AI AGENT ORCHESTRATION

SECURITY ASSESSMENT

Paperclip v0.3.1 — Open-Source Multi-Agent Platform

Prepared by

Paul Holder

Security Professional | AI Red Team Researcher

Dyismo Holdings LLC

March 14, 2026

CONFIDENTIAL — For Portfolio Demonstration Purposes

1. Executive Summary

Paperclip is an open-source, self-hosted orchestration platform (MIT license) that coordinates teams of AI agents — Claude Code, OpenClaw, Codex, Cursor, and arbitrary HTTP-reachable agents — into structured company hierarchies with org charts, budgets, goals, and governance. The project launched in early March 2026 and accumulated 14,200+ GitHub stars in its first week.

This assessment examines the security posture of Paperclip v0.3.1 through static source code analysis of the server, authentication middleware, secrets management, agent adapters, and configuration subsystems. Findings are mapped to the OWASP Top 10 for LLM Applications (2025) and MITRE ATLAS framework to provide actionable context for organizations evaluating deployment.

Assessment Overview

Target	Paperclip v0.3.1 (github.com/paperclipai/paperclip)
Methodology	Static source code analysis (server, auth, adapters, config)
Frameworks	OWASP LLM Top 10 (2025), MITRE ATLAS, NIST AI RMF, Google SAIF
Date	March 14, 2026
Critical Findings	**2 Critical, 2 High, 1 Medium**

2. Findings Summary

#	Severity	Finding	OWASP LLM	MITRE ATLAS
F-01	CRITICAL	Default local_trusted mode grants full admin to all requests	LLM06: Excessive Agency	AML.T0040
F-02	CRITICAL	Hardcoded fallback authentication secret in public source	LLM02: Sensitive Info Disclosure	AML.T0024
F-03	HIGH	Agent adapter supports --dangerously-skip-permissions flag	LLM06: Excessive Agency	AML.T0051
F-04	HIGH	Agent child processes inherit sensitive environment variables	LLM02: Sensitive Info Disclosure	AML.T0048
F-05	MEDIUM	Cross-agent prompt injection via shared goal hierarchy	LLM01: Prompt Injection	AML.T0051

3. Detailed Findings

F-01: Default Authentication Bypass via local_trusted Mode

CRITICAL	OWASP: LLM06	ATLAS: AML.T0040	CVSS Est: 9.1

Description: The default deployment mode is local_trusted (server/src/config.ts, line 117). In this mode, the authentication middleware (server/src/middleware/auth.ts, lines 22-25) automatically promotes every incoming request to a full board-level actor with isInstanceAdmin: true, regardless of whether any authentication credentials are provided.

Impact: Any process, script, or application on the same machine — or any device on the same network if the server binds to 0.0.0.0 — can execute any API call with full administrative privileges. This includes creating/terminating agents, modifying budgets, accessing audit logs, and reading secrets. If exposed via Tailscale (recommended in Paperclip docs for mobile access), the entire Tailscale network gains admin access.

Evidence: server/src/middleware/auth.ts line 22-25: req.actor = opts.deploymentMode === "local_trusted" ? { type: "board", userId: "local-board", isInstanceAdmin: true, source: "local_implicit" } : { type: "none", source: "none" };

OWASP LLM06 Mapping (Excessive Agency): The system grants maximum privileges by default with no authentication challenge. Agents and the management plane operate with unrestricted authority unless explicitly reconfigured — violating the principle of least privilege.

MITRE ATLAS AML.T0040 (ML Model Inference API Access): An adversary with network access to the host can interact with the full agent orchestration API without credentials, enabling unauthorized model invocation, task creation, and data exfiltration through the agent system.

Recommendation: Default deployment mode should be authenticated, not local_trusted. Require explicit opt-in for trust-all mode via environment variable with console warning.

F-02: Hardcoded Fallback Authentication Secret

CRITICAL	OWASP: LLM02	ATLAS: AML.T0024	CVSS Est: 8.6

Description: The Better Auth instance creation (server/src/auth/better-auth.ts, line 70) uses a hardcoded fallback secret: "paperclip-dev-secret". This value is used for session signing when neither BETTER_AUTH_SECRET nor PAPERCLIP_AGENT_JWT_SECRET environment variables are set. Because the source code is public (MIT license), any attacker who reads the repository can forge valid session cookies.

Impact: On any Paperclip instance deployed in authenticated mode without explicitly setting a secret, an attacker can craft valid session tokens, impersonate any user including administrators, and gain full control of all companies and agents managed by that instance.

Evidence: server/src/auth/better-auth.ts line 70: const secret = process.env.BETTER_AUTH_SECRET ?? process.env.PAPERCLIP_AGENT_JWT_SECRET ?? "paperclip-dev-secret";

OWASP LLM02 Mapping (Sensitive Information Disclosure): The authentication secret is embedded in public source code. Any deployment that does not override this default exposes its session signing material, enabling full authentication bypass.

MITRE ATLAS AML.T0024 (Exfiltration via Cyber Means): Forged admin sessions enable extraction of all agent configurations, API keys stored in the secrets vault, audit logs, and any data processed by agents — constituting complete system compromise.

Recommendation: Remove the hardcoded fallback. Require BETTER_AUTH_SECRET to be set at startup in authenticated mode. Refuse to start if missing. Generate a random secret on first run and persist it.

F-03: Dangerous Permission Skip Flag in Agent Adapter

HIGH	OWASP: LLM06	ATLAS: AML.T0051	CVSS Est: 7.8

Description: The Claude local adapter (packages/adapters/claude-local/src/server/execute.ts, line 398) supports a --dangerously-skip-permissions flag that disables all permission checks in the Claude Code runtime. When enabled, the agent can execute arbitrary file system operations, run shell commands, and modify system state without any human-in-the-loop confirmation.

Impact: A compromised or misconfigured agent can execute unrestricted code on the host system. Combined with F-01 (default admin mode), any network-adjacent attacker could create an agent with this flag enabled and achieve remote code execution.

Evidence: packages/adapters/claude-local/src/server/execute.ts line 398: if (dangerouslySkipPermissions) args.push("--dangerously-skip-permissions");

OWASP LLM06 Mapping (Excessive Agency): The platform provides a mechanism to completely remove execution guardrails from AI agents. While intended for trusted

environments, the combination with weak default authentication creates a privilege escalation path from network access to arbitrary code execution.

MITRE ATLAS AML.T0051 (LLM Prompt Injection): An attacker who can influence agent prompts — through task descriptions, goal text, or inter-agent messages — can leverage the unrestricted execution environment to perform actions beyond the intended scope, including data exfiltration and lateral movement.

Recommendation: Require explicit per-agent opt-in for permission skipping, log all usages to the immutable audit trail with alert triggers, and prevent this flag from being set via the API in authenticated mode without board approval.

F-04: Environment Variable Exposure to Agent Child Processes

HIGH	OWASP: LLM02	ATLAS: AML.T0048	CVSS Est: 7.2

Description: Agent adapters spawn child processes (via runChildProcess) with environment variables containing the Paperclip API URL, agent identity tokens, run IDs, workspace paths, and potentially API keys (ANTHROPIC_API_KEY). These values are passed directly to the agent runtime environment where they are accessible to any code the agent executes.

Impact: A prompt-injected or malicious agent task can read process.env or /proc/self/environ to harvest authentication tokens, API keys, and internal URLs. These credentials can be exfiltrated to external services or used to impersonate other agents within the system.

Evidence: packages/adapters/claude-local/src/server/execute.ts lines 148-226: Extensive environment variable construction including PAPERCLIP_API_KEY, PAPERCLIP_RUN_ID, PAPERCLIP_TASK_ID, workspace paths, and runtime service URLs passed to child process.

OWASP LLM02 Mapping (Sensitive Information Disclosure): Sensitive credentials are exposed to the AI agent runtime without scoping or time-limited access. The agent — and any code it generates or executes — has full access to these secrets.

MITRE ATLAS AML.T0048 (Exfiltration via ML Inference API): An adversary who achieves prompt injection on an agent can instruct it to read its own environment variables and include them in outputs, API calls, or generated files — enabling credential theft through the LLM's normal output channels.

Recommendation: Use short-lived, scope-limited tokens for agent processes. Inject secrets via temporary files with restrictive permissions rather than environment variables. Implement secret rotation after each agent run.

F-05: Cross-Agent Prompt Injection via Shared Goal Hierarchy

MEDIUM	OWASP: LLM01	ATLAS: AML.T0051	CVSS Est: 6.5

Description: Paperclip's architecture traces every task back to a company mission through a goal hierarchy. Agents receive context including task descriptions, project goals, company mission statements, and inter-agent delegation notes as part of their prompt context. These text fields are set by other agents or users and are not sanitized for prompt injection payloads before being included in agent prompts.

Impact: A malicious or compromised agent (e.g., an "SEO Analyst" role) could embed prompt injection payloads in its task outputs, delegation notes, or @-mention responses. When a downstream agent (e.g., a "CTO" role with broader permissions) processes these messages as context, the injection payload executes in the higher-privilege agent's context.

Evidence: The adapter execute functions construct prompts from context.paperclipSessionHandoffMarkdown, rendered templates including agent and company metadata, and task content — all of which flow between agents as unescaped text. No input

sanitization or instruction hierarchy separation was observed in the prompt construction pipeline.

OWASP LLM01 Mapping (Prompt Injection): Indirect prompt injection through the inter-agent communication channel. Task content and delegation context serve as untrusted input that is injected into agent prompts without boundary enforcement.

MITRE ATLAS AML.T0051 (LLM Prompt Injection): The multi-agent architecture creates a transitive trust chain where any agent's output becomes another agent's input. Adversarial content planted in any node of the hierarchy can propagate to agents with elevated permissions or broader access.

Recommendation: Implement prompt boundary markers between system instructions and inter-agent content. Apply content filtering on agent-to-agent message passing. Enforce role-based output validation before cross-agent context injection.

4. Positive Security Controls

The assessment identified several well-implemented security controls that demonstrate intentional security design:

4.1 Secrets Encryption (AES-256-GCM)

The local encrypted secrets provider uses AES-256-GCM with random 12-byte IVs and authentication tags. The master key file is created with 0o600 permissions (owner-only read/write). Key derivation supports hex, base64, and raw 32-byte formats. This implementation follows current cryptographic best practices.

4.2 Timing-Safe JWT Verification

The agent JWT implementation uses Node.js crypto.timingSafeEqual for signature comparison, preventing timing side-channel attacks. Tokens are scoped with agent ID, company ID, adapter type, and run ID with configurable TTL (default 48 hours).

4.3 Log Redaction

The log redaction module automatically strips OS usernames and home directory paths from all log output, reducing accidental PII exposure in audit trails and error logs.

4.4 Immutable Audit Trail

All agent actions, tool calls, decisions, and conversations are logged to an append-only audit system. This provides forensic capability for incident response and compliance.

4.5 Per-Agent Budget Enforcement

Token spending limits are enforced per agent with atomic checkout to prevent double-work and runaway costs. Budget circuit breakers halt agent execution when limits are reached.

5. Framework Cross-Reference

5.1 OWASP Top 10 for LLM Applications (2025)

OWASP ID	Category	Findings Mapped
LLM01	Prompt Injection	F-05: Cross-agent prompt injection via goal hierarchy
LLM02	Sensitive Information Disclosure	F-02: Hardcoded auth secret; F-04: Env var exposure to agents
LLM06	Excessive Agency	F-01: Default admin access; F-03: Permission skip flag

5.2 MITRE ATLAS Technique Mapping

Technique	Name	Findings Mapped
AML.T0040	ML Model Inference API Access	F-01: Unauthenticated API access to full agent orchestration
AML.T0024	Exfiltration via Cyber Means	F-02: Forged sessions enable full data extraction
AML.T0051	LLM Prompt Injection	F-03: Unrestricted execution via injection; F-05: Cross-agent injection
AML.T0048	Exfiltration via ML Inference API	F-04: Credential theft through agent output channels

5.3 NIST AI RMF Alignment

Findings F-01 and F-03 map to NIST AI RMF GOVERN 1.1 (Legal and regulatory requirements are identified) and MAP 3.5 (Scientific integrity and information quality). The default-insecure configuration contradicts NIST's guidance that AI systems should implement "secure by default" configurations and require explicit opt-in for reduced security postures.

5.4 Google SAIF Alignment

Finding F-05 maps to Google SAIF Principle 3 (Automate defenses to keep pace with threats). The lack of inter-agent prompt sanitization violates SAIF's guidance on maintaining trust boundaries between AI components in multi-model architectures.

6. Conclusion and Risk Rating

Overall Risk Rating: HIGH

Paperclip v0.3.1 demonstrates thoughtful security engineering in its secrets management, audit logging, and budget enforcement subsystems. However, the default deployment configuration prioritizes ease of onboarding over security, creating a critical gap between the platform's intended governance model and its actual out-of-box security posture.

The two critical findings (F-01 and F-02) represent a fundamental disconnect: Paperclip markets governance, approval gates, and human oversight as core features, but ships with defaults that bypass all of them. A developer who follows the quickstart guide (npx paperclipai onboard --yes) gets a system where every request is auto-promoted to admin and session secrets are publicly known.

For organizations evaluating Paperclip for production use, the platform requires explicit hardening before deployment: setting authentication mode to authenticated, generating unique secrets, disabling the dangerous permissions flag, and implementing network segmentation to restrict API access.

For the AI security community, Paperclip represents an important case study in how multi-agent orchestration platforms introduce novel attack surfaces — particularly cross-agent prompt injection (F-05) — that traditional application security frameworks do not fully address.

7. Methodology

This assessment was conducted through static analysis of the Paperclip v0.3.1 source code cloned from the public GitHub repository on March 14, 2026. The following components were reviewed:

- server/src/auth/better-auth.ts — Authentication instance configuration
- server/src/agent-auth-jwt.ts — Agent JWT creation and verification
- server/src/middleware/auth.ts — Request authentication and authorization middleware
- server/src/config.ts — Server configuration and deployment mode defaults
- server/src/secrets/local-encrypted-provider.ts — Secrets encryption implementation
- server/src/log-redaction.ts — Log sanitization for PII
- packages/adapters/claude-local/src/server/execute.ts — Claude Code agent adapter execution

No dynamic testing, penetration testing, or live exploitation was performed. All findings are based on source code analysis and documented behavior. CVSS scores are estimated based on the assessed attack vectors and impacts.

8. Disclaimer

This assessment is provided for educational and portfolio demonstration purposes. The author is not affiliated with Paperclip or its maintainers. Findings are based on publicly available source code under the MIT license. No unauthorized access was performed. Responsible disclosure practices should be followed for any vulnerabilities identified in active open-source projects.