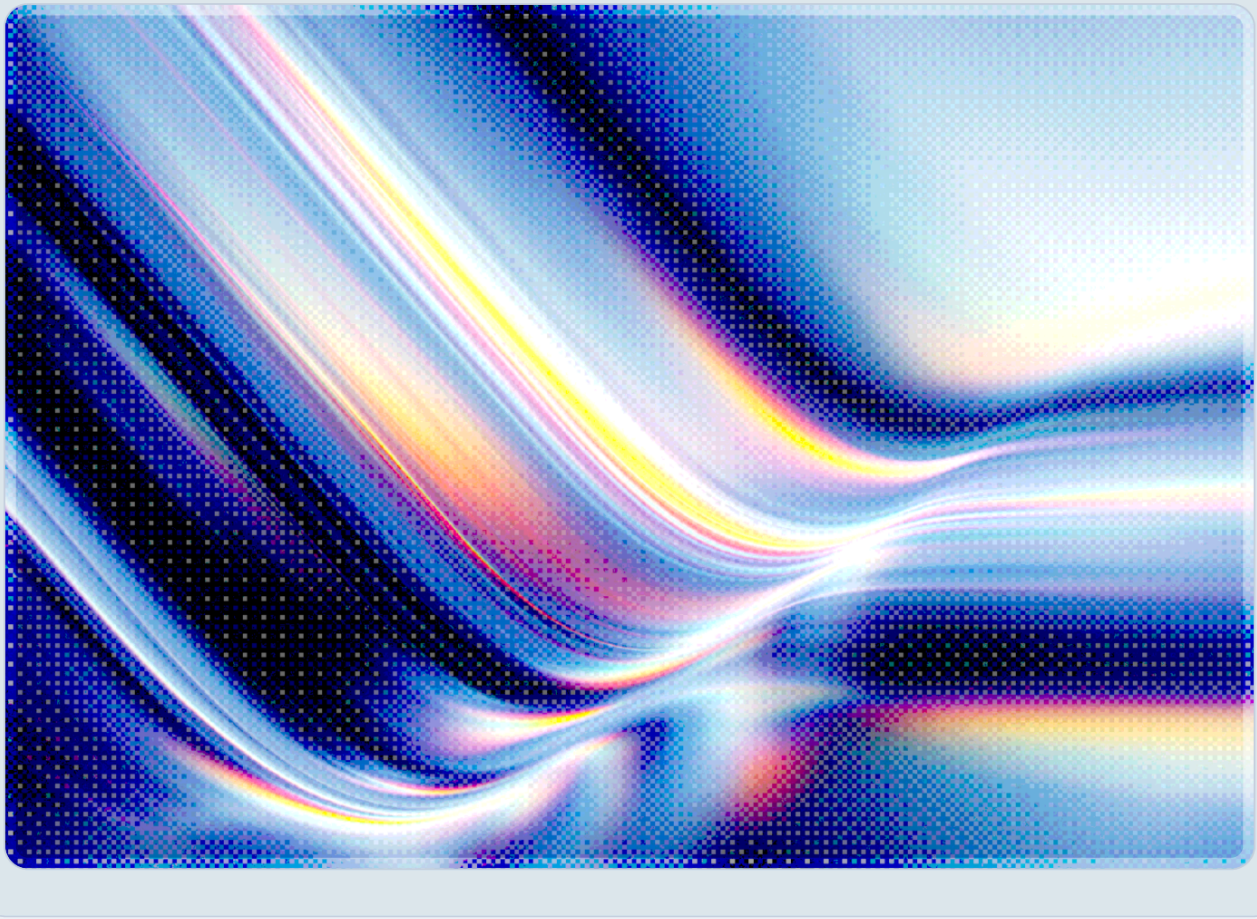
CSAI Foundation | Cloud Security Alliance

AI Toolchain Hijacked: IDE Plugin API Key Theft

Supply Chain Attacks on the AI-Integrated Developer Environment

2026-06-21

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- Three coordinated campaigns active between October 2025 and June 2026 — the JetBrains fake AI assistant campaign, the GlassWorm self-propagating worm, and the Nx Console supply chain compromise — demonstrate that IDE plugin ecosystems have become a primary attack surface for AI credential theft.
- AI API keys (OpenAI, Anthropic, DeepSeek, SiliconFlow) are high-value targets because they enable attackers to consume expensive inference at the victim's cost, access or exfiltrate the content of AI queries, and resell valid credentials to third parties.
- IDE plugins operate with full trust at the user-session level: they can read local files, access environment variables, query keychains, and make outbound network connections without user notification or OS-level sandboxing.
- Supply chain integrity controls designed for application software — including SBOM, SLSA provenance, and dependency pinning — have not been systematically extended to the IDE plugin ecosystem, leaving a significant governance gap.
- Organizations should immediately audit installed IDE plugins against approved allowlists, rotate all AI API keys configured in developer tools, migrate those credentials to vault-backed secrets managers, and apply strict scope and rate limits to API keys used in development environments.

Background

The proliferation of AI-assisted development tools has substantially changed how developers configure and manage credentials in their working environment. Where a developer's IDE once contained little more than syntax highlighting preferences and keybindings, modern development environments now routinely hold API keys for large language model (LLM) providers, cloud infrastructure credentials, package registry tokens, and version control authentication. This concentration of sensitive material within a single, plugin-extensible application has created an attractive new target for adversaries pursuing developer-focused supply chain attacks.

The IDE plugin marketplace model — exemplified by the JetBrains Marketplace, the Visual Studio Code Marketplace, and the OpenVSX Registry — was designed to accelerate developer productivity by enabling community-authored extensions. Each marketplace operates its own plugin review process, but automated static analysis and human review have not reliably detected credential-harvesting logic embedded in otherwise functional plugins, particularly when that logic activates only after user input is received [1]. The trust model underlying these marketplaces assumes plugin authors are legitimate; attackers have learned to exploit this assumption by publishing plugins under fresh vendor accounts, maintaining plausible functionality, and embedding exfiltration routines that activate only when a user enters an API key.

The AI coding assistant category has proved especially attractive to malicious plugin authors. Tools that help developers write code, generate commit messages, and conduct code reviews require API key input by design, and users have a reasonable expectation that those keys will be transmitted to LLM providers. Disguising exfiltration traffic as a normal LLM API call provides natural cover. The rapid commercial growth of AI coding tools — with major providers such as Anthropic, OpenAI, DeepSeek, and SiliconFlow each offering developer API programs — means that a single malicious plugin can harvest credentials from many different providers depending on which service each victim uses.

The attacks documented in this report span a spectrum of sophistication. Some campaigns deployed straightforward credential-harvesting plugins with minimal technical novelty. Others demonstrated advanced tradecraft, including self-propagating worms, blockchain-based command-and-control (C2) infrastructure, and payload delivery through orphan commits hidden inside legitimate open-source repositories. Together they indicate that the developer toolchain is now a well-recognized attack surface that attracts actors across a spectrum of sophistication, from opportunistic credential thieves to technically advanced adversaries whose full attribution has not yet been established.

Security Analysis

The JetBrains Fake AI Assistant Campaign

In June 2026, researchers at Aikido Security disclosed a coordinated campaign that had placed at least 15 malicious plugins on the JetBrains Marketplace over a period beginning in October 2025 [1][13]. The plugins accumulated approximately 70,000 combined installations across seven vendor accounts before detection [2]. Each plugin was presented as an AI coding assistant built on one or more popular LLM providers, including DeepSeek, OpenAI, and SiliconFlow, and offered genuine-looking features such as conversational code assistance, automated commit message generation, and AI-driven code review [3].

The credential theft mechanism was technically straightforward but effective. When a user entered an API key into the plugin's settings panel and clicked Apply, the settings handler stored the key locally — as expected — but simultaneously transmitted it via an HTTP POST request to a hardcoded C2 server at the IP address 39.107.60.51, using plaintext HTTP rather than HTTPS [1]. No user-visible notification accompanied the transmission. The interface gave no indication that credentials were leaving the developer's machine.

Aikido's analysis also identified a monetization dimension that distinguishes this campaign from simple credential theft. Several plugins offered a paid tier, and the remote C2 server appeared capable of supplying API keys to authenticated paid users [1]. The researchers assessed with moderate confidence that stolen API keys harvested from free-tier users were being redistributed as the "paid" service — a mechanism that simultaneously generates revenue for the operator and depletes the quota and billing of the original credential holder. The two most widely installed plugins in the campaign were DeepSeek AI Assist, with approximately 27,700 downloads, and CodeGPT AI Assistant, with approximately 25,570 downloads [3].

GlassWorm: The Self-Propagating Extension Worm

A more technically sophisticated threat emerged in October 2025, when researchers at Koi Security identified GlassWorm, a self-propagating malware campaign distributed through malicious extensions on the Visual Studio Code Marketplace and the OpenVSX Registry [4][14]. By the time of initial disclosure, at least 35,000 developer installations had been compromised; the campaign subsequently expanded through additional malicious Open VSX extensions, npm packages, Python repositories, and poisoned GitHub repository pushes [5].

GlassWorm's most distinctive technical trait is the use of invisible Unicode characters — specifically variation selectors and Private Use Area characters — to embed malicious code within extension source files [4]. These characters produce no visual output in standard editors or code review interfaces, allowing the malicious logic to appear as empty lines during manual inspection or automated diff review. This technique defeats review processes that rely on visual inspection of extension source code, and represents a meaningful increase in sophistication over character-substitution obfuscation approaches.

The malware's C2 infrastructure employed four distinct channels to ensure resilience against takedown: Solana blockchain transaction memo fields encoded C2 server addresses in an immutable public ledger; BitTorrent Distributed Hash Table (DHT) stored configuration data against hardcoded public keys using a decentralized peer-to-peer network; Google Calendar event titles served as dead-drops for Base64-encoded C2 paths; and traditional C2 servers on commercial VPS providers handled payload delivery [4][6]. The use of decentralized and public-infrastructure channels makes C2 blocking substantially harder than for campaigns relying solely on privately operated servers.

Once active on a developer's machine, GlassWorm searched for developer credentials including GitHub tokens, npm authentication tokens, OpenVSX tokens, and cryptocurrency wallets, then used stolen GitHub credentials to force-push malicious code into every repository the victim's account could access — propagating the infection to any developer who subsequently cloned or updated those repositories [5]. Subsequent variants deployed a WebSocket-based remote access trojan designated GlassWormRAT, which captured screenshots, keystrokes, and clipboard content, and installed a malicious Google Chrome extension for persistent browser credential harvesting [5]. A coordinated disruption operation led by CrowdStrike in partnership with Google and the Shadowserver Foundation simultaneously neutralized all four C2 channels on May 26, 2026, though the attribution of the campaign and assessment of its full impact remain ongoing [6].

Nx Console and the GitHub Internal Repository Breach

The incident with the most confirmed downstream impact in this period occurred on May 18, 2026, when attackers published a compromised version of the Nx Console Visual Studio Code extension — version 18.95.0 of the legitimate nrwl.angular-console extension, which carries over two million installations [7]. The malicious version was live for approximately eighteen minutes before the Nx team removed it from the VS Code Marketplace [8][12]. Despite the brief exposure window, the consequences were severe: GitHub publicly confirmed on May 19 that an employee's developer workstation had been compromised through the extension, leading to the exfiltration of approximately 3,800 internal source code repositories [7].

The attack chain was technically sophisticated in several respects. Within seconds of a developer opening any workspace in VS Code, the compromised extension silently fetched and executed a 498-kilobyte obfuscated payload [9]. Rather than hosting the payload on an attacker-controlled server — which would be more easily detected and blocked — the attackers embedded it as an orphan commit inside the official nrwl/nx GitHub repository itself, exploiting the implicit trust that developers place in authenticated, established repositories. The payload targeted a broad credential surface: GitHub tokens, npm OIDC tokens, AWS credentials, HashiCorp Vault tokens, Kubernetes configuration, 1Password vault contents, and files associated with Anthropic Claude Code configurations [9].

The npm OIDC token theft carried a second-order supply chain risk. The stolen tokens included Sigstore credentials, meaning the attacker was positioned to publish downstream npm packages bearing valid, cryptographically signed SLSA provenance attestations, making malicious packages appear as legitimate, verified builds to any downstream consumer relying on supply chain integrity tooling [9]. The U.S. Cybersecurity and Infrastructure Security Agency (CISA) added both associated vulnerabilities —

CVE-2026-45321 (CVSS 9.6) and CVE-2026-48027 (CVSS 9.3) — to the Known Exploited Vulnerabilities (KEV) catalog on May 27, 2026, with a remediation deadline of June 10, 2026 for Federal Civilian Executive Branch agencies [8].

Structural Factors and the AI Credential Economy

These three campaigns share a common structural vulnerability: the IDE occupies a uniquely privileged position in the developer's trust model. Unlike a browser extension, which operates in a sandboxed renderer process, or a downloaded utility, which a security-conscious user might run with reduced privileges, IDE plugins operate at the full level of the user's session. They can read and write the local filesystem, access environment variables and configuration files, initiate outbound network connections, and interact with the operating system's credential store. This access level is broadly necessary for legitimate development tooling but creates an expansive attack surface when abused. The attack vectors documented across all three campaigns map to the Supply Chain Compromise: Software Supply Chain technique (ATT&CK T1195.002) catalogued in the MITRE ATT&CK framework [11], underscoring that IDE plugin ecosystems now represent an established vector within the formal taxonomy of supply chain attacks.

The growing monetization of stolen AI API keys warrants specific attention. AI inference is expensive: enterprise customers may pay substantial monthly fees for access to frontier models. A stolen API key allows an attacker to consume that quota at zero cost while the legitimate holder receives the bill. Aikido's discovery of a credential resale economy within the JetBrains campaign — where stolen keys appeared to be redistributed to paying users of the malicious plugin — illustrates that AI API key theft has evolved beyond opportunistic access to a structured criminal business model. As AI inference costs remain elevated and API key access controls vary widely across providers, the economic incentive for this class of attack is likely to grow.

Marketplace review processes have not kept pace with this threat. JetBrains' review did not detect the credential-harvesting logic in 15 separately submitted plugins over an eight-month period. The VS Code Marketplace's automated controls failed to prevent the publication of the compromised Nx Console version, and the eighteen-minute exposure window was sufficient to compromise at least one high-profile victim. Plugin review at scale — across millions of submissions — is a genuinely hard problem, but the current situation represents a significant gap between the trust developers extend to marketplace-distributed software and the assurance those marketplaces actually provide.

Recommendations

Immediate Actions

Organizations with active developer populations should treat these campaigns as requiring urgent response. Security teams should immediately identify all JetBrains IDE plugins and VS Code extensions installed across developer workstations, compare those inventories against any existing approved software lists, and flag any plugins in the AI coding assistant category published between October 2025 and June 2026 for review against the specific plugins identified in the JetBrains campaign. Developers who installed any of the 15 malicious JetBrains plugins should treat their AI API keys — OpenAI, Anthropic, DeepSeek, SiliconFlow, and any other provider configured in the plugin — as compromised and rotate them immediately. Similarly, any developer who installed VS Code extensions from the OpenVSX Registry between October 2025 and May 2026 should audit their installed extension list and rotate credentials if any GlassWorm-associated extensions are found. For the Nx Console incident specifically, any developer who had VS Code open on May 18, 2026 and who was running the extension should rotate all credentials stored in the affected vault systems: GitHub tokens, npm tokens, AWS credentials, and secrets accessible via 1Password or HashiCorp Vault.

API key usage logs should be reviewed for anomalous activity. Unexpected spikes in LLM API usage — particularly outside business hours or from unusual geographic locations — may indicate that harvested keys are already being used by attackers or resold. Most major LLM providers expose usage dashboards and support webhook or email alerts for quota thresholds; these controls should be enabled across all development API accounts.

Short-Term Mitigations

Developers and platform security teams should migrate AI API keys out of IDE plugin settings fields and into vault-backed secrets managers as a matter of policy. Storing credentials in plugin configuration means the plugin has direct read access to the plaintext credential; storing them in a secrets manager allows the credential to be retrieved programmatically at call time without ever being written to plugin configuration storage. All major LLM provider SDKs — including Anthropic, OpenAI, and DeepSeek — support reading API keys from environment variables following the 12-factor app convention, and this pattern should be standardized across development environments.

API keys issued for development use should be scoped to the minimum necessary permissions. Where providers support it, development keys should be restricted to specific models, capped at a monthly spend or token limit, and bound to expected source IP ranges or network egress points. This limits the

blast radius if a development key is stolen: an attacker who obtains a key with a $50 monthly cap and no production data access is constrained relative to one who obtains a key with unrestricted access. Key rotation should occur on a schedule — quarterly at minimum for development keys — regardless of whether compromise is suspected.

Plugin installation policies should be formalized and enforced. Enterprises using managed device programs can apply VS Code settings to restrict extension installs to an approved list, and JetBrains IDEs support plugin repository restrictions via fleet management. IT and security teams should establish a lightweight vetting process for any AI coding assistant plugin before approval, including review of the plugin's network access patterns, the track record and identity of the publisher, and whether the plugin source code is publicly available for inspection.

Strategic Considerations

The broader structural issue is the absence of supply chain integrity standards for IDE plugin ecosystems equivalent to those being developed for software packages (SBOM, SLSA) and container images. IDE plugins are software supply chain components that arrive on developer workstations and execute with full user-session privileges, yet they are rarely included in the software composition analysis and dependency verification programs that security teams apply to application code and container images. Organizations should extend their software composition analysis programs to cover IDE plugins, cataloging which plugins are in use, which versions, from which publishers, and whether those plugins have documented security review histories.

Vendors operating plugin marketplaces — Microsoft, JetBrains, and the Eclipse Foundation for OpenVSX — should be pressed to adopt stricter publisher identity verification, mandatory code signing with verified certificates, and enhanced behavioral analysis of plugin updates that introduce new network destinations or permission requests. The Nx Console incident demonstrates that even a legitimate, widely trusted plugin can be compromised at the distribution level; marketplace operators should treat compromise of existing publisher accounts as a design assumption rather than an edge case.

At the platform level, developer workstations warrant the same zero-trust network controls applied to production infrastructure. Outbound connections from IDE processes to unexpected IP addresses or domains should generate alerts and, where feasible, be blocked by default. The JetBrains campaign's use of plaintext HTTP to a single hardcoded IP address (39.107.60.51) would have been visible to any organization performing egress traffic monitoring or network-layer content inspection — controls that are frequently omitted from developer network segments where productivity requirements have historically taken precedence over egress monitoring.

CSA Resource Alignment

These incidents map directly to threat scenarios modeled within CSA's MAESTRO framework, specifically at Layer 7 (Ecosystem) of the MAESTRO stack [10]. MAESTRO's ecosystem layer addresses supply chain risks to agentic AI deployments, including the compromise of developer tooling that constructs, configures, or operates AI agents. An IDE plugin that harvests the API keys used by AI coding assistants is an ecosystem-layer attack against the infrastructure on which AI applications are built and maintained. Security teams conducting MAESTRO threat modeling exercises for their AI development workflows should include IDE plugin integrity as an explicit threat scenario.

The AI Controls Matrix (AICM) provides control domains directly relevant to the mitigations described in this note [15]. AICM's Identity and Access Management controls apply to API key lifecycle management — including issuance, scoping, rotation, and revocation for AI provider credentials. Its Supply Chain Security controls address software composition analysis, dependency verification, and vendor assurance requirements that should be extended to cover IDE plugins. Organizations conducting AICM-aligned assessments of their AI development environments should include IDE plugin governance in scope.

CSA's Software Transparency guidance and its work on SBOM adoption in cloud environments offer a methodological foundation for extending composition tracking to IDE plugins. The same SBOM generation and attestation practices being developed for application software dependencies can be applied to document which IDE extensions are installed in the development environment and to flag deviations from an approved baseline.

The Nx Console incident's exposure of Anthropic Claude Code configuration files as a specific theft target underscores a broader principle: AI coding tools generate working credentials and session context that can be as sensitive as long-lived API keys. The security posture of the AI coding environment — including the IDE extensions operating alongside it — must be treated as part of the AI system's security boundary, not merely as a developer workstation hygiene issue.

References

[1] Aikido Security. "Multiple JetBrains IDE plugins caught stealing AI keys." Aikido Security Blog, June 2026.

[2] Cybersecurity News. "Multiple JetBrains IDE Plugins 70,000+ Installs Caught Stealing AI Keys." Cybersecurity News, June 2026.

[3] The Hacker News. "Malicious JetBrains Plugins Steal AI API Keys from Developers." The Hacker News, June 2026.

[4] SecurityWeek. "Supply Chain Attack Targets VS Code Extensions With 'GlassWorm' Malware." SecurityWeek, October 2025.

[5] The Hacker News. "GlassWorm Supply-Chain Attack Abuses 72 Open VSX Extensions to Target Developers." The Hacker News, March 2026.

[6] The Hacker News. "GlassWorm Malware Takedown Disrupts Developer Supply Chain Attack Infrastructure." The Hacker News, May 2026.

[7] The Hacker News. "GitHub Internal Repositories Breached via Malicious Nx Console VS Code Extension." The Hacker News, May 2026.

[8] CISA. "Supply Chain Compromises Impact Nx Console and GitHub Repositories." CISA Alert, May 28, 2026.

[9] StepSecurity. "Nx Console VS Code Extension Compromised." StepSecurity Blog, May 2026.

[10] Cloud Security Alliance. "Applying MAESTRO to Real-World Agentic AI Threat Models: From Framework to CI/CD Pipeline." CSA Blog, February 11, 2026.

[11] MITRE ATT&CK. "Supply Chain Compromise: Compromise Software Supply Chain (T1195.002)." MITRE ATT&CK Enterprise, accessed June 2026.

[12] Infosecurity Magazine. "GitHub Breach Traced to Malicious 'Nx Console' VS Code Extension." Infosecurity Magazine, May 2026.

[13] BleepingComputer. "Malicious JetBrains Marketplace plugins steal AI API keys from developers." BleepingComputer, June 2026.

[14] Truesec. "GlassWorm – Self-Propagating VSCode Extension Worm." Truesec Blog, 2025–2026.

[15] Cloud Security Alliance. "Introductory Guidance to the AI Controls Matrix (AICM)." Cloud Security Alliance, 2025.