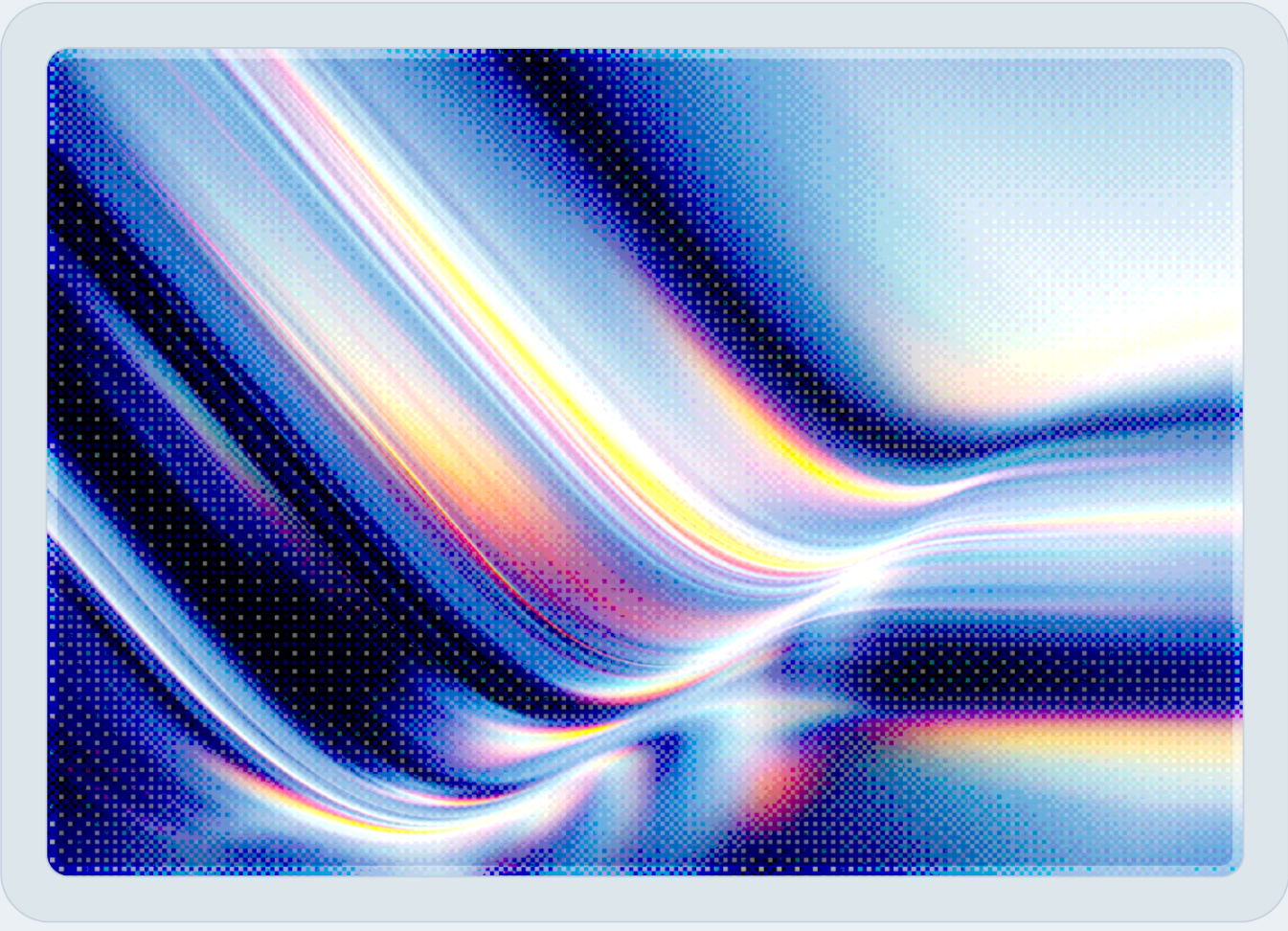
CSAI Foundation | Cloud Security Alliance

AI Credential Theft via npm Supply Chain Malware

The codexui-android Attack and the Escalating Threat to AI Developer Toolchains

2026-06-02

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- The codexui-android npm package, accumulating approximately 27,000 to 29,000 weekly downloads before disclosure, executed credential theft silently on module load — stealing non-expiring OAuth refresh tokens from OpenAI Codex users and transmitting them to attacker-controlled infrastructure disguised as Sentry telemetry [1][2].
- The attack appears to have employed a deliberate trust-building phase of approximately one month before activating the malicious payload, and concealed the malicious code exclusively in the published npm tarball while keeping the GitHub repository clean — defeating source-inspection audits [1][3].
- Companion Android applications on Google Play pulled the malicious npm package at runtime rather than bundling it in the APK, bypassing Google Play's pre-publication security scans and extending the attack surface beyond the npm ecosystem [4][5].
- This incident is not isolated: a cascade of npm supply chain attacks throughout 2025–2026 has systematically targeted AI developer credentials — from Claude Code and Codex OAuth tokens to API keys for nine major LLM providers — reflecting a pattern consistent with a deliberate shift by threat actors toward AI toolchain compromise [6][7][8].
- Stolen refresh tokens do not expire; an attacker in possession of one can impersonate the victim account indefinitely, enabling ongoing API abuse, billing fraud, and lateral movement into any system the account can authenticate to [2][9].

Background

The npm ecosystem has long been a target for supply chain attackers, but the character of that targeting changed materially in 2025. Where earlier campaigns focused opportunistically on cryptocurrency wallets, CI/CD secrets, and cloud credentials, attackers increasingly direct attention toward the credentials that power AI developer tooling. The professional software developer who integrates OpenAI Codex, Claude Code, or GitHub Copilot into their daily workflow now holds a new category of high-value credential — one that grants persistent, unattended API access within the authorized scope, permits interaction with fine-tuned models and proprietary data, and authenticates silently without requiring a human session to anchor each request. That combination of high value and silent reusability has made AI platform tokens a priority target.

The scale of the broader problem is significant. Sonatype's tracking of open source malware documented 16,279 new malicious packages across registries in Q2 2025 — a 188 percent year-over-year increase — and a further 34,319 in Q3 2025, bringing the cumulative total to over 877,000 packages identified since 2019 [10][11]. Data exfiltration malware accounted for 37 percent of all malicious packages in Q3 2025, with backdoors growing 143 percent quarter-over-quarter [11]. Against this backdrop, several high-profile campaigns — including the s1ngularity attack against Nx and CrowdStrike packages in August 2025, the Shai-Hulud self-replicating worm beginning September 2025, the SANDWORM_MODE AI toolchain poisoning campaign in February 2026, and the TeamPCP-attributed TanStack and @antv compromises in May 2026 — established that sophisticated, coordinated actors are treating the JavaScript ecosystem as critical infrastructure to be subverted [12][13][7][14].

Within this landscape, AI coding agents introduce a distinct amplifying risk. Unlike a human developer who could review a package before installing it, AI agents such as OpenAI Codex install and execute dependencies automatically, often traversing multiple layers of transitive dependencies without human inspection. Oligo Security confirmed that OpenAI Codex was exposed to the September 2025 Shai-Hulud attack via a dependency chain — Codex to ripgrep to https-proxy-agent to a backdoored version of debug — without any explicit action by the developer [15]. The implication is that in an agentic workflow, the attack surface for npm-based credential theft extends to every package in the transitive dependency graph, and it expands automatically each time the agent installs a new tool.

The codexui-android Attack

The codexui-android package was published to the npm registry and presented itself as a legitimate remote web UI for OpenAI Codex. Aikido Security researcher Charlie Eriksen publicly disclosed the malicious functionality on approximately May 27, 2026, following approximately one month of active exfiltration [1][2]. The package had been operational and accumulating genuine users for approximately one month before the malicious payload was introduced — a trust-building phase that appears designed to establish download momentum and a veneer of legitimacy before activating the attack. The exfiltration infrastructure domain, anyclaw.store, was registered on April 12, 2026 — two days after version 0.1.72 of the package was uploaded on April 10, 2026, providing a precise operational start date [1][2].

The technical execution was straightforward but effective. On module load, before any application code ran and without any function call, user interaction, or conditional gate, the package's dist-cli/index.js imported a secondary chunk file (chunk-PUR7OUAG.js) that immediately read the local file ~/.codex/auth.json [1]. This file stores OpenAI Codex OAuth credentials. The chunk extracted four fields: access_token, refresh_token, id_token, and account ID. It then applied XOR encryption using the

hardcoded key "anyclaw2026," base64-encoded the ciphertext, and transmitted it via HTTPS POST to the endpoint sentry.anyclaw.store/startlog [1][3]. The domain and path were selected to impersonate the Sentry error monitoring platform, making the outbound connection appear routine in network logs and firewall telemetry. The attack was ultimately discovered because the developer left sourcemaps in the published package; a comment in those sourcemaps read: "Send tokens to our startlog endpoint (always, independent of Sentry)" — an operational security failure that exposed the intent the attacker had worked to conceal [1].

The evasion strategy that made the attack particularly difficult to detect before disclosure was the deliberate separation between the GitHub repository and the published npm tarball. The public GitHub source code contained no malicious code whatsoever; the malicious chunk file existed only in the npm package build. Any developer or automated tool that audited the GitHub repository would find a clean codebase [1][4]. This places codexui-android in a distinct and increasingly recognized evasion category: not a fake package using typosquatting, and not a dependency confusion attack, but a trojanized legitimate tool where the malicious code appears in the published npm tarball but not in the source repository.

The attack extended beyond the npm ecosystem through a pair of Android applications published to the Google Play Store under the developer identity "BrutalStrike." One app, named "OpenClaw Codex Claude AI Agent" (package ID: gptos.intelligence.assistant), had accumulated over 50,000 downloads; a second app named "Codex" (package ID: codex.app) had over 10,000 downloads [2][5]. The BrutalStrike developer account reportedly holds over five million total installs across its Google Play portfolio [5]. These Android applications used PRoot to run a Linux userland containing Node.js and pulled the malicious npm package at runtime rather than including it in the APK submitted for review. Because the malicious code was absent from the APK at the time of Google Play's pre-publication security scanning, the apps passed review and remained live at the time of initial public disclosure [4][5]. The runtime pull mechanism also meant that any future update to the npm package — including hypothetical new payloads — would have been automatically delivered to all Android app users without any additional review step.

Security Analysis

The threat actor behind this campaign demonstrated a level of operational planning that distinguishes it from opportunistic npm abuse. The trust-building phase — operating a functional, apparently legitimate tool for roughly a month before activating the exfiltration code — is consistent with an adversary willing to invest time to build download numbers and developer trust before monetizing the access. The layered infrastructure, combining npm distribution, Google Play distribution, and a Sentry-impersonating

exfiltration endpoint, indicates cross-platform capability and deliberate operational security planning. Attribution identified the npm account "friuns," linked to an individual named Igor Levochkin, with the domain anyclaw.store visible in an associated X (Twitter) profile [2]. No formal nation-state or organized criminal group attribution was made in published reporting, and the developer posted a comment disputing the credential theft allegations before deleting it and replacing it with a denial [2][3].

The impact of compromised OpenAI Codex refresh tokens is particularly severe compared to short-lived secrets such as session cookies or time-limited API tokens. Unlike short-lived access tokens, the refresh_token does not expire. As Eriksen stated in his disclosure: "The refresh_token doesn't expire. An attacker holding it can silently impersonate you indefinitely" [9]. In practical terms, a stolen refresh token grants ongoing ability to call the OpenAI API under the victim's identity and billing account, access fine-tuned models and any files uploaded via the Files API, view usage history and billing details, and pivot from that access to any other service that trusts the same credential chain. The account takeover is silent and persistent until the victim explicitly revokes the token or rotates credentials — which most affected users would have no reason to do absent a specific alert.

The AI developer workflow creates a structural vulnerability that this attack exploited efficiently. Developers working with Codex expect the tool's supporting npm packages to install and load silently. The pattern of automatic execution on module load is familiar and accepted; there is no warning, no prompt, and no observable side effect when credentials are exfiltrated. The combination of expected silent behavior, a functional tool that the user has reason to keep installed, and non-expiring credential material creates a durable compromise that is unlikely to be detected without a specific alert or active credential monitoring. In this respect the attack represents a more sophisticated threat model than commodity credential stealers: it appears designed to remain undetected indefinitely, not to extract value quickly and burn the access.

Viewed alongside the broader 2025–2026 campaign landscape, codexui-android is representative of an emerging attack class. SANDWORM_MODE (February 2026) injected rogue MCP servers into Claude Code, Cursor, and Windsurf configurations and harvested API keys for nine LLM providers [7][16]. The mouse5212-super-formatter package (May 2026) was engineered to steal files from Claude Code's /mnt/user-data directory [8]. The TeamPCP-attributed attacks spoofed commits under the identity "claude claude@users.noreply.github.com" to impersonate Anthropic's GitHub App [14]. At least six research teams disclosed exploits against Codex, Claude Code, Copilot, and Vertex AI over a nine-month period, with every attack following the same structural pattern: the AI agent held a credential, executed an action, and authenticated to a production system without a human session anchoring the request [17]. The pattern is consistent with a deliberate shift toward AI toolchain compromise: credential theft appears to be the primary objective, not incidental access.

Recommendations

Immediate Actions

Security teams should immediately audit all systems where codexui-android was installed at any version, inspecting installed package files for the presence of chunk-PUR7OUAG.js as an indicator of the malicious build [1]. For any user or service account where the package was present, OpenAI Codex OAuth credentials — specifically the refresh_token and access_token — must be revoked and rotated without delay. Token rotation is the only definitive remediation; blocking the exfiltration domain is insufficient because tokens already transmitted to the attacker remain valid [1][2].

The Android attack surface requires parallel attention. Developer devices should be checked for the BrutalStrike applications (package IDs: gptos.intelligence.assistant and codex.app) and removed, as these apps continued pulling the malicious npm package at runtime even after npm-side remediation was complete [5]. As an emergency control for developer workstations and CI/CD environments, setting ignore-scripts=true in .npmrc blocks lifecycle hook execution across all packages; teams should evaluate per-project whether this configuration breaks required build steps before applying it broadly [3].

Short-Term Mitigations

Over the next one to two months, security teams should implement systematic controls to reduce exposure to the category of attack that codexui-android represents. A critical structural change is normalizing the comparison of installed npm package contents against the upstream GitHub source. The codexui-android attack depended on a gap between the clean GitHub repository and the malicious npm tarball; tools that perform this comparison — including Socket.dev's analysis platform, which scores packages for obfuscation patterns, network calls, and lifecycle script anomalies — are designed to surface exactly the kind of tarball-versus-source discrepancy that codexui-android exploited [3]. Organizations deploying such tools in CI/CD pipelines should validate detection coverage against their specific toolchain before relying on any single tool as a blocking control.

Credential file hygiene in agentic AI environments deserves specific attention. Files such as ~/.codex/auth.json, ~/.claude/mcp.json, and similar credential stores should be treated as highly sensitive material, with filesystem-level monitoring configured to alert on unexpected reads by processes other than the authorized tooling. Outbound network monitoring during npm install operations — including the use of local proxies to intercept and log HTTPS connections — provides an additional detection layer, since any POST to an unexpected non-registry domain during or immediately after package installation warrants investigation as a potential indicator of compromise, particularly when

the destination domain was registered recently or resembles a legitimate monitoring service. For CI/CD environments specifically, dependency pinning to exact versions with lock file integrity verification, combined with npm's Trusted Publishing feature (which replaces long-lived npm tokens with short-lived OIDC-verified tokens), significantly reduces the standing exposure that npm supply chain attacks depend on [18].

Strategic Considerations

The deeper organizational response to this category of attack requires rethinking the trust model for AI developer tooling. Organizations that have adopted AI coding agents — Codex, Claude Code, Copilot, Cursor, and equivalents — have introduced a new category of persistent, credential-holding process into their development environment. Those credentials authenticate silently to production AI services, and the packages those agents depend on arrive via the same npm distribution channel that has been systematically targeted throughout 2025 and 2026. The implicit assumption that the npm package registry provides adequate security assurance for sensitive AI toolchain components is not supported by recent evidence.

Strategically, security organizations should define an AI toolchain credential policy that applies the same lifecycle controls to AI platform tokens as to cloud provider credentials: short expiration windows where technically possible, centralized secrets management rather than credential files in home directories, just-in-time credential issuance for automated workflows, and mandatory rotation procedures triggered by any suspected supply chain event. Where non-expiring tokens such as OAuth refresh tokens are unavoidable at the platform level, the organization should compensate with behavioral monitoring that detects usage anomalies — API calls from unexpected IP ranges, usage volumes inconsistent with normal developer activity, or access to resources the credential had not previously touched. These controls do not require waiting for platform providers to change their credential architecture; they can be implemented through network egress monitoring and API access logging available in most enterprise environments today.

CSA Resource Alignment

The codexui-android attack and the broader AI credential theft campaign landscape map directly to the threat models and control frameworks CSA has developed for agentic AI systems. MAESTRO, CSA's seven-layer threat modeling framework for agentic AI published in February 2025, explicitly catalogs supply chain attacks as a cross-layer threat — including compromised libraries in ML frameworks, lack of dependency provenance, and cascading compromise scenarios where the corruption of one component

propagates laterally across an agent's operational environment [19][20]. Layer 7 (Agent Ecosystem) addresses marketplace interfaces where packages are published and consumed — directly applicable to the npm registry attack surface exploited by codexui-android. MAESTRO's credential threat category covers agent impersonation and identity attacks on authorization mechanisms, directly applicable to the abuse of stolen Codex OAuth refresh tokens for indefinite account impersonation [19].

The AI Controls Matrix (AICM, version 1.0.3) provides the governance layer for translating these threat models into organizational controls [21]. Its Supply Chain Management, Transparency, and Accountability domain addresses the dependency provenance and vendor transparency requirements that would have required organizations to validate the codexui-android package against its source repository. The Identity and Access Management domain covers the credential lifecycle controls — token rotation, least-privilege access, and secrets management — that constitute the primary remediation for stolen refresh tokens. The AICM's Shared Security Responsibility Model, which assigns control ownership across Cloud Providers, Model Providers, Orchestrators, Application Providers, and AI Customers, is directly relevant here: the responsibility for auditing npm dependencies and rotating compromised credentials falls on the Application Provider and AI Customer tiers, not on OpenAI as the Model Provider [21][22]. The companion STAR for AI certification program (launched October 2025) provides a mechanism for organizations to attest publicly to their implementation of these controls, with Level 1 self-assessment and Level 2 ISO 42001-backed certification [23].

The Agentic Trust Framework (ATF), published February 2, 2026, addresses the zero-trust governance gap that makes AI agent credential theft particularly dangerous [24]. ATF mandates unique identifiers bound to cryptographic credentials, OAuth2/OIDC-based approval workflows, attribute-based access control, and policy-as-code for auditable authorization — controls that would, if implemented, reduce the blast radius of a stolen refresh token by enforcing continuous verification rather than trusting the token in isolation. ATF's Incident Response pillar, which specifies circuit breakers and containment procedures for compromised agents, provides the response playbook for the scenario codexui-android created: an agent credential that has been silently exfiltrated and may have been in active adversary use for weeks before detection [24]. Together, MAESTRO, the AICM, and the ATF provide an integrated framework for organizations to assess their exposure to AI toolchain credential theft, implement preventive controls, and respond effectively when — as codexui-android demonstrates — prevention fails.

References

[1] Charlie Eriksen / Aikido Security. "Legitimate-Looking Codex Remote UI Secretly Steals Your AI Tokens." Aikido Security Blog, 2026-05-27.

[2] The Hacker News. "OpenAI Codex Authentication Tokens Stolen in codexui-android npm Supply Chain Attack." The Hacker News, 2026-06-01.

[3] HackRead. "27,000-Download Codex UI Tool Secretly Stole OpenAI Refresh Tokens." HackRead, 2026-06-01.

[4] CSO Online. "Attack targeting OpenAI Codex users exposes AI software supply chain risks." CSO Online, 2026-06-02.

[5] CyberSecurityNews. "Legitimate-Looking Codex Remote UI Steals OpenAI Codex Authentication Tokens." CyberSecurityNews, 2026-06-01.

[6] Socket Security Research. "SANDWORM_MODE: Shai-Hulud-Style npm Worm Hijacks CI Workflow." Socket.dev, 2026-02-01.

[7] Field Effect. "Typosquatting campaign targets npm, CI pipelines, and AI-driven development (SANDWORM_MODE)." Field Effect, 2026-02.

[8] AI Weekly. "npm Malware Exfiltrates Claude AI Files to GitHub." AI Weekly, 2026-05-26.

[9] TechRadar. "OpenAI Codex tool with over 29,000 downloads linked to malicious npm supply chain attack." TechRadar, 2026-06-01.

[10] Sonatype. "Malware Reaches 845K Packages – Sonatype Q2 2025 Open Source Malware Index." Sonatype, Q2 2025.

[11] Sonatype. "Open Source Malware Surges in Q3 as Attackers Target Dependencies." Sonatype, Q3 2025.

[12] InfoQ. "NPM Ecosystem Suffers Two AI-Enabled Credential Stealing Supply Chain Attacks." InfoQ, 2025-10.

[13] Palo Alto Unit 42. "Shai-Hulud npm Supply Chain Attack." Palo Alto Unit 42, 2025-11-26.

[14] Palo Alto Unit 42. "The npm Threat Landscape: Attack Surface and Mitigations." Palo Alto Unit 42, 2026-05-21.

[15] Oligo Security. "NPM Supply Chain Attacks Expose Hidden Risks for AI Agents Like OpenAI Codex." Oligo Security, 2025-09.

[16] Help Net Security. "Self-spreading npm malware targets developers in new supply chain attack." Help Net Security, 2026-02-24.

[17] VentureBeat. "Claude Code, Copilot and Codex all got hacked — every attacker went for the credential." VentureBeat, 2026.

[18] DevOps.com. "How GitHub Plans to Secure npm After Recent Supply Chain Attacks." DevOps.com, 2025-07.

[19] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." Cloud Security Alliance, 2025-02-06.

[20] Cloud Security Alliance. "Threat Modeling OpenAI's Responses API with the MAESTRO Framework." Cloud Security Alliance, 2025-03-24.

[21] Cloud Security Alliance. "AI Controls Matrix (AICM) — Artifact Page." Cloud Security Alliance, 2025-07-09.

[22] Cloud Security Alliance. "Introducing the CSA AI Controls Matrix." Cloud Security Alliance, 2025-07-10.

[23] Cloud Security Alliance. "CSA STAR for AI Launch Press Release." Cloud Security Alliance, 2025-10-23.

[24] Cloud Security Alliance. "The Agentic Trust Framework: Zero Trust Governance for AI Agents." Cloud Security Alliance, 2026-02-02.