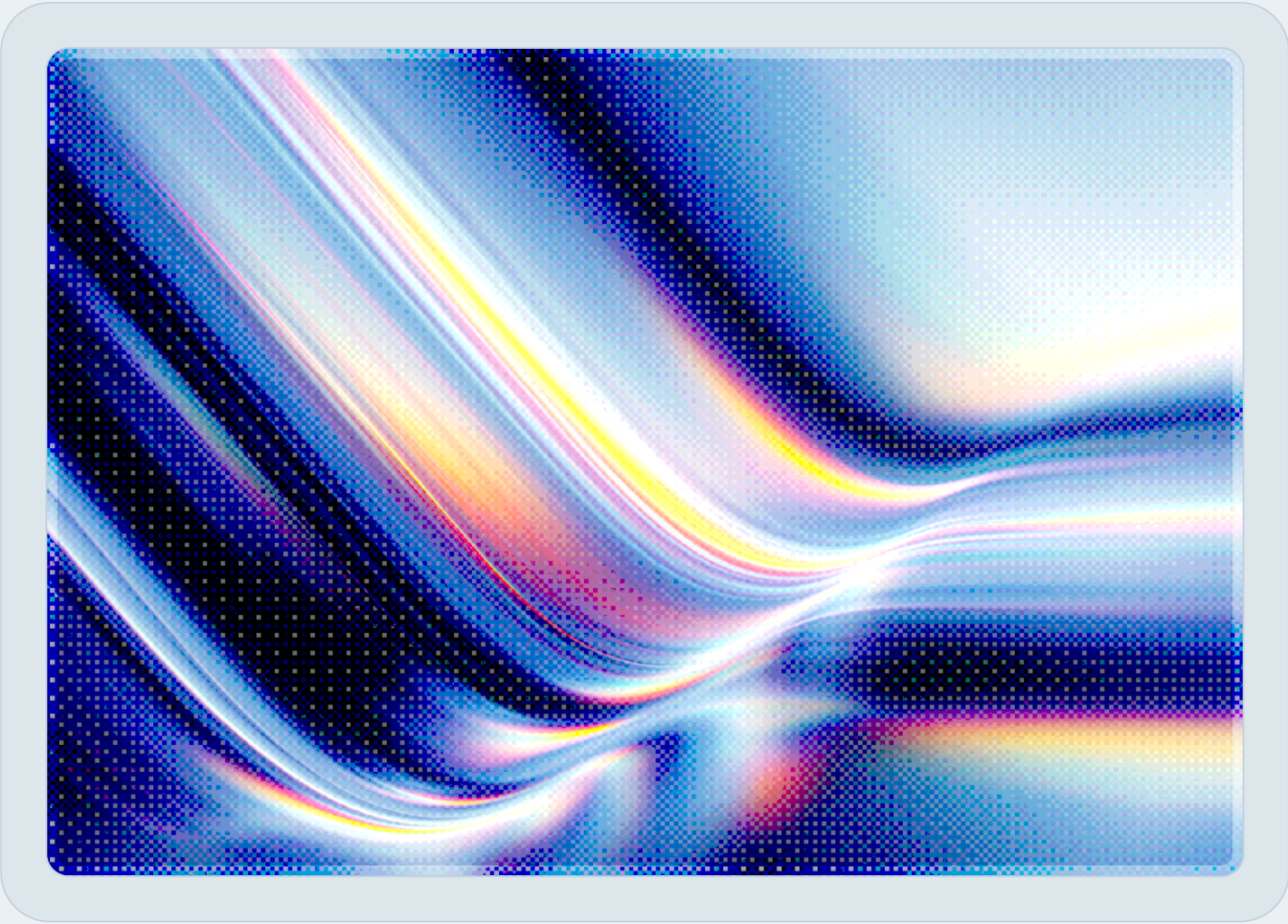
CSAI Foundation | Cloud Security Alliance

VSCode Marketplace Poisoning: How 18 Minutes Breached GitHub

TeamPCP's Trojanized Nx Console Extension Exfiltrates 3,800 Internal Repositories

2026-05-25

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

The following findings summarize the attack mechanics, confirmed breach scope, and organizational implications of the May 18, 2026 supply chain attack attributed to TeamPCP/UNC6780.

- Between 12:30 and 12:48 UTC on May 18, 2026, a trojanized version of the Nx Console VS Code extension (nrwl.angular-console v18.95.0) was live on the Visual Studio Marketplace — an 18-minute window sufficient for TeamPCP to compromise a GitHub employee device and ultimately exfiltrate approximately 3,800 of GitHub's internal source code repositories. [1][2]
- The malicious extension did not require active user installation; the VS Code auto-update mechanism delivered the backdoored release to existing users as a routine update, enabling widespread distribution within a minimal exposure window. [3]
- The payload specifically targeted AI developer credentials alongside traditional secrets, harvesting 1Password vault contents, Anthropic Claude Code configurations, npm tokens, GitHub personal access tokens, and AWS credentials from affected developer environments. [4]
- The attack succeeded because VS Code extensions run with editor-level operating system privileges — the same privilege context as the developer — with no per-extension sandbox, and because a verified publisher badge on the marketplace indicates only that Microsoft has validated the publisher's account, not that individual releases have been code-reviewed. [3] [5]
- The malicious payload was hidden inside an orphan commit (558b09d7) pushed to the legitimate nrwl/nx repository using a stolen contributor token, demonstrating that even trusted, high-visibility open-source repositories can be used as distribution infrastructure for malicious code without alerting maintainers or downstream consumers. [4]
- GitHub confirmed the breach was confined to its corporate repositories; no evidence of impact to customer data, enterprise accounts, or user repositories has been identified to date, though the investigation is ongoing. [1][10]

Background

The GitHub breach represents the highest-profile individual incident to emerge from TeamPCP's campaign series to date, involving a globally recognized platform and confirmed exfiltration of approximately 3,800 internal repositories. The series began in March 2026, when TeamPCP first exploited mutable GitHub Actions version tags in Aqua Security's Trivy scanner to harvest credentials from an estimated 10,000-plus CI/CD pipelines. A cascading chain of credential reuse and downstream compromises across Checkmarx KICS, LiteLLM, and the TanStack npm ecosystem eventually provided TeamPCP with a GitHub personal access token belonging to a legitimate contributor to the Nrwl Nx project — the organization behind the Nx Console VS Code extension. [4][6][9] CSA has separately analyzed TeamPCP's full campaign history and its AI infrastructure targeting in an accompanying research note published May 24, 2026 (available at cloudsecurityalliance.org). This document focuses specifically on the VSCode extension supply chain vector, the mechanics that made an 18-minute exposure window sufficient for a high-impact breach, and the organizational controls necessary to reduce exposure to this class of attack.

The Nx Console extension is a developer productivity tool for Angular and React monorepo workflows, published by Nrwl under the verified publisher designation on the Visual Studio Marketplace. At the time of the incident, it held approximately 2.2 million cumulative installs and a verified publisher badge — trust signals that indicate marketplace legitimacy without conveying anything about the security of any individual release. [1][3][11][12] The extension's large install base and active use in professional development environments made it a high-value target: delivering a malicious update to even a small fraction of that population creates access to developer workstations across a wide variety of organizations simultaneously.

Understanding this incident requires briefly contextualizing the VS Code extension security model. The Visual Studio Code Marketplace hosts more than 100,000 published extensions as of 2025, reflecting the platform's deep integration into professional development workflows across the industry. [17] Extensions execute in the same Node.js process as the editor and run with the operating system privileges of the user who launched VS Code — typically a developer's primary account on a workstation that also holds cloud credentials, SSH keys, container registry tokens, source code checkouts, and, increasingly, AI coding assistant configurations. There is no per-extension permission model that restricts what an extension can read from the filesystem or transmit over the network. Auto-update is enabled by default, meaning that every installed extension will silently receive new versions on editor restart without prompting the user. This combination — broad ambient privilege, no runtime isolation, and automated update delivery — creates an attack surface whose risk profile most organizations have not formally assessed. [3][5]

Security Analysis

The Attack Chain: From TanStack to GitHub's Codebase

The breach did not originate with VS Code or Nx Console. It originated on May 11, 2026, when TeamPCP exploited a `pull_request_target` workflow misconfiguration in the TanStack router repository, combining cache poisoning with OIDC token extraction from GitHub Actions runner memory to hijack TanStack's own release pipeline mid-workflow. [6] Rather than stealing npm tokens directly, the attack published 84 malicious versions across 42 @tanstack/* npm packages through TanStack's own compromised publishing process. More critically for the subsequent GitHub breach, the stolen credential material included a GitHub personal access token belonging to an Nx contributor — a developer whose token granted publish rights to the nrwl.angular-console extension in the Visual Studio Marketplace. [4] [6]

Armed with that token, TeamPCP waited seven days before acting. On May 18 at 03:18 UTC, the group pushed an orphan commit — designated 558b09d7 in the nrwl/nx repository — containing a multi-stage credential-stealing payload. An orphan commit is a commit that exists in a git repository but is not reachable through any branch or tag reference; it occupies storage in the repository and can be fetched by anyone with repository read access, but it does not appear in the default commit history that maintainers and contributors review. [4] Nine hours later, at 12:30 UTC, TeamPCP published nrwl.angular-console v18.95.0 to the marketplace. The malicious extension, on workspace activation, created a hidden VS Code task that fetched and executed the orphan commit payload directly from GitHub's CDN for the nrwl/nx repository. The Nx security team detected the anomaly at 12:47 UTC and removed the malicious version at 12:48 UTC — 18 minutes after publication. [1][4]

During those 18 minutes, VS Code's auto-update mechanism distributed v18.95.0 to users who had any previous Nx Console version installed and restarted or opened their editors. A GitHub employee received the update during this window. The exact count of affected installs beyond GitHub remains uncertain; the Nx team assessed the number at over 6,000 users based on marketplace update telemetry during the exposure window. [1]

Payload Capabilities: What Was Harvested

The payload delivered via the orphan commit was a multi-stage credential stealer and persistence mechanism, not a simple keylogger or exfiltration script. Upon execution, it immediately began harvesting secrets from the local developer environment across several categories. [4] It extracted contents from 1Password vaults accessible to the running user, read Anthropic Claude Code session

configuration files (which may contain API keys and project context), collected npm authentication tokens, GitHub personal access tokens stored in the git credential manager, and AWS credentials from both environment variables and `~/.aws/credentials` files. On macOS systems, the payload also established persistence by writing a Python backdoor named `cat.py` to `/Users/<username>/.local/share/kitty/cat.py`, a path chosen to blend with the Kitty terminal emulator's configuration directory and avoid attracting attention during casual filesystem inspection. [4][11]

The targeting of Anthropic Claude Code configurations is a notable indicator of how specifically TeamPCP has characterized the AI development ecosystem as an attack surface. Claude Code configurations can contain API keys, project system prompts, and tool permission settings. In enterprise deployments, these configurations may represent access to AI-powered development workflows with broad read-write access to codebases. The deliberate inclusion of this credential category alongside more conventional targets suggests that TeamPCP has specifically mapped the credentials held by AI-forward development teams, not simply the result of sweeping for generic developer secrets using a broad file-path enumeration.

Why the Breadth: Auto-Update as an Attack Multiplier

The 18-minute window proved sufficient for mass distribution via auto-update — suggesting that TeamPCP either anticipated this delivery velocity or was prepared to act opportunistically given it. VS Code's extension auto-update mechanism polls the marketplace for new versions of installed extensions and applies updates silently on editor restart, without requiring user review or approval of the new version's contents. For an extension with 2.2 million installs and high developer engagement, a new version published on a Monday morning UTC will reach a substantial fraction of its active install base within the first hour. [3] The 18-minute window was sufficient because the delivery mechanism did not require user intent — it required only that developers opened their editors, which many were doing at the start of a workday.

This dynamic fundamentally changes the risk calculus for IDE extension governance. An extension that an organization vetted and approved six months ago may have silently received multiple updates since then, each of which added or modified functionality that was never reviewed. The supply chain attack surface for developer tooling is therefore not a static inventory; it is a continuously updating collection of software components being silently delivered to developer workstations without the per-release scrutiny applied to production dependencies in a CI/CD pipeline.

Scope and Downstream Impacts

GitHub's confirmed breach involved approximately 3,800 internal repositories, with the company stating that its investigation found no evidence of impact to customer repositories, enterprise accounts, or user data. [1][2][10] GitHub responded by isolating the compromised endpoint, rotating credentials in priority order from highest-impact to lowest, and removing the malicious extension version from the marketplace. [1] TeamPCP subsequently listed the exfiltrated data for sale on the Breached cybercrime forum at a floor price of $50,000 USD. [2][13]

GitHub was not the only organization affected during this window. OpenAI confirmed that two employee devices were compromised during the same exposure window, with limited credential material exfiltrated from a subset of internal source code repositories. [7] Mistral AI confirmed that its npm and PyPI SDKs were trojaned as part of the same campaign, with TeamPCP separately advertising Mistral AI code repositories for sale. [7] Grafana Labs disclosed a breach of portions of its internal codebase traced to the same Nx Console supply chain event. [8] Each of these organizations represents a significant node in the AI and cloud development ecosystem; the collective breach of their internal codebases constitutes a systemic exposure event for the AI infrastructure sector.

Recommendations

Immediate Actions

Any organization whose developers may have had Nx Console installed and active on May 18, 2026 between 12:30 and 12:48 UTC should treat those developer workstations as potentially compromised. The appropriate response is to rotate all credentials accessible from the affected environment — including cloud provider keys, container registry tokens, npm and PyPI publish credentials, GitHub personal access tokens, and any AI API keys stored in IDE configuration files — without waiting for forensic confirmation that exfiltration occurred. Given that the payload targeted 1Password vault contents, organizations should also notify affected employees to review vault access logs for anomalous reads during the exposure window. Security teams should specifically check macOS systems for the presence of `/Users/<username>/.local/share/kitty/cat.py`; its presence is a strong forensic indicator of payload execution and warrants immediate endpoint isolation. [4][11]

Beyond the Nx Console incident specifically, security teams should audit the full VS Code extension inventory across developer workstations. The focus should be on extensions that have not been explicitly approved through an internal review process, that received updates in the past 30 days without

corresponding internal review, or that are installed on devices with access to production credentials or source code for sensitive systems. Any extension that cannot be traced to a reviewed and approved version should be removed pending evaluation.

Short-Term Mitigations

Organizations should configure VS Code to disable automatic extension updates and instead manage extension versions through an internal provisioning process — whether a managed device policy, a curated `.vscode/extensions.json` in project repositories, or an enterprise extension gallery that mirrors only reviewed versions from the public marketplace. Disabling auto-update removes the automated component of the primary delivery mechanism that made the 18-minute exposure window effective. [3][5] Extensions should be treated as third-party dependencies subject to the same version-pinning and change-review discipline applied to npm packages in a production application.

Credential storage practices for developer tooling warrant review independent of this incident. API keys, cloud credentials, and AI service tokens stored in IDE configuration files, shell profiles, or plaintext dotfiles represent an exposure surface that credential-stealing payloads can readily harvest. Organizations should adopt credential provider patterns — such as AWS credential process integration, short-lived OIDC-based tokens for CI/CD contexts, and API key management through a secrets manager rather than flat configuration files — that limit the dwell time and reusability of any credential an attacker might successfully capture from a developer workstation. Fine-grained GitHub personal access tokens scoped to specific repositories and expiring on a 30-day rotation cycle limit the blast radius of any single token compromise significantly compared to classic long-lived tokens with broad repository access.

Strategic Considerations

This incident illustrates a security gap that many organizations have not yet formally assessed: the developer workstation and its tooling ecosystem may sit largely outside the control and monitoring frameworks applied to production infrastructure. Developer endpoints that hold access to source code repositories, cloud infrastructure, container registries, and AI service APIs frequently face less systematic software governance than the servers those same developers produce — a posture this breach makes difficult to sustain.

VS Code marketplace trust signals require reinterpretation in light of this event. A verified publisher badge provides meaningful assurance that the publisher's account belongs to the named organization, but it does not guarantee that the organization's supply chain was free from compromise when any particular release was built and published. Organizations should establish controls that treat any

extension update as a potential supply chain event requiring evaluation — not because extensions are inherently untrustworthy, but because the mechanisms by which trust can be subverted are now well-documented and actively exploited. The pattern that TeamPCP demonstrated — compromising a contributor's token through a separate supply chain event and using it to publish a malicious extension version under a legitimate publisher identity — does not require sophisticated social engineering of the primary publisher. It requires only that the publisher's contributor base have touched any other package in the ecosystem that TeamPCP has already compromised.

CSA Resource Alignment

The GitHub Nx Console breach maps directly to multiple areas of CSA guidance for cloud and AI security practitioners. CSA's AI Controls Matrix (AICM) addresses supply chain integrity for AI tooling under its Development and Operations (DEV) and Infrastructure (INF) control domains, both of which require that organizations maintain current inventories of AI-layer software dependencies and apply integrity verification controls — controls that apply with equal force to the VS Code extensions that AI developers use to build those systems. The AICM's requirements for developer endpoint security are particularly relevant here: the credential types most aggressively targeted by the payload (Anthropic Claude Code configurations, AI API keys, LLM gateway tokens) are AI-specific credentials that may not yet be covered by an organization's existing secrets management and rotation policies. [14]

CSA's MAESTRO framework for agentic AI threat modeling classifies the integrity of the development and build environment as a foundational trust requirement; a developer workstation compromised at the IDE layer represents a compromise of the environment in which AI systems are designed, configured, and deployed. Adversaries who understand the structure of AI development tooling can use developer credential theft to influence the AI systems themselves — not through model-layer attacks, but through access to the source code, configuration, and API keys that define how those systems behave. The targeting of Anthropic Claude Code configurations in the Nx Console payload exemplifies this attack path. [15]

CSA's Zero Trust guidance is directly applicable to the credential management gaps this incident exposed. A Zero Trust architecture treats all credentials as potentially compromised and relies on short-lived, scoped, workload-bound access tokens rather than long-lived keys that can be harvested and replayed from any endpoint. Organizations that have implemented Zero Trust principles for cloud infrastructure access but have not applied equivalent controls to developer tooling credentials have a material gap in their posture. [16] Finally, organizations using CSA's STAR self-assessment program to communicate supply chain security posture to customers and partners should review whether their

disclosures accurately reflect their current controls over developer endpoint security, IDE extension governance, and AI-specific credential management — areas where the TeamPCP campaign has revealed systemic gaps across a broad population of technology organizations.

References

[1] Help Net Security. "TeamPCP Breached GitHub's Internal Codebase via Poisoned VS Code Extension." Help Net Security, May 20, 2026.

[2] BleepingComputer. "GitHub Confirms Breach of 3,800 Repos via Malicious VSCode Extension." BleepingComputer, May 2026.

[3] BlueOptima. "VS Code Extension Security Risks: The Supply Chain That Auto-Updates on Your Developers' Laptops." BlueOptima, 2026.

[4] Ox Security. "Nx Console 18.95.0 Incident: How TeamPCP Breached GitHub." Ox Security, May 2026.

[5] Varonis. "GitHub Breach via Malicious VS Code Extension: What You Need to Know." Varonis, May 2026.

[6] TanStack. "Postmortem: TanStack npm Supply-Chain Compromise." TanStack Blog, May 2026.

[7] Notebookcheck. "VS Code Supply Chain Attack Hits GitHub, OpenAI, and Mistral AI." Notebookcheck, May 2026.

[8] Help Net Security. "GitHub, Grafana Labs Breaches Traced Back to TanStack Supply Chain Compromise." Help Net Security, May 21, 2026.

[9] Arctic Wolf. "TeamPCP Supply Chain Attack Campaign Targets Trivy, Checkmarx (KICS), and LiteLLM." Arctic Wolf, March 2026.

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

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

[12] Aikido Security. "GitHub Breached via VS Code Extension: Developer Supply Chain Attack 2026." Aikido Security, May 2026.

[13] Infosecurity Magazine. "GitHub Confirms Breach of Internal Repositories Via Malicious VS Code Extension." Infosecurity Magazine, May 2026.

[14] Cloud Security Alliance. "AI Controls Matrix (AICM)." Cloud Security Alliance, 2025.

[15] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." Cloud Security Alliance, February 2025.

[16] Cloud Security Alliance. "Zero Trust Guiding Principles." Cloud Security Alliance, 2024.

[17] Microsoft. "Celebrating 50 Million Developers: The Journey of Visual Studio and Visual Studio Code." Microsoft Developer Blog, 2025.