before signing it off as production-ready, plus an **appendix of real-world examples** — the kind of incidents you actually get paged for.

A note on where this comes from. We build CentralHost, a platform for managing cPanel fleets, so you'll see it referenced where it answers a question a section raises: fleet-wide visibility, an SSH access map, an AI assistant that investigates incidents. Read those as the *"how do I do this across 200 servers"* answer. Everything else stands on its own — the hardening is identical whether you run CentralHost or not.

And one opinion up front, because it shapes several of the steps: **the most effective thing you can do for SSH isn't to harden it — it's to stop exposing it.** Closing port 22 to the public internet entirely (Steps 4 and 5) removes a whole category of attack instead of mitigating it. The guide keeps coming back to that idea.

Part 1 — Keep cPanel Updated

Step 1: Enable automatic updates on the STABLE tier

The most impactful single thing you can do before touching anything else is ensure the server stays patched automatically. cPanel manages its own update lifecycle — the OS, the panel itself, EasyApache components, and bundled packages like PHP and MySQL all go through cPanel's update mechanism. Fighting that with external tools causes more problems than it solves.

In **WHM → Update Preferences**, set:

- **Update Tier:** STABLE — not EDGE or CURRENT in production. STABLE receives updates after they have been validated across the cPanel fleet; the few days of lag are worth the stability.
- **Automatic Updates:** Enabled — cPanel will apply updates on its own schedule without manual intervention.

That is it for OS-level maintenance. cPanel handles the rest. Schedule a reboot window after kernel updates — cPanel services restart cleanly on boot and the process is straightforward.

Part 2 — SSH

Step 2: Generate SSH keys and test before restricting root login

Before setting `PermitRootLogin prohibit-password`, you must verify that key authentication actually works. Locking yourself out of root on a production server because the key was never tested is one of the most avoidable incidents in this guide.

Follow this order strictly: generate key → install on server → verify login → then restrict password.

Generate your SSH key pair

Linux / macOS

Open a terminal on your local machine:

```
ssh-keygen -t ed25519 -C "your-name@yourcompany"
```

When prompted for a file location, press Enter to accept the default (`~/.ssh/id_ed25519`). Set a passphrase — this protects the key if your machine is compromised.

This creates two files:

- `~/.ssh/id_ed25519` — your private key (never share this)
- `~/.ssh/id_ed25519.pub` — your public key (this goes on the server)

Windows

On Windows 10/11, OpenSSH is built in. Open PowerShell and run:

```
ssh-keygen -t ed25519 -C "your-name@yourcompany"
```

The keys are saved to `C:\Users\YourName\.ssh\`. If you use PuTTY instead of OpenSSH, use PuTTYgen to generate an ed25519 key and export the public key in OpenSSH format.

Install the public key on the server

Linux / macOS — use `ssh-copy-id`:

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub root@your-server-ip
```

This command appends your public key to `/root/.ssh/authorized_keys` on the server.

Windows (PowerShell) — no `ssh-copy-id` available natively, so do it manually:

```
# Read your public key
$pubkey = Get-Content "$env:USERPROFILE\.ssh\id_ed25519.pub"

# Append it to the server's authorized_keys (you'll be prompted for the root password)
ssh root@your-server-ip "mkdir -p ~/.ssh && chmod 700 ~/.ssh && echo '$pubkey' >>
~/.ssh/authorized_keys && chmod 600 ~/.ssh/authorized_keys"
```

Verify key login — before changing anything in sshd_config

Open a **new terminal window** (keep your current session open) and test the key login explicitly:

```
# Linux / macOS
ssh -i ~/.ssh/id_ed25519 root@your-server-ip

# Windows PowerShell
ssh -i $env:USERPROFILE\.ssh\id_ed25519 root@your-server-ip
```

You should log in without being prompted for a password (only for the key passphrase if you set one). **Do not proceed to the next step until this works.**

Now harden sshd_config

Only after confirming key login works, edit `/etc/ssh/sshd_config`:

```
# Root login allowed, but via SSH key only — never password
PermitRootLogin prohibit-password

# Disable password authentication — keys only
# No client access breaks: clients use FTP/FTPS, not SSH (see Step 10)
PasswordAuthentication no

# Disable empty passwords
PermitEmptyPasswords no

# Limit login attempts per connection
MaxAuthTries 3

# Disconnect idle sessions after 5 minutes (300s × 1)
ClientAliveInterval 300
ClientAliveCountMax 1

# Disable X11 forwarding
X11Forwarding no
```

Restart SSH — again, keeping your current session open while testing:

```
systemctl restart sshd
```

Open a new terminal and confirm you can still log in as root with your key. Only then close the previous session.

On password authentication: on a cPanel server where clients use FTP/FTPS for file transfers, SSH carries only administrative traffic from your team. There is no reason to keep `PasswordAuthentication yes`. Set it to `no` — no client access breaks, and brute-force attacks against SSH passwords stop being a concern entirely. Combined with `PermitRootLogin prohibit-password`, the SSH configuration accepts keys only, for everyone.

> **On a fleet:** CentralHost's security posture view surfaces servers where `PermitRootLogin` is not restricted to keys, across your entire fleet — so configuration drift is visible without logging into each server individually.

Step 3: One key per operator, never shared

Each person who needs root SSH access should have their own key pair. Never share a private key between team members.

Keep a clean `/root/.ssh/authorized_keys` on each server — only keys that are currently active. Audit it regularly and remove keys belonging to former employees, contractors who have finished their work, and decommissioned systems.

```
# Review current authorized keys on root
cat /root/.ssh/authorized_keys

# Check for keys added under hosting accounts
find /home -name authorized_keys 2>/dev/null
```

For temporary access (contractors, external support), use CentralHost's delegated SSH access — scoped, time-limited, audited, and revocable without touching `authorized_keys` on the server.

Step 4: Reduce or eliminate port 22 exposure

Port 22 open to the public internet is a constant source of brute-force noise. There are two practical approaches:

Option A — Close port 22 entirely (recommended if using CentralHost)

CentralHost gives you two ways to access your servers without an open port 22:

- **Browser-based SSH terminal.** A full SSH session directly in your browser, authenticated through CentralHost. No local SSH client required, works from anywhere.
- **SSH bastion.** CentralHost provides a bastion host — a single, hardened entry point through which all SSH connections to your fleet are routed. Instead of connecting directly to each server, your SSH client connects to the bastion, which proxies the session to the target server over the encrypted agent channel. From the operator's perspective it behaves like a normal SSH connection; from the server's perspective, no inbound port 22 traffic ever arrives from the internet.

Both methods allow you to close port 22 to the public internet via PXF — with one difference: if you use the bastion, add CentralHost's bastion IP as an explicit allow rule in PXF before closing port 22. The web terminal has no such requirement, as it does not connect inbound to your server's SSH port directly.

Client file transfers use FTP/FTPS on port 21 (see Step 10), so closing port 22 has no impact on hosting clients.

Option B — Restrict port 22 to known IPs

If closing SSH entirely is not immediately feasible, restrict port 22 to your office IP or VPN endpoint via PXF firewall rules and block everything else. This eliminates internet-wide exposure while preserving direct access from trusted locations.

In practice, this approach requires either a static IP at every location your team works from, or a VPN with a fixed egress IP. Without one of those, the allowlist becomes difficult to maintain — dynamic IPs change, team members work remotely, and every exception punches another hole in the rule. This is precisely why the bastion model gains traction: instead of managing a moving list of trusted IPs, you whitelist a single fixed IP — the bastion — and access your entire fleet through it from anywhere.

Either option is a substantial improvement over a wide-open port 22. If neither is immediately feasible, PXF's brute-force protection (Step 5) combined with `PermitRootLogin prohibit-password` and `PasswordAuthentication no` removes the meaningful attack vectors while you plan the migration.

Part 3 — Firewall

Step 5: Install and configure PXF

For over a decade the default answer here was **CSF (ConfigServer Security & Firewall)**. That era is over: CSF/LFD is no longer actively maintained and has reached end of life, with no further development from ConfigServer. Running an abandoned firewall as your front line is not a security posture — unpatched, it ages into a liability. If you still have CSF on your servers, plan a migration off it.

PXF is a free, actively maintained firewall built specifically for cPanel servers — a direct replacement for CSF in that role. It integrates natively with WHM, the command line, and CentralHost, and supports both `iptables` and `nftables`, covering the full range of current cPanel-supported distributions.

Install PXF by following the instructions at pyxsoft.com. Once installed, it is available in WHM as a plugin.

PXF closes everything by default and automatically opens the ports that a cPanel server needs — HTTP, HTTPS, SMTP, IMAP, POP3, FTP, and the cPanel/WHM panel ports. You do not need to configure a list of ports to open or close manually. The cPanel-aware defaults are applied on installation.

The only decisions you need to make after installing PXF are the exceptions:

SSH port

PXF detects the SSH port configured on the server and keeps it open by default. To restrict or close it, you need to explicitly configure an exception after installation. Depending on how your team accesses servers:

- **CentralHost web terminal:** the web terminal works over an outbound connection from the agent — it is not affected by the SSH port being open or closed. You can safely close it.
- **CentralHost bastion:** restrict the SSH port to the bastion's IP only. Your team connects to the bastion from anywhere and the bastion connects inbound to your server — only that single IP needs to be allowed.
- **Direct SSH from fixed IPs:** restrict the SSH port to your office or VPN IP.

Plain-text panel ports (2082, 2086, 2095)

PXF opens these by default for compatibility. If you enforce HTTPS on cPanel, WHM, and Webmail (Step 11), close these in PXF — there is no reason to keep plain-text panel access open.

Enable brute-force protection in PXF settings. PXF monitors failed login attempts against SSH and automatically blocks offending IPs. This is only relevant if the SSH port is open — if you have closed it or restricted it to a single trusted IP, SSH brute-force is already a non-issue at the network level. Brute-force protection for cPanel, WHM, FTP, and email is handled separately by cPHulk (Step 12).

Because PXF integrates with CentralHost, the rules above don't have to be set server by server: you manage ports, exceptions, and IP blocks across the whole fleet from one panel, without logging into each WHM or opening a terminal.

Step 6: Block known bad IPs and networks

PXF supports IP blocklists. Enable automatic blocklist updates to keep known malicious networks blocked without manual intervention.

Also consider blocking traffic from regions you do not serve, if your client base is geographically limited. PXF's country blocking feature handles this at the firewall level.

Part 4 — Web Application Firewall

Step 7: Replace ModSecurity with pxShield

ModSecurity is the default WAF engine on cPanel servers. It works, but has well-known operational pain points: it runs inside Apache's request processing pipeline, false positives are opaque and difficult to tune, and rule management requires direct config file access. The dominant commercial option, **Imunify360**, does not escape this — its WAF is a ruleset and cloud reputation layer running *on top of the same ModSecurity engine*, so it inherits the in-Apache execution model (its separate Proactive Defense component is PHP-level runtime protection, not the WAF). Whether you run plain ModSecurity or Imunify360, filtering still happens inside Apache, consuming worker processes.

pxShield is a WAF built specifically for cPanel hosting environments. The key architectural difference is not the rules but *where it runs*: **pxShield sits as a reverse proxy in front of Apache**, inspecting and filtering traffic before it ever reaches your web server.

Practical advantages:

- **False positive management with full UI.** When a legitimate request is blocked, you can review, whitelist, and tune rules directly from WHM or CentralHost — no config file editing required.
- **Full visibility in WHM and CentralHost.** Blocked requests, active rules, traffic patterns and alerts are visible in both interfaces. From CentralHost you get a fleet-wide view across all servers.
- **Better performance.** Filtering before Apache means rejected requests never consume Apache worker processes.
- **Consistent ruleset.** Rules and tuning are managed centrally and applied uniformly.
- **Client-facing visibility.** pxShield adds a plugin to each client's cPanel account showing active blocks and WAF activity on their site. Clients can see that their hosting provider has active security measures in place — a tangible trust signal that most shared hosting providers do not offer.

Install pxShield by following the instructions at pyxsoft.com. After installation, enable it per domain or globally, then review the default ruleset and tune any false positives before putting it in full blocking mode.

> **Across the fleet:** the question that's hard to answer at scale isn't "is the WAF tuned?" — it's "which of my servers have no WAF coverage at all?" CentralHost's security posture panel answers exactly that, listing pxShield status, blocked-request counts, and active alerts per server, so a server that slipped through provisioning without a WAF stops being invisible.

Part 5 — cPanel and WHM Specific

Step 8: Restrict WHM access by IP

WHM (port 2087) should not be reachable from arbitrary IP addresses. The most reliable method on a cPanel server is to block port 2087 at the firewall level via PXF: allow only your known IPs, deny everything else. This approach survives cPanel updates and does not depend on any cPanel-specific configuration layer.

In PXF, add a rule to restrict port 2087 (and 2086 if you have not yet closed it) to your admin IP range. For example, if your team accesses WHM from a single office IP:

```
# Restrict WHM (2087) to your admin IP, then apply the change.
# Scoping allow-port to a --cidr opens the port only for that source.
pxf ports allow-port --port 2087 --proto tcp --cidr YOUR.ADMIN.IP/32
pxf apply
```

Alternatively, use **WHM → Security Center → Host Access Control** to set TCP wrapper rules — but be aware that TCP Wrappers (`/etc/hosts.allow` / `/etc/hosts.deny`) have been deprecated since RHEL 8 and only work for services that are explicitly compiled with libwrap support, which excludes most modern services. PXF firewall rules are the correct and reliable method here.

If your team accesses WHM from dynamic IPs, the practical alternative is a VPN: route all WHM access through a fixed VPN endpoint, then whitelist that endpoint in PXF.

Step 9: Enable two-factor authentication on WHM

Require 2FA for all WHM accounts. Go to **WHM → Two-Factor Authentication** and enforce it for root and all reseller accounts.

This is particularly important if WHM is accessible over the public internet.

Step 10: Disable unused cPanel services

In **WHM → Service Manager**, disable services your clients do not use:

- **Anonymous FTP** — disable entirely. There is no legitimate use case for anonymous FTP on a shared hosting server.
- **FTP** — keep it enabled for client file access. FTP with TLS (FTPS) is the standard client file transfer method on cPanel servers. The choice of FTPS over SFTP here is **operational, not a security ranking** — both are encrypted. FTPS runs on a dedicated port (21), which keeps client file transfer decoupled from SSH so you can restrict or close port 22 without affecting clients; it is also universally supported by tools like FileZilla and maps cleanly to cPanel's per-account FTP users. Ensure Pure-FTPd (cPanel's default FTP daemon) is configured to require TLS connections (FTPS without TLS is plain-text — that is the part that actually matters for security).
- **Cpsrvd** plain-text port (2082) — disable if you enforce HTTPS on the panel.

Keeping client file access on FTP/FTPS rather than SFTP also simplifies SSH hardening: the SSH port carries only administrative traffic (your team, automation), not client connections, making it far easier to restrict by IP or close entirely.

Step 11: Force HTTPS on cPanel, WHM, and Webmail

In **WHM → Tweak Settings**, enable:

- Always redirect to SSL
- Require SSL for cPanel, WHM, and Webmail

Also ensure your server's hostname has a valid SSL certificate. cPanel's AutoSSL handles this automatically — verify it is enabled and running in **WHM → Manage AutoSSL**.

Step 12: Configure cPHulk brute-force protection

cPHulk is cPanel's built-in brute-force protection layer for the panel itself — it monitors failed login attempts to WHM, cPanel, Webmail, FTP, and SSH and blocks IPs that exceed the configured thresholds. Enable and configure it in **WHM → Security Center → cPHulk Brute Force Protection**.

Recommended settings:

- **Number of failures that trigger a two-week brute force IP block:** 15
- **Number of failures that trigger a two-week brute force account block:** 15
- **Number of failures per IP address before an IP is reported:** 25
- **Look-back timeframe (minutes):** 15

One critical step before enabling: **whitelist your own admin IPs** under the Whitelist tab. If you fat-finger your own password three times from your office IP, cPHulk will block you out of WHM. Add your office IP and any other fixed admin IPs to the whitelist before turning brute-force protection on. PXF's brute-force protection operates at the network layer before cPHulk, so both run together effectively — PXF drops repeated attacks at the firewall, cPHulk handles application-level failures that get through.

Part 6 — Email Security

Step 13: Enforce SPF, DKIM, and DMARC for all domains

Email authentication is both a security and deliverability requirement. On cPanel servers, DKIM and SPF are managed at the domain level — cPanel generates the keys and publishes the DNS records automatically when a domain is added, provided the server is also the authoritative DNS server for that domain.

Enable DKIM and SPF globally so they are active for all new accounts by default: go to **WHM → Email → Email Authentication** and ensure both DKIM and SPF are enabled. This controls the default for newly created accounts.

Audit existing domains in bulk via **WHM → Email → Email Deliverability**. This view shows every domain on the server with the status of its SPF, DKIM, and DMARC records, and offers a one-click repair for domains where the records are missing or incorrect. On a server with many accounts, this is the practical way to fix gaps without touching each account individually.

For domains where the DNS is hosted externally (not on the cPanel server), cPanel cannot publish the records automatically — the zone needs to be updated manually at the DNS provider. The Email Deliverability page shows the exact records to add.

DMARC is not set automatically by cPanel. Add a DMARC record per domain starting with monitoring mode:

```
_dmarc.yourdomain.com  TXT  "v=DMARC1; p=none; rua=mailto:dmarc-reports@yourdomain.com"
```

Move to `p=quarantine` and then `p=reject` after reviewing the aggregate reports and confirming no legitimate mail sources are missing from SPF/DKIM.

Auditing this one server at a time via Email Deliverability is fine for a handful of domains; across a fleet it doesn't scale. CentralHost's email operations panel rolls up SPF/DKIM/DMARC status and blacklist presence per domain across every server, so the domains with authentication gaps surface as a list to work through rather than something you discover when mail starts bouncing.

Step 14: Set per-account email sending limits

Compromised cPanel accounts are a common spam source. Set hourly email sending limits per account to contain the blast radius:

In **WHM → Tweak Settings → Mail**:

- Max hourly emails per domain: 200 (adjust based on your client base)
- Maximum percentage of failed or deferred messages: 20%

When an account hits the limit, cPanel suspends mail sending for that account automatically. You will see the spike in CentralHost's monitoring before it becomes a deliverability problem for your other clients.

Part 7 — Backups

Step 15: Configure remote backup destinations

Local backups on the same server are not backups — they are copies that disappear with the server. Configure remote backup destinations:

In **WHM → Backup Configuration**:

- Enable backups
- Set a remote destination: S3-compatible storage, SFTP, or a dedicated backup server
- Retain at minimum: daily backups for 7 days, weekly for 4 weeks

Test restore procedures periodically. A backup you have never restored is a backup you cannot trust.

> **The failure mode this prevents:** backups fail silently, and you find out the day a client asks for a restore. CentralHost turns that around — its backup panel shows status across the whole fleet (which servers succeeded, which failed, which accounts have no coverage at all) and a failed backup becomes an alert *before* it becomes an incident.

Part 8 — Monitoring and Auditing

Step 16: Know what is running on your servers

cPanel does not have a dedicated "process monitoring" toggle in WHM. What it does have is useful if you know where to look:

WHM → Server Status → Service Status shows the health of all cPanel-managed services at a glance — httpd, exim, dovecot, mysql, named, ftpd, and others. Check this first when something behaves unexpectedly.

WHM → Server Status → CPU/Memory/MySQL Usage gives a current snapshot of resource consumers. For historical metrics, cPanel includes Munin integration if enabled during setup.

For process-level visibility that goes beyond what WHM surfaces, the relevant system logs on a cPanel server are:

```
# Failed SSH and su attempts
/var/log/secure

# Exim mail logs — spam, delivery failures, relay abuse
/var/log/exim_mainlog
/var/log/exim_rejectlog

# cPanel service events
/usr/local/cpanel/logs/error_log

# Apache access and error per domain (under /home/user/logs/)
# Plus the main Apache error log:
/etc/apache2/logs/error_log
```

For fleet-wide visibility — spotting a server with an unusual process, a spike in exim queue depth, or a cron job added by a compromised account — this is where centralized monitoring matters:

This is where CentralHost stops being a dashboard and starts doing the work. Its **AI Operations Assistant** investigates anomalies directly on any server in your fleet — you ask in plain language (*"Why is web-03 consuming 90% memory?"*, *"Is there anything unusual in the exim logs on this server?"*) and it reads the logs, correlates the metrics, and explains what it found. Nothing changes on the server unless you approve it. The full appendix at the end of this guide walks through real investigations.

Step 17: Audit SSH access regularly

Review who has SSH access to each server periodically. Check:

```
# List users with shell access
grep -v '/sbin/nologin\|/bin/false' /etc/passwd

# Check authorized_keys for root and all users
find /root /home -name authorized_keys -exec cat {} \;

# Recent successful SSH logins
last -n 50

# Recent failed SSH attempts
grep "Failed password" /var/log/secure | tail -50
```

Remove any authorized keys that belong to former employees, contractors, or systems no longer in use.

Running those commands by hand on every server is exactly the chore that doesn't get done consistently. CentralHost's security posture panel keeps a standing SSH access map for the whole fleet — who has shell access on each server, which keys are authorized, and which users can escalate to root — so the audit above is something you read, not something you have to run.

Step 18: Enable audit logging for critical changes

`auditd` provides kernel-level logging of file changes and system calls. On cPanel servers it is worth enabling, but the rule set needs to be conservative: a blanket rule logging every command run by root (`-S execve -F euid=0`) generates an enormous volume of events on an active server — Apache, Exim, MySQL, and cPanel's own daemons all run processes as root or with elevated privileges constantly. The audit log fills fast and becomes impossible to review.

Install and enable auditd:

```
dnf install audit -y
systemctl enable --now auditd
```

Use targeted rules that capture meaningful changes without flooding the log. Create `/etc/audit/rules.d/cpanel-hardening.rules`:

```
# Track changes to user and authentication files
-w /etc/passwd -p wa -k passwd_changes
-w /etc/shadow -p wa -k shadow_changes
-w /etc/sudoers -p wa -k sudoers_changes
-w /etc/sudoers.d/ -p wa -k sudoers_changes

# Track SSH configuration changes
-w /etc/ssh/sshd_config -p wa -k sshd_config

# Track cPanel configuration changes
-w /etc/cpanel/ -p wa -k cpanel_config
-w /usr/local/cpanel/etc/ -p wa -k cpanel_config

# Track cron modifications (common persistence vector after compromise)
-w /var/spool/cron/ -p wa -k cron_changes
-w /etc/cron.d/ -p wa -k cron_changes

# Track changes to PXF and firewall rules
-w /etc/pxf/ -p wa -k firewall_changes
```

Load the new rules:

```
augenrules --load
```

These rules log configuration changes, not every command — which is what matters for auditing. If you need command-level logging for specific users (e.g. resellers with shell access), scope it by UID rather than enabling it globally.

> **Attribution, not just logging:** `auditd` records what changed on the host; it can't tell you *who* did it once several people share root. Every action taken *through* CentralHost — commands in the SSH terminal, AI assistant actions, delegated access sessions — is logged with full attribution: who ran it, when, on which server, and what the server returned. The two layers complement each other.

Hardening Checklist

Use this as a quick reference before signing off on a server as production-ready:

cPanel Updates

- [ ] Update tier set to STABLE in WHM → Update Preferences
- [ ] Automatic updates enabled

SSH

- [ ] Root password authentication disabled — `PermitRootLogin prohibit-password`
- [ ] Password authentication disabled — `PasswordAuthentication no`
- [ ] `PermitEmptyPasswords no`
- [ ] Port 22 closed or restricted to known IPs via PXF
- [ ] MaxAuthTries set to 3
- [ ] Idle timeout configured
- [ ] authorized_keys audited — one key per operator, no shared or stale keys

Firewall

- [ ] PXF installed and configured
- [ ] Unnecessary ports closed
- [ ] Brute-force protection enabled
- [ ] Known bad IP blocklists active

WAF

- [ ] pxShield installed and in blocking mode
- [ ] False positives reviewed and tuned
- [ ] WAF coverage verified across all hosted domains

cPanel / WHM

- [ ] WHM access restricted by IP
- [ ] Two-factor authentication enforced on WHM
- [ ] Anonymous FTP disabled
- [ ] FTP/FTPS enabled for clients — TLS required in Pure-FTPd
- [ ] Plain-text panel ports (2082, 2086, 2095) disabled
- [ ] HTTPS enforced on all panel interfaces
- [ ] cPHulk brute-force protection enabled

Email

- [ ] SPF enabled for all domains
- [ ] DKIM signing enabled globally
- [ ] DMARC records published

- [ ] Per-account sending limits configured

Backups

- [ ] Remote backup destination configured
- [ ] Backup retention policy set
- [ ] Restore procedure tested

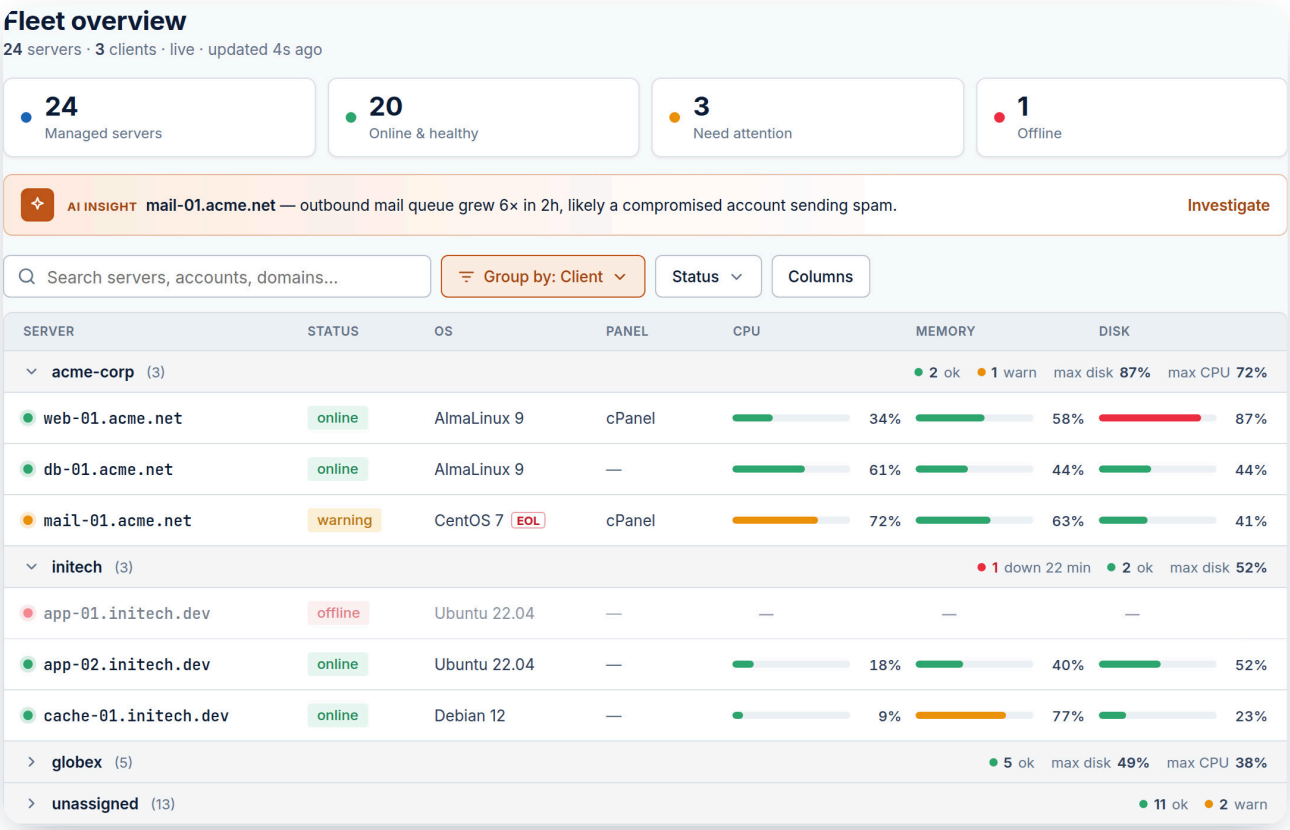
Monitoring and Auditing

- [ ] Process and login monitoring active
- [ ] SSH authorized keys audited
- [ ] auditd installed and rules configured

Managing This Across a Fleet

Running through this checklist once on one server is straightforward. The operational challenge is maintaining it consistently across tens or hundreds of servers over time — as configurations drift, new servers are added, and team members come and go.

CentralHost centralizes the security posture of your entire fleet: weak SSH configurations, missing WAF coverage, failed backups, exposed ports, email authentication gaps, and suspicious processes all surface as prioritized findings — without logging into each server. When something needs fixing, the AI Operations Assistant investigates and proposes the remediation. You approve the change.



The fleet view: per-server status grouped by client, and an AI insight already flagging the outbound spam — all without logging into a single box.

The first server is free, no credit card required: centralhost.sh

---

Last updated: June 2026

Appendix I: AI applied to systems administration

Ask your server.

For decades, diagnosing a server has meant the same thing: SSH in, open five or six logs, chain `grep`, `tail` and `awk`, and rebuild by hand what happened. It works, but it doesn't scale — and at 3 a.m., with a client waiting, it's slow and error-prone.

CentralHost flips that around: an AI assistant with access to every server in your fleet, which you ask in plain language. It reads the logs, correlates the metrics, inspects the processes and explains what it found — with the concrete lines of evidence, not a vague answer.

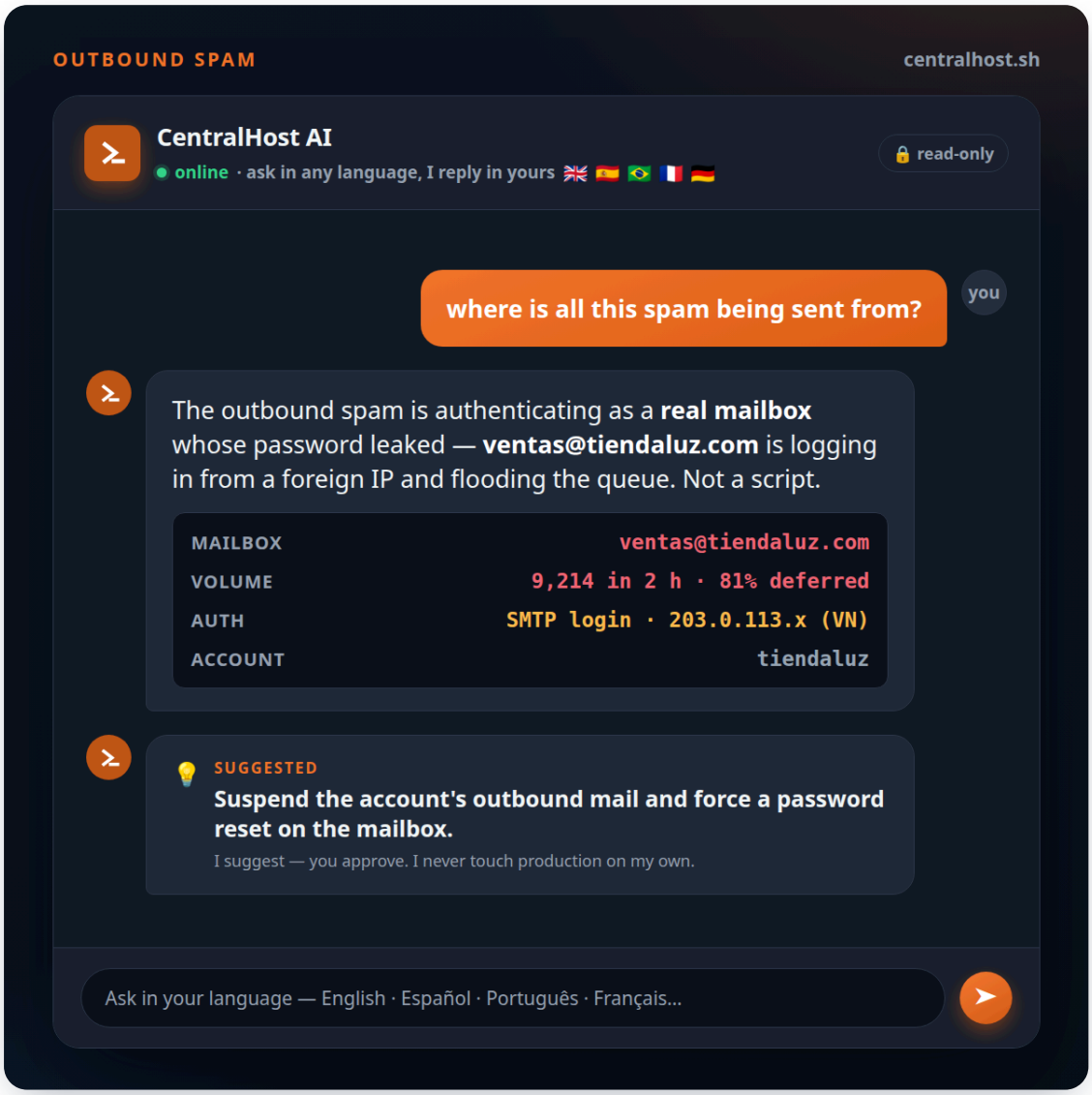
Two things before the examples:

- **It reaches your servers over the agent's outbound channel**, not an open port. It works even with port 22 closed (Step 4) — in fact, that's the point.
- **It operates in two modes, and you're in charge.** It *investigates* read-only: it reads, correlates and proposes, changing nothing. When you ask it to *act*, every action requires your explicit approval — one action, one approval; never in batches, never silently.

Below, three real investigations exactly as they look in the product.

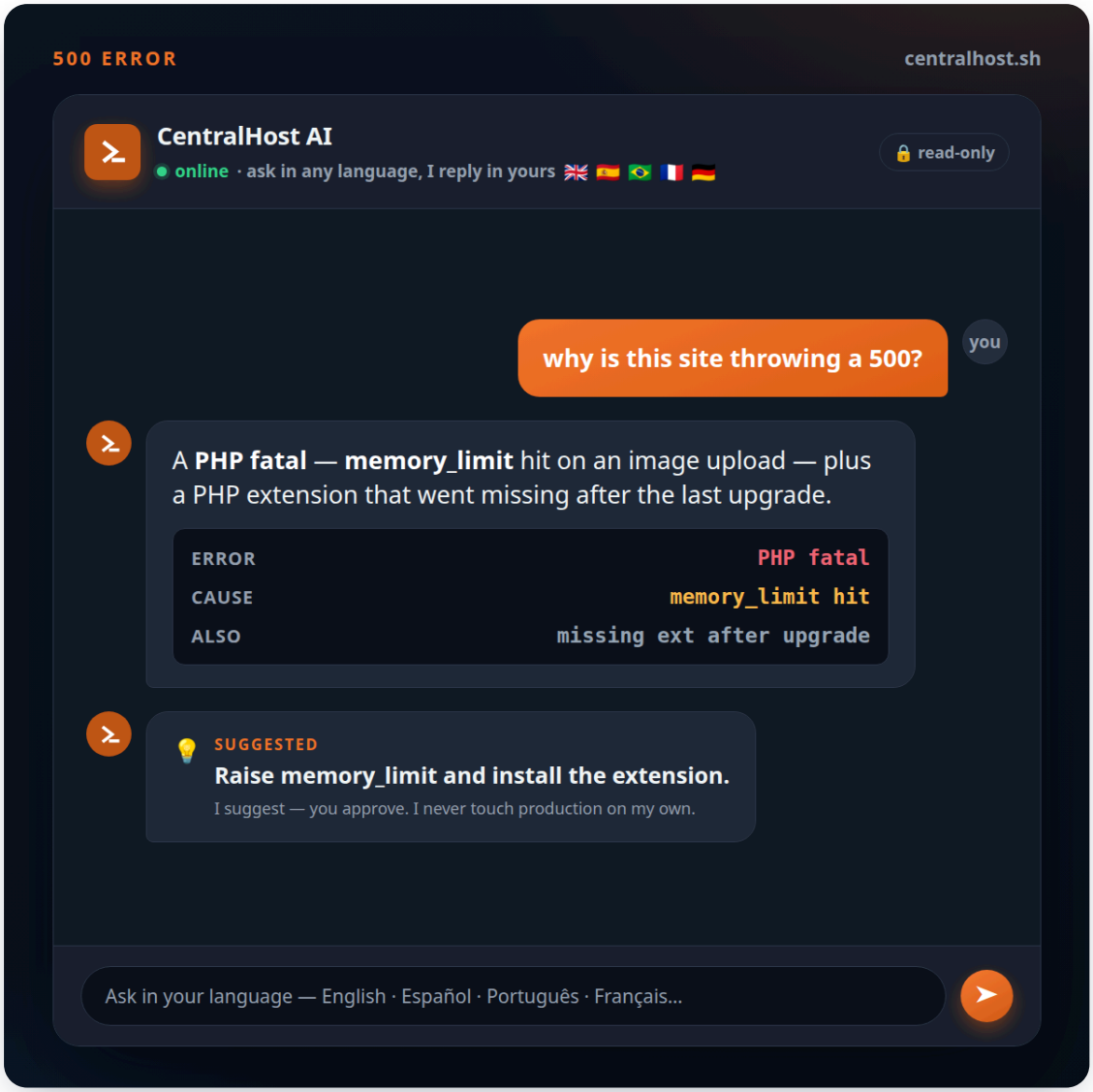
Investigate: ask what's wrong

A compromised mailbox sending spam. Your IP lands on a blacklist and you don't know who burned it. Finding it by hand is 15-20 minutes digging through `exim_mainlog`; the assistant pinpoints the exact mailbox in seconds and proposes the containment:



It traces the outbound spam to the compromised mailbox and proposes suspending the account's outbound mail and forcing a password reset — and you approve.

A site throwing a 500. Instead of guessing between Apache, PHP-FPM and half a dozen logs, you ask in one sentence:



It correlates the PHP fatal with the memory_limit and with an extension that went missing after the last upgrade.

An exposure audit. Exactly what Steps 4–8 of this guide preach, answered instantly:

It finds the database and the unauthenticated Redis open to the internet, and proposes closing them.

Act: ask it to do something

So far the assistant has only read. When you ask it to *act*, it runs commands on the server — but it shows you exactly what it's about to do, waits for your confirmation, and only then executes. One action, one approval; never in batches, never silently. A few examples:

- *"Suspend outbound email for account* `clientxyz` *while I investigate."*
- *"Kill the process running from /tmp and remove the file."*
- *"Block this IP in PXF across all servers in my fleet."*
- *"Restart PHP-FPM on this server."*
- *"Add a DMARC record for this domain."*
- *"Raise PHP-FPM's* `max_children` *from 10 to 20 on this server."*

Appendix II: Web Terminal and External Access

A full SSH terminal, without exposing port 22

CentralHost includes a browser-based SSH terminal that gives you a complete shell session on any server in your fleet — from any device, without a local SSH client, and without port 22 open to the internet. Authentication goes through CentralHost; the server itself never receives an inbound connection from the public internet.

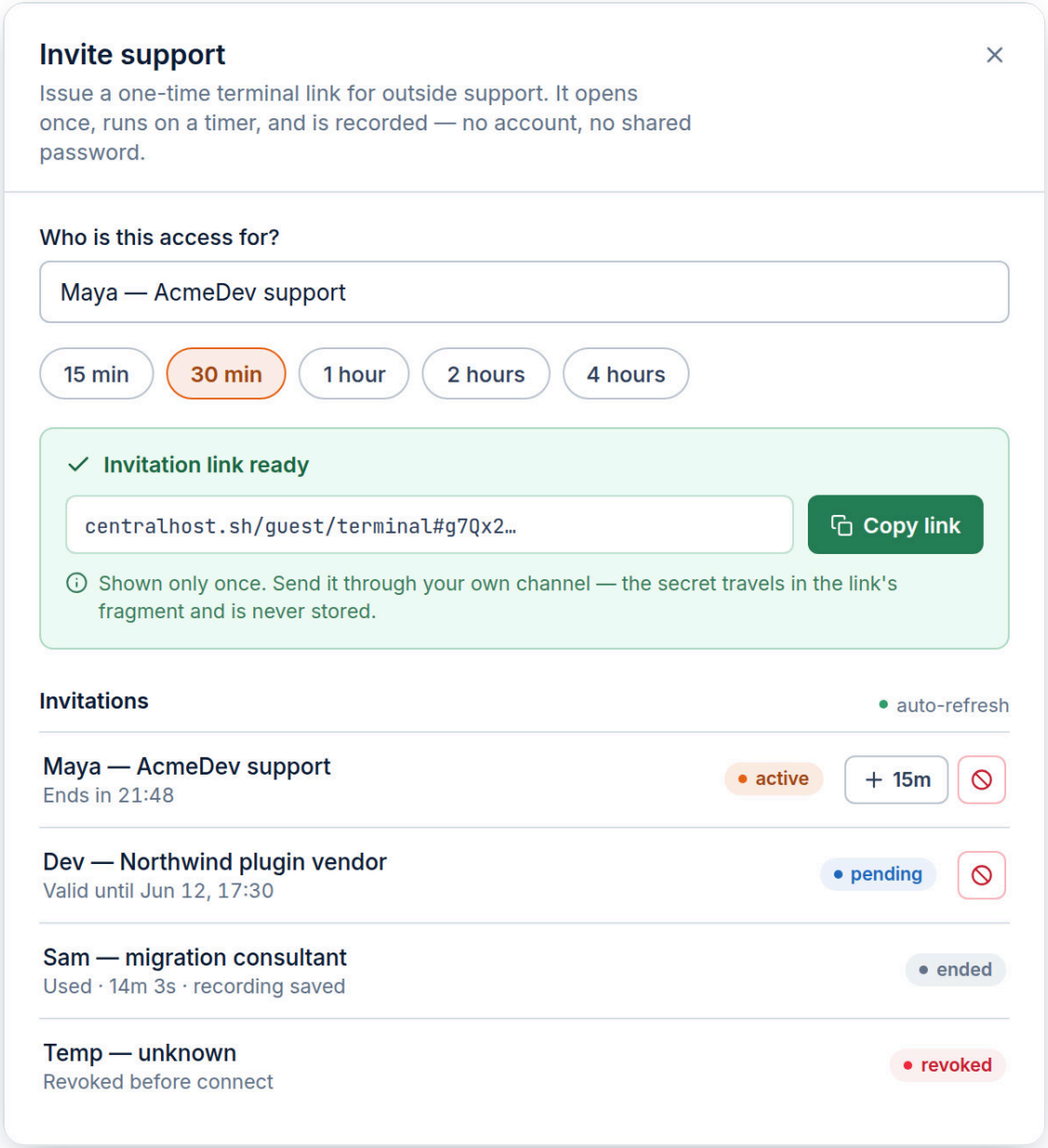
For day-to-day operations this means your team can work from anywhere without VPN requirements or firewall exceptions. For security it means the attack surface is reduced to a single authenticated entry point rather than an open port on every server.

Inviting external support

When an incident requires outside help — a contractor, a specialist, or a vendor's support team — the standard approach is to share root credentials or temporarily open SSH access. Both create risk: credentials get reused, access does not always get revoked, and there is no record of what was done.

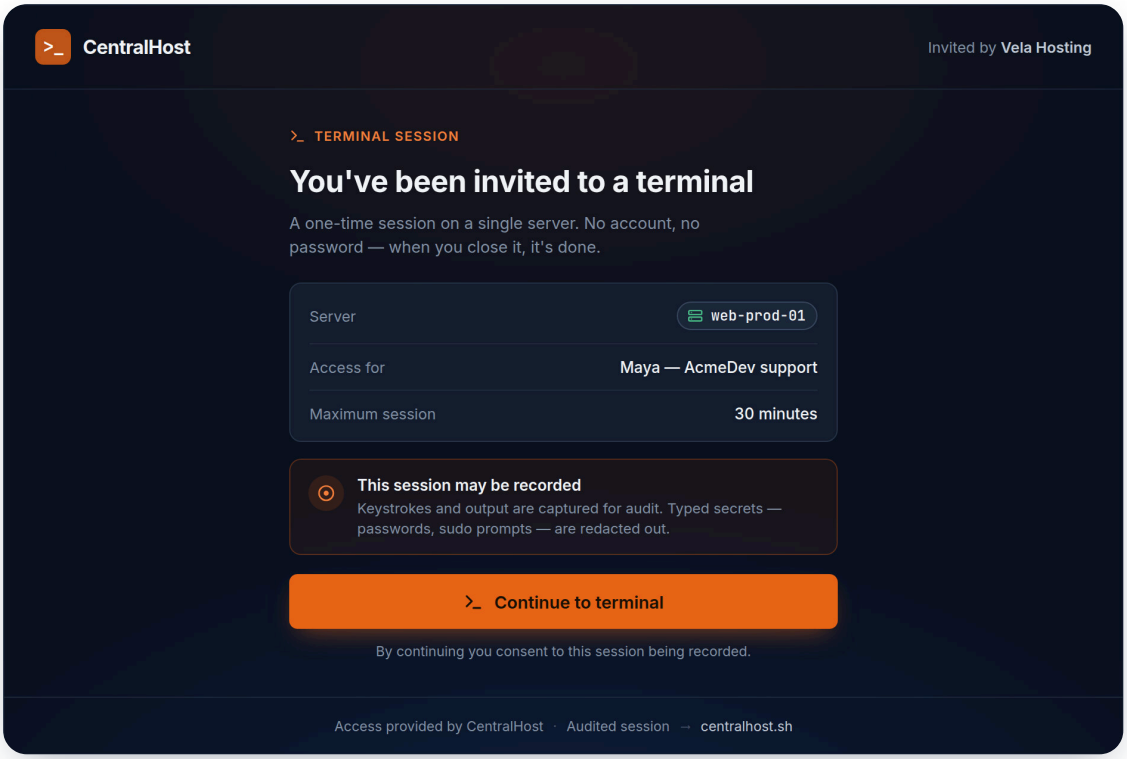
CentralHost handles this differently. You send an invitation link to the external party. They get access to a web terminal session scoped to the specific server you choose — nothing else in your fleet is visible or reachable. No credentials are shared. No firewall rules need to change.

The invitation link is single-use and the terminal session has a maximum time limit. Once the session expires, access is gone automatically — there is nothing to revoke and nothing to forget to clean up.



Pick the duration, copy the single-use link and send it over your own channel — no shared credentials, and the session is recorded.

When the external party opens the link, they don't land in your dashboard or a login form. They see a single consent screen that names who invited them, which server they are about to reach, how long the session lasts, and the fact that it is recorded — followed by one button to continue. No account, no password, nothing else in your fleet within reach.



What the invited operator sees when opening the link: a chrome-less page naming the host, the target server and the time limit, with a recording notice and a single button to enter — no login, no sidebar, no password.

Every session is recorded

Every web terminal session — whether run by your own team or by an invited external party — is recorded in full. The recording captures:

- Every command typed and its exact output
- The complete session timeline with timestamps
- The identity of the operator who ran the session

Recordings are stored and reviewable after the fact. If a contractor makes a change that causes problems three days later, you can replay the session and see exactly what was done, in what order, and what the server responded. If a client disputes that a configuration was changed, the record is there.

For hosting providers with compliance requirements — GDPR audits, ISO 27001 processes, or simply internal accountability — this is the kind of traceability that is normally only available with expensive dedicated PAM (Privileged Access Management) solutions. On CentralHost it is built in.