GlassWorm Returns: Developer Toolchain Worm Expands to GitHub and npm

Supply Chain Compromise Across npm, GitHub, VSCode, and OpenVSX

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-20

Key Takeaways

- GlassWorm, a self-propagating developer toolchain worm first identified in October 2025, has returned in a fourth and most expansive wave, compromising an estimated 433 components across OpenVSX, the VS Code Marketplace, GitHub, and npm as of mid-March 2026 [1][2].
- The campaign's defining technique — embedding payloads in invisible Unicode Private Use Area characters — renders malicious code invisible to standard visual inspection in editors, terminals, and GitHub diff views, enabling it to evade routine code review without specialized Unicode analysis tooling [3][4].
- Stolen developer credentials — GitHub tokens, npm authentication tokens, and Open VSX credentials — fuel a self-replication cycle in which each newly compromised account becomes a vector for further propagation [5].
- The campaign's command-and-control infrastructure exploits the Solana blockchain and Google Calendar as dead-drop resolvers to deliver C2 URLs, bypassing conventional network-based detection controls [4][6].
- Legitimate, established extensions are being poisoned via developer account takeover, not just newly published malicious packages — rendering extension age and download count unreliable as trust signals [7].
- Immediate rotation of all developer credentials, audit of installed VS Code extensions, and review of GitHub and npm publish tokens are the highest-priority mitigations, based on the account takeover and credential-harvesting scope documented below.

Background

The GlassWorm campaign was first publicly disclosed in October 2025 by researchers Idan Dardikman, Yuval Ronen, and Lotan Sery at Koi Security, who characterized it as the first self-propagating VS Code extension worm observed in the wild [8][22]. The initial wave, identified on October 17, 2025, involved 13 infected extensions on the Open VSX Registry and one on the Microsoft VS Code Marketplace, accumulating approximately 35,800 downloads before Open VSX declared the incident contained on October 21 [8][9].

The October 21 declaration of containment proved incorrect: a second wave emerged just sixteen days later, when three additional extensions with a combined 9,700 downloads were identified operating on attacker infrastructure that had remained active through the purported containment period [10]. A third wave in December 2025, documented by Secure Annex researcher John Tuckner and Nextron Systems, introduced Rust-based implants with improved resistance to reverse engineering, alongside techniques for artificially inflating extension download counts to improve search ranking among legitimate tools [11].

The fourth and current wave, active from late January through at least mid-March 2026, represents a qualitative escalation. Researchers at Aikido Security, Socket, Step Security, and the OpenSourceMalware community collectively documented the campaign's expansion beyond VS Code's extension marketplace ecosystem into GitHub repository hosting and npm package registries — the two largest open-source distribution channels for developer code [1][2][12]. The naming "GlassWorm Returns" reflects the Koi Security and Aikido Security disclosure of this cross-platform expansion, which brought the total confirmed compromised components to 433 as of the time of this writing [2].

CSA previously published an analysis of GlassWorm's transitive dependency escalation technique targeting Open VSX in a prior research note [6]. This document updates that analysis in light of the expanded attack surface and provides consolidated guidance for security teams managing developer toolchain risk.

Security Analysis

Attack Vector Architecture

GlassWorm operates across four distinct delivery vectors that collectively span the major surfaces of a modern JavaScript/TypeScript developer's toolchain. Understanding how these vectors interact is essential for effective defense, as blocking one without addressing the others leaves meaningful exposure in place.

The primary vector remains VS Code and Open VSX extensions. The campaign's earliest and most refined technique exploits VS Code's `extensionPack` and `extensionDependencies` manifest fields to achieve transitive delivery: a benign extension is published and passes initial review, then later updated to list a separately-published malicious extension as a dependency. Because VS Code and compatible editors automatically install declared dependencies, the malicious extension reaches end users without triggering the scrutiny that accompanies new extension submissions [6][14]. A related

technique — Remote Dynamic Dependencies — enables malware authors to fetch payloads from external URLs at runtime, allowing the active payload to be swapped without publishing a new package version. This defeats version pinning as a mitigation strategy.

The account takeover vector, first observed in the January 30, 2026 incident involving developer account `oorzc`, introduces a fundamentally different threat model. Rather than publishing new, unknown extensions, attackers compromise the npm, Open VSX, or GitHub accounts of legitimate developers and push malicious updates to their existing, trusted packages [7]. Four established extensions with a combined 22,000 prior downloads were compromised in this manner. Because extension age and user count are commonly cited as trust signals for extension selection, this technique systematically undermines risk-based vetting heuristics used by individual developers and enterprise procurement teams alike.

The GitHub propagation vector relies on GitHub personal access tokens harvested from compromised developer environments. Attackers use these tokens to force-push malicious commits into victim repositories — including repositories belonging to accounts with no prior connection to the GlassWorm campaign. To reduce the likelihood of human review triggering an alert, commits are disguised using cover text crafted to appear as routine code changes — consistent with LLM-assisted generation, according to researchers [2]. Among the confirmed compromised repositories are `pedronauck/reworm` (a project with 1,460 GitHub stars), `wasmer-examples/hono-wasmer-starter`, and `doczjs/docz-plugin-css` [2]. Named compromised npm packages include `@aifabrix/miso-client` (v4.7.2) and versions 1.3.0 through 1.3.4 of `@iflow-mcp/watercrawl-watercrawl-mcp` [2].

The Invisible Unicode Technique

GlassWorm's defining technical innovation is its use of Unicode Private Use Area (PUA) characters in the ranges U+FE00 through U+FE0F and U+E0100 through U+E01EF as payload carriers. These characters render as zero-width whitespace in all major code editors — including VS Code, Vim, and JetBrains IDEs — as well as in terminals and GitHub's web diff viewer [3][4]. A minimal decoder embedded in the extension or package extracts the hidden bytes from these characters and passes them to JavaScript's `eval()` function, executing the encrypted payload at runtime.

The technique is conceptually related to the Trojan Source vulnerability (CVE-2021-42574), published in 2021, which demonstrated that Unicode bidirectional control characters could be used to cause code to display differently to human readers than it compiles or executes [15]. GlassWorm adapts this insight from a proof-of-concept manipulation of code semantics to a full operational steganography technique for payload delivery. Unlike bidirectional control characters, which some editors now flag, PUA characters

generate no warning in standard developer tooling. Specialized inspection utilities — such as `cat -v`, hexdump tools, or custom Unicode linting rules — can surface these characters, but such tooling is not present in default developer workflows [3][21].

This technique has practical implications beyond evasion of manual code review. Automated static analysis tools that operate on tokenized representations of source code may silently strip or ignore PUA characters, causing the payload to be invisible to security scanners as well. Organizations relying solely on SAST tooling for pre-publish verification of open source dependencies should treat this as an unmitigated gap in their detection coverage.

Command-and-Control Infrastructure

GlassWorm employs two redundant C2 channels designed to resist both detection and takedown. The primary channel uses the Solana blockchain as a dead-drop resolver: malware polls a hardcoded Solana wallet address and reads C2 URLs embedded in the memo fields of recent transactions. Between November 27, 2025, and March 13, 2026, researchers documented at least 50 transactions updating payload URLs through this channel [6]. Blocking the blockchain ledger itself through conventional enterprise controls is not feasible — the Solana network is globally distributed and the transaction record is immutable. However, organizations can block outbound connections to Solana RPC endpoints at the network perimeter, preventing enrolled malware from polling for updated C2 URLs and limiting the channel's operational effectiveness without requiring action against the blockchain itself.

The fallback C2 channel queries a publicly accessible Google Calendar event, with the C2 address encoded in Base64 within the event title [4]. This technique is notable precisely because it routes through infrastructure that most enterprise security tools explicitly allowlist. Organizations that inspect TLS traffic may be positioned to detect anomalous Calendar API calls from non-browser processes, but those relying solely on domain or IP reputation for network filtering have no practical blocking control available through that mechanism, since `calendar.google.com` is broadly trusted. Process-level application controls, behavioral EDR rules, or browser isolation for developer workstations represent additional layers that can limit non-browser access to the Calendar API.

Scope and Confirmed Impact

Across all four waves, security researchers from Koi Security, Aikido Security, Socket, Step Security, and the OpenSourceMalware community have documented 433 compromised components spanning VS Code extensions, Open VSX extensions, GitHub repositories, and npm packages [1][2]. The 72 malicious

Open VSX extensions documented in the January 31 through March 13, 2026 period and approximately 151 compromised GitHub repositories identified in the March 3 through March 9 window represent the largest single-wave expansions to date [1][2].

The campaign's credential-harvesting scope spans GitHub personal access tokens, npm authentication tokens, Open VSX credentials, Git credentials, SSH keys, CI/CD environment secrets, and authentication material for 49 cryptocurrency wallet browser extensions [5]. The breadth of credential targets is consistent with the campaign's self-propagation logic: any authentication token that grants publish access to a package, extension, or code repository is both a theft objective and a propagation mechanism.

Organizations with contributors who use VS Code or compatible editors and have publish rights to internal or public packages, extensions, or repositories should treat the current threat level as elevated. The ForceMemo follow-on campaign, documented by SecurityWeek in early March 2026, demonstrates that GlassWorm-harvested tokens are being used for downstream compromise operations beyond the worm's own propagation, including forced commits into Python repository ecosystems unrelated to VS Code development [16].

Recommendations

Immediate Actions

Security teams should treat the current wave as an active incident requiring immediate credential triage. All developers with VS Code or compatible editor installations should rotate their GitHub personal access tokens, npm authentication tokens, and Open VSX credentials on an emergency basis, regardless of whether individual compromise is confirmed. Credential rotation is cheap relative to the cost of a successful supply chain compromise, and the campaign's account takeover vector means that absence of a known infection is insufficient grounds for deferral.

Concurrently, security and IT teams should conduct an audit of installed VS Code extensions across developer workstations. Extensions should be cross-referenced against the lists of confirmed malicious extensions published by Koi Security, Aikido Security, and Socket [9][10][11]. Any extension that has not been reviewed for its use of `extensionPack` or `extensionDependencies` manifest fields should be treated as potentially transitive and audited accordingly.

GitHub organizations should review audit logs for push events using personal access tokens from the past ninety days, with particular attention to force-push activity and commits containing binary or non-printable characters. For packages under organizational control, the npm registry records publish history accessible via `npm info <package> time` or the registry web interface, which should be reviewed for any anomalous versions published during the campaign's active period. The npm organization audit log provides a consolidated view of publish events across all packages under organizational ownership.

Short-Term Mitigations

Extension governance policies for developer organizations should be updated to prohibit installation of extensions from the Open VSX Registry or VS Code Marketplace without prior security review. Where feasible, organizations should maintain an approved extension allowlist managed through group policy or VS Code's extension restriction settings (consult VS Code enterprise documentation for the applicable configuration key for your specific deployment and distribution). Extensions should be pinned to specific verified versions, and automatic update mechanisms should be disabled or gated on security team approval — recognizing that version pinning does not fully address the Remote Dynamic Dependencies technique, which requires runtime network monitoring to detect.

Network monitoring should be configured to alert on outbound connections to Solana RPC endpoints (`api.mainnet-beta.solana.com` and similar) from developer workstations and CI/CD runners where such connections are not expected. Similarly, anomalous access patterns to the Google Calendar API from non-browser processes warrant investigation. These are not high-confidence indicators on their own, but in combination with other signals — such as recent extension installation or unusual credential use — they can inform incident investigation.

Organizations publishing packages to npm, PyPI, or extension marketplaces should implement two-factor authentication for all publisher accounts without exception. The January 2026 account takeover of developer `oorzc` illustrates that a single compromised account with publish rights provides attackers with a fully trusted delivery channel for subsequent victims. npm and PyPI both support organization-level 2FA enforcement; security teams should verify availability and activate this control for all registries used in their environment.

Strategic Considerations

The GlassWorm campaign illustrates a structural weakness in the developer toolchain trust model that extends beyond this specific campaign. Extension marketplaces and package registries have historically operated on a model in which new submissions are reviewed but subsequent updates to established

packages receive less scrutiny. The account takeover and transitive dependency techniques both exploit this asymmetry. Registry operators — including Microsoft for the VS Code Marketplace and npm, the Eclipse Foundation for Open VSX, and others — are best positioned to address this structurally through mandatory 2FA, automated behavioral analysis of package updates, and enhanced review of dependency graph changes in updates to high-download packages.

Organizations consuming open source packages and extensions should evaluate their software composition analysis tooling for coverage of VS Code extensions and their transitive dependency trees — a category that, as of this writing, most widely deployed SCA tools do not cover, and security teams should confirm vendor capability documentation for specific tooling in use [21]. The SBOM (Software Bill of Materials) model, which CSA has addressed in prior supply chain guidance, provides a framework for this: requiring SBOMs for extension and package dependencies enables organizations to identify when a transitive dependency changes, establishing the visibility needed to assess the risk of that change before it reaches developer workstations.

CI/CD pipeline security merits attention as a related consideration. The GlassWorm campaign's use of stolen GitHub tokens to push commits into public repositories — and the concurrent, unrelated compromise of GitHub Actions workflows via CVE-2025-30066 [23] — demonstrates that the CI/CD pipeline is a high-value attack surface. Organizations should apply principle of least privilege to all CI/CD credentials, scope GitHub tokens to the minimum required repository permissions, and implement branch protection rules that require signed commits and prevent force pushes. Step Security's `harden-runner` and similar tools provide runtime monitoring of GitHub Actions workflows that can detect anomalous outbound network connections from pipeline steps.

CSA Resource Alignment

GlassWorm exemplifies the software supply chain threat scenarios described in CSA's *Software Transparency: Securing the Digital Supply Chain*, which identifies Software Composition Analysis (SCA), Poisoned Pipeline Execution (PPE), and dependency confusion attacks as leading attack patterns against open source-dependent organizations [17]. The recommendation to generate and maintain SBOMs for all first- and third-party software components directly addresses the visibility gap that allows transitive dependency abuse to succeed undetected.

The *Six Pillars of DevSecOps* series, specifically the Automation and Pragmatic Implementation volumes, provides the most operationally detailed CSA guidance for implementing pipeline-stage controls relevant to this campaign [18][19]. The DevSecOps Software Delivery Pipeline (CDDP) framework defines SCA integration points at the Coding, Integration, and Delivery stages — each of which

represents a checkpoint where GlassWorm-style payloads could be detected if tooling coverage extends to extension packages and their dependency graphs. The Bridging Compliance and Development volume additionally calls out open-source supply chain risk and insider threat as explicit threat categories warranting Compliance-as-Code controls [20].

Within the MAESTRO agentic AI threat model, GlassWorm represents a particularly high-risk scenario for organizations deploying AI coding assistants. Extensions that integrate with AI coding tools — a category that observed GlassWorm samples appear to target, based on extension naming patterns consistent with AI assistant impersonation [1][2] — have access to the same developer credential context as any other extension, but may also have access to code being generated by, or fed to, AI models. Compromise of an AI-adjacent extension extends the attack surface to include AI-generated code repositories and any system prompts or organizational context shared with the AI coding tool. Organizations evaluating MAESTRO Layer 1 (model inputs) and Layer 2 (tool integrations) threat models should include VS Code extension permissions as part of that analysis.

CSA's Zero Trust guidance is directly applicable to the credential management and publisher account controls recommended above. The principle that no implicit trust should be granted based on network location or account history applies equally to extension marketplace accounts: an established publisher account with a long history is not, under a Zero Trust model, a sufficient basis for treating its updates as trusted without independent verification.

References

1. Socket, Aikido Security, Step Security, OpenSourceMalware Community. "GlassWorm Supply-Chain Attack Abuses 72 Open VSX Extensions to Target Developers." *The Hacker News*, March 2026. https://thehackernews.com/2026/03/glassworm-supply-chain-attack-abuses-72.html

2. Aikido Security. "Glassworm Is Back: A New Wave of Invisible Unicode Attacks Hits Hundreds of Repositories." *Aikido Security Blog*, March 2026. https://www.aikido.dev/blog/glassworm-returns-unicode-attack-github-npm-vscode

3. Koi Security (Idan Dardikman, Yuval Ronen, Lotan Sery). "GlassWorm Returns: New Wave Strikes as We Expose Attacker Infrastructure." *Koi Security Blog*, November 2025. https://www.koi.ai/blog/glassworm-returns-new-wave-openvsx-malware-expose-attacker-infrastructure

4. SecurityOnline. "GlassWorm Supply Chain Worm Uses Invisible Unicode and Solana Blockchain for Stealth C2." *SecurityOnline*, 2025–2026. https://securityonline.info/glassworm-supply-chain-worm-uses-invisible-unicode-and-solana-blockchain-for-stealth-c2/

5. BleepingComputer. "GlassWorm malware hits 400+ code repos on GitHub, npm, VSCode, OpenVSX." *BleepingComputer*, March 2026. https://www.bleepingcomputer.com/news/security/glassworm-malware-hits-400-plus-code-repos-on-github-npm-vscode-openvsx/

6. CSA Labs. "GlassWorm: Open VSX Transitive Dependency Supply-Chain Escalation." *Cloud Security Alliance Labs*, 2026. https://labs.cloudsecurityalliance.org/research/csa-research-note-glassworm-open-vsx-transitive-dependency-a/

7. The Hacker News. "Open VSX Supply Chain Attack Used Compromised Dev Account to Spread GlassWorm." *The Hacker News*, February 2026. https://thehackernews.com/2026/02/open-vsx-supply-chain-attack-used.html

8. Koi Security (Idan Dardikman, Yuval Ronen, Lotan Sery). "Self-Spreading 'GlassWorm' Infects VS Code Extensions." *The Hacker News*, October 2025. https://thehackernews.com/2025/10/self-spreading-glassworm-infects-vs.html

9. SecurityWeek. "Supply Chain Attack Targets VS Code Extensions With 'GlassWorm' Malware." *SecurityWeek*, October 2025. https://www.securityweek.com/supply-chain-attack-targets-vs-code-extensions-with-glassworm-malware/

10. The Hacker News. "GlassWorm Malware Discovered in Three VS Code Extensions." *The Hacker News*, November 2025. https://thehackernews.com/2025/11/glassworm-malware-discovered-in-three.html

11. The Hacker News. "GlassWorm Returns with 24 Malicious Extensions Impersonating Popular Developer Tools." *The Hacker News*, December 2025. https://thehackernews.com/2025/12/glassworm-returns-with-24-malicious.html

12. Sonatype. "Hijacked npm Packages Deliver Malware via Solana, Linked to Glassworm." *Sonatype Blog*, 2026. https://www.sonatype.com/blog/hijacked-npm-packages-deliver-malware-via-solana-linked-to-glassworm

13. CSA AI Safety Initiative. "GlassWorm: Open VSX Transitive Dependency Supply-Chain Escalation." Cloud Security Alliance, 2025–2026. (Same publication as reference [6]; see CSA Labs entry for URL.)

14. CSO Online. "Open VSX extensions hijacked: GlassWorm malware spreads via dependency abuse." *CSO Online*, 2026. https://www.csoonline.com/article/4145579/open-vsx-extensions-hijacked-glassworm-malware-spreads-via-dependency-abuse.html

15. Boucher, Nicholas et al. "Trojan Source: Invisible Vulnerabilities." University of Cambridge, 2021. CVE-2021-42574.

16. SecurityWeek. "ForceMemo: Python Repositories Compromised in GlassWorm Aftermath." *SecurityWeek*, March 2026. https://www.securityweek.com/forcememo-python-repositories-compromised-in-glassworm-aftermath/

17. Hughes, Chris (Aquia Inc.). "Software Transparency: Securing the Digital Supply Chain." Cloud Security Alliance, 2023.

18. Cloud Security Alliance and SAFECode. "The Six Pillars of DevSecOps: Automation." Cloud Security Alliance, 2020.

19. Cloud Security Alliance and SAFECode. "The Six Pillars of DevSecOps: Pragmatic Implementation." Cloud Security Alliance, December 2022.

20. Cloud Security Alliance and SAFECode. "The Six Pillars of DevSecOps: Bridging Compliance and Development." Cloud Security Alliance, February 2022.

21. Snyk. "Defending Against Glassworm." *Snyk Blog*, 2026. https://snyk.io/articles/defending-against-glassworm/

22. Veracode. "GlassWorm: The First Self-Propagating VS Code Extension Worm." *Veracode Blog*, 2025–2026. https://www.veracode.com/blog/glassworm-vs-code-extension/

23. GitHub Security Advisory / National Vulnerability Database. CVE-2025-30066: GitHub Actions tj-actions/changed-files Compromise. https://github.com/advisories/GHSA-mrrh-fwg8-r2c3