OpenClaw Security Hardening — Every Fix in February 2026

February 27, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** February 2026 was OpenClaw's biggest security month. The SSRF guard had a bypass via IPv6 multicast addresses (CVE-2026-26324). Symlink-to-hardlink chains could escape the sandbox. Unauthorized Telegram users could trigger disk writes by sending your bot media. Session keys were getting duplicated, breaking routing. All fixed in 2026.2.23 through 2026.2.26. If you're running anything older than 2026.2.24, update now — you're on known-vulnerable code. Run `openclaw doctor --fix` after updating.

📚 **Related:** OpenClaw Security Guide · ClawHub Security Alert · OpenClaw After Steinberger · OpenClaw Setup Guide

If you're running OpenClaw and haven't updated since January, stop reading and update first:

```
npm update -g openclaw
# or
brew upgrade openclaw-cli
```

Then come back and read why.

February 2026 was the most significant security month in OpenClaw's history. The project went from 170,000 to 230,000 GitHub stars while external security researchers filed serious vulnerability reports — SSRF bypasses, sandbox escapes, unauthorized disk writes, session hijacking. The maintainers shipped fixes across five releases (2026.2.22 through 2026.2.26), sometimes with breaking changes that tightened previously permissive defaults.

This article covers every security-relevant fix from that stretch. For each one, I'll explain what the vulnerability was, what an attacker could do with it, and what changed. If you just want the "am I safe" checklist, skip to the verification section.

The big one: SSRF bypass via IPv6 (CVE-2026-26324)

Severity: High

What happened: OpenClaw's SSRF guard checks URLs before the browser tool fetches them, blocking requests to localhost, private networks, and cloud metadata endpoints. The guard was parsing IP addresses to decide if they were internal. But it wasn't canonicalizing IPv4-mapped IPv6 addresses.

An attacker could feed OpenClaw a URL containing `0:0:0:0:0:ffff:7f00:1` — which is just `127.0.0.1` written in IPv6 notation. The SSRF guard saw an IPv6 address and didn't recognize it as loopback. The request went through.

What an attacker could do: Make OpenClaw fetch internal services — your local network, cloud metadata endpoints (like AWS's `169.254.169.254`), or anything else running on localhost. If you had OpenClaw's browser tool enabled and an attacker could influence the URLs it visited (through a crafted message, a malicious skill, or a prompt injection), they could reach services that should have been off-limits.

The fix (2026.2.14, then hardened further in 2026.2.23 and 2026.2.26): All IPv4-mapped IPv6 addresses are now converted to standard IPv4 before being checked against the blocklist. IPv6 multicast literals (`ff00::/8`) are also blocked. In 2026.2.23, the browser SSRF policy was changed to default to "trusted-network" mode — a breaking change that blocks internal network access unless you explicitly opt in.

Credit: @yueyueL (initial report), @zpbrent (IPv6 multicast variant)

What you need to do: If you had browser access configured and need it to reach local services, set your SSRF policy explicitly after updating. Run `openclaw doctor --fix` to check.

Sandbox escapes via symlinks and hardlinks

Three separate but related vulnerabilities, all fixed across 2026.2.23 through 2026.2.26.

Symlink-to-hardlink chain escape

OpenClaw's sandbox restricts file access to a specific directory tree. The path resolver followed symlinks to determine if a target was inside the sandbox boundary. But it didn't account for hardlinks.

An attacker could create a symlink inside the sandbox pointing to a hardlink outside it. The resolver would follow the symlink, find the hardlink (which technically exists on the filesystem), and approve the access. The hardlink then pointed to an arbitrary file outside the sandbox.

In plain English: a malicious attachment or skill could trick OpenClaw into reading files it shouldn't — SSH keys, config files, environment variables, anything readable by the OpenClaw process.

The fix: Hardlinked file aliases are now rejected in workspace boundary checks. Broken symlink targets are resolved through existing ancestors and fail closed if the target is outside the root. This applies to `tools.fs.workspaceOnly`, `tools.exec.applyPatch.workspaceOnly`, and sandbox mount path guards.

Credit: @bmendonca3

Browser temp path escape

The browser tool writes trace files and downloads to a temporary directory. The path handling wasn't checking for symlink-root or symlink-parent escapes. A crafted download filename could redirect writes outside the temp directory.

The fix: Realpath-based write-path checks with secure fallback temp-directory validation.

Config include escape

OpenClaw's `$include` directive loads config from other files. A hardlinked include alias could load files outside the config root, and there were no file-size guardrails.

The fix: Verified-open reads for include files, hardlink aliases rejected, file-size limits enforced.

Telegram DM authorization bypass

What happened: When someone sent your OpenClaw bot a Telegram message with media (photos, files, voice messages), the bot would download and write the media to disk before checking if the sender was authorized.

What an attacker could do: Any Telegram user who knew your bot's username could trigger disk writes on your server. They'd send media, and OpenClaw would dutifully save it to disk regardless of whether that user was in your allowlist. With enough requests, they could fill your disk. Depending on how OpenClaw processed the media afterward, there were further exploitation possibilities.

The fix (2026.2.24 through 2026.2.26): DM authorization is now enforced before media download. The `groupAllowFrom` setting is now required for group access — the old fallback to

the pairing store is gone. Across 2026.2.26, this enforcement was expanded to all account-capable channels: Telegram, Discord, Slack, Signal, iMessage, IRC, BlueBubbles, and WhatsApp.

Credit: @v8hid

What you need to do: Check your Telegram bot's DM policy. After updating to 2026.2.26, `openclaw doctor` will warn you if your allowlist config could cause silent message drops.

Workspace path escape via @-prefixed paths

OpenClaw's workspace boundary guard checked whether file operations stayed within the allowed directory. But it wasn't normalizing `@`-prefixed paths before the boundary check.

An attacker could craft an absolute-path escape that the guard didn't catch, allowing reads and writes outside the workspace boundary.

The fix: Paths are now normalized before workspace boundary validation. This affects workspace-only read, write, edit operations, and sandbox mount path guards.

Arbitrary file read via attachment paths

The `sendAttachment` and `setGroupIcon` actions accept file paths. When `sandboxRoot` wasn't set, these actions didn't enforce a local media root check.

An attacker could provide an absolute path (like `/etc/passwd` or `~/.ssh/id_rsa`), and OpenClaw would read that file and send it as an attachment. On your behalf. To whatever channel was active.

The fix: Local media root checks are now enforced even when `sandboxRoot` is unset.

Credit: @GCXWLP

Anthropic OAuth PKCE verifier exposure

On macOS, the onboarding flow for Anthropic authentication used an OAuth path that exposed the PKCE verifier through the OAuth `state` parameter. The PKCE verifier is supposed to be

secret — if it leaks, an attacker who intercepts the OAuth flow could complete the authentication and gain access to your Anthropic account.

The fix (2026.2.25): The legacy `oauth.json` onboarding path was removed entirely. Anthropic subscription auth is now setup-token-only.

Credit: @zdi-disclosures

Worth noting: ZDI (Zero Day Initiative) is Trend Micro's vulnerability research program. When they report something, it means professional security researchers found it worth disclosing through a formal program. This wasn't a hobbyist finding — it was serious enough for ZDI's process.

Session key duplication

Not a traditional security vulnerability, but it caused real problems: session keys like `agent:main:main` were getting duplicated to `agent:main:agent:main:main`. This broke routing (messages going to the wrong session), caused missing session history, and could leak context between conversations that should have been isolated.

The fix: Session key canonicalization with legacy entry migration. Keys are now normalized before use, and old duplicated keys are cleaned up automatically.

WebSocket authentication hardening

Three fixes tightening how the gateway handles browser connections:

1. **Origin checks for browser WebSocket clients**: Previously, only the Control UI was origin-checked. Now all direct browser WebSocket connections get origin verification.
2. **Password brute-force throttling**: Failed password authentication attempts from browser-origin loopback connections (including localhost) are now throttled.
3. **Auto-pairing blocked for non-Control-UI browsers**: A browser connecting to the gateway could previously auto-pair as a trusted device. Now only the Control UI can do this.

Together, these prevent cross-origin session takeover and brute-force attacks against the gateway.

HTTP security headers

Optional `Strict-Transport-Security` headers for direct HTTPS deployments. If you self-host OpenClaw behind HTTPS (not behind a reverse proxy that handles TLS), this mitigates man-in-the-middle downgrade attacks.

Not a vulnerability fix per se, but a missing security baseline that matters for anyone exposing OpenClaw directly to the internet. (You probably shouldn't be doing that, but if you are, turn this on.)

Node execution approval hardening

When OpenClaw runs commands on node hosts, it asks for approval. The approval system had multiple weaknesses:

- Approval matching wasn't bound to exact argv identity — trailing-space executable path swaps could bypass it
- The `cwd` (working directory) wasn't checked for symlinks — an attacker could rebind it between approval and execution
- Mutable fields in the execution plan could be changed after approval was granted

The fix: Execution plans are now frozen at approval time via `system.run.prepare`. Symlink cwd paths are rejected. Argv is canonicalized before spawn. Whitespace in command text is preserved to prevent path swaps. `GIT_EXTERNAL_DIFF` was added to blocked host environment keys (it could be used to execute arbitrary code during git operations).

Cross-channel reply hijacking

In shared sessions (where multiple channels connect to the same agent), a reply targeting one channel could fall back to the active dispatcher if the target route failed. A webchat or Control UI context could hijack a reply meant for Discord, or vice versa.

The fix (2026.2.24): Cross-channel replies now fail closed instead of falling back. Turn-source channel metadata is wired through the gateway and treated as authoritative for delivery routing.

Heartbeat DM blocking (breaking change)

What changed: Heartbeat messages (periodic status updates from the agent) now block direct/DM delivery by default. Previously, the agent could proactively send you DMs through any connected channel.

Why: An agent proactively sending DMs is a privacy and security surface. If the agent's behavior was manipulated (through prompt injection, a malicious skill, or a compromised session), it could exfiltrate data through DMs to channels the user wasn't monitoring.

What to do if this breaks your workflow: If you rely on the agent messaging you proactively, add this to your config:

```
agents:
  defaults:
    heartbeat:
      directPolicy: "allow"
```

Docker sandbox namespace isolation (breaking change)

What changed: Docker `network: "container:<id>"` namespace-join mode is now blocked by default in sandbox containers. This mode lets a container share the network namespace of another container — useful for performance but it also means the sandbox container can access the other container's network interfaces.

What to do: If you were using namespace-join mode, you need an explicit break-glass flag to re-enable it. The default is now standard isolation.

All security fixes at a glance

Fix	Version	Severity	Credit
SSRF IPv6 bypass (CVE-2026-26324)	2026.2.14+	High	@yueyueL
SSRF IPv6 multicast block	2026.2.26	Medium	@zpbrent
SSRF trusted-network default	2026.2.23	Breaking	—

Fix	Version	Severity	Credit
Sandbox symlink/hardlink escape	2026.2.23-26	High	@bmendonca3
Browser temp path escape	2026.2.26	Medium	—
Config include escape	2026.2.26	Medium	—
Telegram DM auth bypass	2026.2.24-26	High	@v8hid
Workspace @-path escape	2026.2.23	High	—
Attachment arbitrary file read	2026.2.23	High	@GCXWLP
Anthropic OAuth PKCE leak	2026.2.25	High	@zdi-disclosures
Session key duplication	2026.2.24	Medium	—
WebSocket auth hardening	2026.2.25	Medium	—
HTTP security headers	2026.2.23	Low	—
Node exec approval bypass	2026.2.26	High	—
Cross-channel reply hijacking	2026.2.24	Medium	—
Heartbeat DM blocking	2026.2.24	Breaking	—
Docker namespace isolation	2026.2.24	Breaking	—

How to verify you're patched

```
# 1. Check your version
openclaw --version
# You want 2026.2.24 or later. Ideally 2026.2.26+.

# 2. Update
npm update -g openclaw
# or
brew upgrade openclaw-cli

# 3. Run the diagnostic
openclaw doctor --fix
# This catches SSRF policy misconfigs, DM allowlist issues,
# and gateway memory embedding readiness.

# 4. Verify SSRF policy
```

```
# If you use the browser tool with local services:
openclaw doctor | grep -i ssrf
```

After updating, check for these breaking changes:

1. **SSRF policy**: Browser fetch defaults to trusted-network mode. Local service access requires explicit opt-in.
2. **Heartbeat DMs**: Blocked by default. Add `directPolicy: "allow"` if you need proactive agent messages.
3. **Docker namespace-join**: Blocked by default. Requires break-glass flag if needed.

If you can't update immediately

Some setups can't update on short notice. At minimum: disable the browser tool if you don't use it (this eliminates the SSRF surface entirely), and don't expose OpenClaw's port to the internet. If your Telegram bot is public-facing, unauthorized users can trigger disk writes through media on unpatched versions — consider taking it offline until you update.

Beyond that, audit your connected channels. Any channel OpenClaw can read from is a potential prompt injection input. Any channel it can write to is a potential exfiltration path. Remove what you aren't actively using, and don't install skills you haven't reviewed. Our ClawHub security alert covers the risks of community skills in detail.

The bigger picture

February 2026 was the month OpenClaw went from "fun AI project" to "software that professional security researchers take seriously." Having CVEs filed, ZDI disclosures, and half a dozen credited researchers finding real bugs is a sign the project has crossed a threshold.

The maintainers responded fast — five releases in a week, with breaking changes when the permissive defaults were the problem. Credit given to every researcher who reported. That's how security in open source is supposed to work.

But 230,000 people are now running software that connects to their messaging apps and file system. The attack surface is enormous and the February fixes raised the floor, not the ceiling. Running an autonomous agent on your personal infrastructure will always carry risk that a normal application doesn't.

If you're comfortable with that tradeoff, update and keep building. If the list above makes you uneasy, our security guide covers the hardening steps that reduce your exposure, and our alternatives roundup covers options with smaller attack surfaces.

Get notified when we publish new guides.

Subscribe — free, no spam

Source: https://insiderllm.com/guides/openclaw-security-february-2026/

Free guides for running AI locally