OpenClaw Security Assessment

Vulnerability Analysis & Risk Report

Repository: github.com/openclaw/openclaw | Date: February 21, 2026

Classification: CONFIDENTIAL

Executive Summary

This report presents the findings of a source-code security assessment of the **OpenClaw** open-source project (github.com/openclaw/openclaw), a multi-channel personal AI assistant that connects WhatsApp, Telegram, Slack, Discord, and other messaging platforms to large-language-model providers. The review covered authentication mechanisms, session management, input/output sanitisation, browser-automation security, messaging-channel integrations, dependency risk, and architectural patterns.

The assessment identified **12 distinct security issues** spanning four severity levels. Three findings are rated **Critical or High** and could allow an unauthenticated attacker to bypass access controls, inject malicious instructions into the LLM agent, or escalate privileges under reachable network conditions. The remaining findings require lower effort to remediate but collectively increase the attack surface.

Severity	Count	Summary
CRITICAL	1	LLM prompt injection via external content
HIGH	2	Rate-limit bypass; authentication mode downgrade
MEDIUM	6	CSRF gaps, origin bypass, silent plugin failures, DNS rebinding, header injection, DM policy errors
LOW	3	Memory exposure, token persistence, missing output encoding

Table 1 — Finding severity distribution

1. Scope & Methodology

The assessment was performed as a white-box static code review of the OpenClaw main branch (commit snapshot as of February 2026). The following components were examined in depth:

- Gateway HTTP / WebSocket server (*src/gateway/*)
- Authentication & rate-limiting (*auth.ts, auth-rate-limit.ts*)
- CSRF protection middleware (*browser/csrf.ts*)
- Origin validation (*gateway/origin-check.ts*)
- Chat message sanitisation (*gateway/chat-sanitize.ts*)
- External-content / prompt-injection handling (*security/external-content.ts*)
- DM allow-list policy (*security/dm-policy-shared.ts*)
- Browser automation bridge (*browser/bridge-server.ts*)
- Server startup & plugin loading (*gateway/server-startup.ts*)
- Dependency manifest (*package.json*)

The review methodology combined manual code inspection with pattern analysis against OWASP Top 10 (2021), OWASP LLM Top 10 (2024), and CWE common weakness enumeration categories. Automated tool output was not used; all findings derive from direct source-code review.

2. Technology Stack

Understanding the stack is essential context for risk assessment:

Layer	Technology
Runtime	Node.js >= 22, TypeScript 5.9
Web framework	Express 5.x (beta channel)
Browser auto.	Playwright-core 1.58 / Chrome DevTools Protocol
LLM providers	Anthropic Claude, OpenAI, AWS Bedrock
Messaging	WhatsApp (baileys 7.0 RC), Telegram (grammy), Slack, Discord, LINE
Storage	SQLite (sqlite-vec 0.1.7-alpha)
Auth transport	Tailscale Serve/Funnel (optional)
Containerise	Docker / Fly.io

Table 2 — Technology stack summary

3. Detailed Findings

F-01 — LLM Prompt Injection via External Content	
Component	src/security/external-content.ts
Description	The external-content module wraps untrusted data (emails, webhooks, API responses) inside XML-style boundary markers before it is passed to the LLM agent. Injection detection relies on pattern matching against a fixed list of phrases such as 'ignore previous instructions'. This defence is bypassable: a sophisticated attacker can craft payloads using Unicode homoglyphs, multi-step multi-turn attacks, encoding variations, or simply novel phrasing not on the block-list. The fallback boundary markers are enforced only at the text level and depend on the model obeying the instruction to treat their contents as untrusted — an unreliable structural guarantee.
Impact	An attacker who can send a message to any configured channel (WhatsApp, email webhook, etc.) may redirect the LLM agent to exfiltrate conversation history, execute OS commands via enabled tools, or perform unintended actions on behalf of the owner.
Recommendation	Move to a structured, schema-validated message format that separates system context from external data at the API level rather than in plaintext. Adopt a sandboxed tool execution layer so LLM-issued commands are reviewed before execution. Apply allowlisting of permitted tool calls for messages originating from external channels. Log and alert on suspicious boundary-override attempts.

F-02 — Rate-Limiting Bypass via IP Header Spoofing	
Component	src/gateway/auth.ts
Description	Authentication rate limiting is keyed on the client IP resolved via X-Real-IP or X-Forwarded-For when allowRealIpFallback is enabled. These headers are attacker-controlled on any network where the gateway is not placed strictly behind a reverse-proxy that strips them. Even with the flag disabled, the code path that resolves IP falls back to req.socket.remoteAddress without validating the proxy chain. An attacker can rotate spoofed IP values to exhaust the gateway token brute-force window indefinitely.
Impact	Unlimited brute-force attempts against gateway tokens or passwords, defeating the primary account-lockout control.
Recommendation	Accept X-Real-IP / X-Forwarded-For only from an explicit, configured allow-list of trusted proxy CIDRs. Default the proxy trust flag to off with a startup warning. Complement IP-based rate limiting with per-token or per-session attempt counting as a secondary control.

F-03 — Authentication Mode Downgrade Attack	
Component	src/gateway/auth.ts
Description	The gateway resolves its authentication mode (none | token | password | trusted-proxy) through a priority chain: override config > explicit config > environment variable > default. If an attacker can partially influence the environment (for example via a compromised .env file, a misconfigured Docker secret, or a sibling container in a shared compose network) they can inject OPENCLAW_GATEWAY_AUTH_MODE=none, disabling authentication entirely. The code does not validate the selected mode against a minimum security baseline nor emit a startup warning when 'none' is chosen.
Impact	Full unauthenticated access to the gateway API, agent execution, tool invocation, and conversation data.
Recommendation	Log a prominent WARNING at startup whenever auth mode 'none' is active. Consider refusing to start in 'none' mode unless a compile-time flag explicitly enables it for development. Document the configuration precedence rules clearly.

F-04 — CSRF Protection Relies Solely on Browser-Supplied Headers	
Component	src/browser/csrf.ts
Description	The CSRF middleware (browserMutationGuardMiddleware) validates mutating requests using Sec-Fetch-Site, then Origin, then Referer — in that priority order. No synchroniser token or double-submit cookie is used. Browsers predating Sec-Fetch-Site do not send it; Referer is suppressible via Referrer-Policy or HTTPS-to-HTTP downgrades; Origin is absent on same-origin navigation in some contexts. An attacker able to trigger a request from a controlled page where all three headers are absent or spoofed bypasses CSRF protection entirely.
Impact	Cross-site request forgery against any mutating gateway endpoint, enabling an attacker's page to issue commands (send messages, trigger tools) on behalf of a logged-in user.
Recommendation	Add a CSRF synchroniser token (e.g. csurf or a custom implementation) as a secondary defence layer. Set the session cookie SameSite=Strict. Consider requiring a custom request header (e.g. X-Requested-With) which simple cross-origin requests cannot set without a CORS preflight.

F-05 — Origin Validation Loopback Fallback Bypass	
Component	src/gateway/origin-check.ts
Description	checkBrowserOrigin permits any request where both the origin and the request host resolve to loopback addresses without consulting the configured allowlist. A DNS-rebinding attack can cause a victim's browser to resolve an attacker-controlled domain to 127.0.0.1, making the origin appear as a loopback address while the attacker controls the page content. Additionally, the function does not enforce a protocol requirement, meaning file:// or custom schemes could satisfy the loopback check.
Impact	A user who visits a malicious website while the gateway is running locally may allow the attacker to interact with the gateway API without credentials.
Recommendation	Reject requests unless the origin is explicitly present in the configured allowlist, even for loopback sources. Enforce https:// or http://localhost as the only accepted protocols. Implement a TTL-pinning strategy or use a private-network access preflight (CORS-RFC1918) to defend against DNS rebinding.

F-06 — Tailscale Identity Header Injection	
Component	src/gateway/auth.ts
Description	When Tailscale mode is active, the gateway trusts the tailscale-user-login and tailscale-user-name HTTP headers as proof of identity. This trust is conditional on the connection originating from a loopback address. However, if a network misconfiguration or port-forwarding rule allows a non-loopback connection to present these headers, an attacker can impersonate any Tailscale user by injecting them directly.
Impact	Privilege escalation to any Tailscale user identity, including the gateway owner, without possessing valid Tailscale credentials.
Recommendation	Verify Tailscale identity via the local Tailscale daemon's whois API (ts.LocalClient.WhoIs) rather than trusting headers. Reject Tailscale headers outright if the connection is not demonstrably from a loopback or Tailscale-managed interface.

F-07 — Hook Payload Deserialisation without Schema Validation	
Component	src/gateway/server-http.ts
Description	Inbound webhook requests are deserialised from JSON and processed through the applyHookMappings pipeline without an explicit schema check or content-type enforcement. If a mapping step evaluates any dynamic code derived from the payload, or if the pipeline passes data directly to a shell command or database query, this represents an injection risk. The absence of payload size limits also opens a denial-of-service vector.
Impact	Potential server-side injection or denial of service through malformed or oversized webhook payloads.
Recommendation	Define and enforce a JSON schema for all webhook payloads. Reject requests with wrong Content-Type. Enforce an explicit payload size cap (e.g. 1 MB). Audit all mapping steps to ensure no dynamic code evaluation occurs on untrusted data.

F-08 — Silent Plugin Failure During Startup	
Component	src/gateway/server-startup.ts
Description	Plugin initialisation failures are caught and logged at WARN level, allowing the gateway to continue serving requests even if a security plugin fails to load. A race condition window of approximately 250 ms also exists between gateway readiness and internal hook activation, during which the server accepts requests before hooks that may enforce policies are fully initialised.
Impact	A security-relevant plugin (e.g. one enforcing DM policies or audit logging) may be silently absent, and the gateway may process requests before rate limiting or audit hooks are ready.
Recommendation	Treat plugin failure as fatal for plugins marked critical. Move hook activation before accepting connections. Expose a /healthz endpoint that returns 503 until all required plugins are confirmed live.

F-09 — DM Policy Silent Fail-Open	
Component	src/security/dm-policy-shared.ts
Description	resolveDmAllowState reads the allow-from list from the persistent store inside a .catch(() => []) handler, silently substituting an empty array on any error. Depending on how downstream code interprets an empty allowCount, this could default to a permissive open state rather than a deny-all. Additionally, the function accepts Array, and numeric entries bypass normalisation that assumes string methods.
Impact	If the pairing store becomes corrupt or unavailable, inbound DMs from unapproved senders may be accepted and forwarded to the LLM agent.
Recommendation	Replace the silent catch with explicit error propagation and a fail-closed default (deny all on error). Enforce strict typing so only strings are accepted in the allow-from list. Add an integration test covering store read failures.

F-10 — Bridge Server Credentials Stored Without Encryption	
Component	src/browser/bridge-server.ts
Description	The browser bridge authentication registry stores tokens and passwords as plaintext strings in process memory keyed by port number. No token rotation, TTL, or expiry is implemented. Cleanup failures during shutdown are silently ignored, potentially leaving stale credentials in memory if the process is not cleanly terminated.
Impact	A process memory dump (via Node.js --inspect, heap snapshot, or OS-level tool) exposes live credentials. Stale port entries could allow replay after server restart.
Recommendation	Zero-out credentials after use using a buffer-wiping utility. Implement short-lived tokens with automatic expiry. Treat cleanup failures as errors rather than silently ignoring them.

F-11 — No Token Revocation or Session Expiry	
Component	src/gateway/auth.ts
Description	Bearer tokens validated by the gateway have no revocation mechanism and no TTL. Once issued, a token remains valid until it is manually removed from the configuration file and the server is restarted. There is no audit trail of successful authentications or token use.
Impact	A leaked or stolen token grants permanent access until manual remediation, which may be delayed or missed entirely.
Recommendation	Implement a token rotation command (openclaw rotate-token). Introduce an optional TOKEN_EXPIRY_DAYS setting. Log each successful authentication event with timestamp and client IP.

F-12 — Supply Chain Risk from Alpha and Pre-Release Dependencies	
Component	package.json
Description	The dependency manifest includes several packages in alpha or release-candidate state: sqlite-vec (0.1.7-alpha.2), @whiskeysockets/baileys (7.0.0-rc.9), and @buape/carbon (beta). Pre-release packages are more likely to contain undisclosed vulnerabilities, to introduce breaking API changes, and to receive infrequent security patches. The use of Express 5.x (beta) for the main HTTP server adds additional risk.
Impact	A vulnerability in an alpha dependency may not receive a timely patch; auto-update policies could pull in a malicious or broken RC that disrupts the agent.
Recommendation	Pin pre-release dependencies to exact versions and document a review schedule. Subscribe to security advisories for each package. Consider vendoring sqlite-vec or replacing it with a stable SQLite extension. Run pnpm audit in CI and fail the build on high-severity findings.

4. Attack Surface Summary

The diagram below maps the principal entry points and how they chain to backend capabilities. Numbers in parentheses reference finding IDs.

Entry Point	Trust Boundary Crossed	Relevant Findings
Messaging channels (WhatsApp, Telegram, Slack, etc.)	External internet → LLM agent	F-01, F-09
Gateway HTTP API	Internet/LAN → Node.js process	F-02, F-03, F-04, F-07
Browser WebSocket / bridge	Local browser → gateway	F-04, F-05, F-10
Tailscale overlay network	Trusted network → gateway auth	F-06
Webhook ingest endpoints	External internet → hook pipeline	F-07
Plugin / skill loader	Filesystem → Node.js runtime	F-08
npm dependency chain	Package registry → runtime	F-12

Table 3 — Attack surface and finding cross-reference

5. Prioritised Remediation Roadmap

5.1 Immediate Actions (0–2 weeks)

Address these items before the next production deployment:

- **F-01** — Deploy a structured, schema-validated message boundary for external content to reduce LLM prompt injection risk.
- **F-02** — Restrict IP header trust to an explicit trusted-proxy CIDR list; disable allowRealIpFallback by default.
- **F-03** — Emit a startup-blocking WARNING (or refuse to start) when auth mode is set to 'none' without an explicit override flag.

5.2 Short-Term Actions (2–6 weeks)

- **F-04** — Add a CSRF synchroniser token and set SameSite=Strict on session cookies.
- **F-05** — Remove the loopback fallback from origin validation; require explicit allowlist membership.
- **F-06** — Verify Tailscale identity via whois API rather than HTTP headers.
- **F-07** — Define and enforce a JSON schema for webhook payloads; cap payload size.
- **F-08** — Mark critical plugins as startup-blocking; delay accepting connections until all hooks are ready.
- **F-09** — Replace silent catch with fail-closed behaviour in DM policy resolution.

5.3 Medium-Term Actions (6–12 weeks)

- **F-10** — Implement credential zeroing and short-lived bridge tokens with automatic expiry.
- **F-11** — Add token rotation command, optional expiry setting, and authentication audit log.
- **F-12** — Pin pre-release dependencies; add pnpm audit to CI pipeline with build-break on high findings.

6. Conclusion

OpenClaw demonstrates a security-conscious architecture with components such as timing-safe secret comparison, Tailscale overlay integration, and an external-content wrapping system. However, the combination of a large messaging-channel attack surface, an LLM agent with access to execution tools, and several gaps in authentication and CSRF defences creates meaningful risk for users who deploy the gateway in reachable network configurations.

The highest-priority concern is the LLM prompt-injection risk (F-01): because the agent processes messages from untrusted messaging channels and may have access to OS-level tools, a successful injection could have serious consequences beyond typical web application XSS. Remediating the authentication bypass vectors (F-02, F-03) should proceed in parallel to prevent unauthenticated API access.

With focused remediation effort over the next three months, the residual risk profile can be substantially reduced. Re-assessment is recommended after the immediate and short-term roadmap items are implemented.

This report was produced by automated static analysis and manual code review. It does not constitute a penetration test and may not capture all runtime or configuration-dependent vulnerabilities. Always combine code review with dynamic testing and threat modelling.