CSAI Foundation

Cloud Security Alliance AI Safety Initiative

GlassWorm Campaign Deploys Zig-Compiled Dropper to Infect All Developer IDEs

How a Trojanized Open VSX Extension Turns Every IDE Into an Entry Point

Unofficial AI-assisted Research

2026-04-12

Key Takeaways

The GlassWorm threat campaign has entered a new phase of technical sophistication. Researchers at Aikido Security identified a trojanized Open VSX extension that ships a Zig-compiled native binary alongside its JavaScript code, exploiting Node.js's native addon interface to break out of the JavaScript sandbox and operate with full operating system privileges [1]. Rather than delivering a traditional payload, the binary's primary purpose is lateral spread: it enumerates every IDE installed on the developer's system that supports VS Code-compatible extensions, then silently downloads and installs a second-stage malicious extension into each discovered IDE using each editor's own command-line tooling [1].

This technique represents a meaningful escalation from prior waves, which relied on JavaScript-layer payloads that became detectable once signatures were established [2][3]. The Zig-compiled dropper bypasses JavaScript-layer inspection entirely, delivers cross-IDE propagation in a single infection event, and chains into a multi-stage framework that uses the Solana blockchain as a command-and-control resolver — a channel that conventional network controls cannot easily block [4]. Developers who installed either the first-stage extension ("specstudio.code-wakatime-activity-tracker") or the second-stage extension it delivers ("floktokbok.autoimport") should treat their systems as fully compromised and rotate all credentials, tokens, and keys without delay.

Three structural issues create the environment in which GlassWorm has thrived: insufficient vetting by alternative IDE extension registries, the inherent trust developers extend to productivity tooling, and the privileged access that modern AI-assisted IDEs require by design. Addressing these issues requires both immediate tactical responses and longer-term changes to how organizations govern developer tooling.

Background

Campaign History: From npm Packages to IDE Supply Chains

Aikido Security began tracking GlassWorm in March 2025 after discovering a cluster of malicious npm packages that hid JavaScript payloads inside invisible Unicode variation-selector characters (ranges U+FE00–U+FE0F and U+E0100–U+E01EF). The encoding caused the malicious code to appear as

empty strings in virtually every editor and terminal, with execution triggered when the JavaScript runtime decoded the characters at runtime [2]. By October 2025, what researchers assess with moderate confidence to be the same operator — based on C2 infrastructure overlap and shared encoding techniques — had pivoted into the Open VSX registry and GitHub repository ecosystem, compromising maintainer accounts to push poisoned extension updates [3].

The campaign's largest single wave arrived in the first weeks of March 2026 and struck across five hosting environments simultaneously. Researchers ultimately documented 72 malicious Open VSX extensions, 88 npm packages, more than 151 poisoned GitHub JavaScript repositories, and over 120 additional Python repositories — more than 400 software components in less than two weeks [3][10]. The extensions impersonated widely used developer utilities including linters, formatters, and productivity trackers. A separate wave targeting Python repositories followed in the campaign's aftermath as threat actors leveraged stolen GitHub tokens to force-push malicious code into projects that had been authenticated against compromised developer accounts [5].

Throughout all waves, the malware's core infrastructure has remained consistent: C2 server addresses are retrieved through Solana blockchain transaction memos, with Google Calendar events serving as a fallback dead-drop resolver [4]. Both channels are public, decentralized, and difficult to block through conventional means. Security researchers from Polish firm AFINE subsequently developed and released an open-source scanning tool, glassworm-hunter, that organizations can use to scan local extension directories, npm and PyPI packages, and git repositories for GlassWorm indicators without making network requests during the scan itself [6].

The Open VSX Vetting Gap

The Open VSX Registry, maintained by the Eclipse Foundation as an open alternative to the Visual Studio Marketplace, serves as the extension source for IDEs that do not integrate Microsoft's proprietary registry — including Cursor, Windsurf, Gitpod, and similar AI-native editors. The Eclipse Foundation announced mandatory pre-publish security checks for Open VSX in February 2026 [12], but a security disclosure in late March 2026 revealed a critical flaw in the implementation: the scanning pipeline returned a single boolean value to indicate both "no scanners are configured" and "all scanners failed to run" [7]. Under load conditions that caused scanner processes to fail, the registry interpreted the failure as a clean result and published extensions without any malware check — meaning GlassWorm had been exploiting the registry as a distribution channel even after the mandatory scanning mandate was nominally in place [7].

Microsoft's Visual Studio Marketplace is documented as employing multiple scan stages, including checks at submission and post-publication periodic bulk re-scans of the full catalog; no equivalent multi-stage process was reliably enforced at Open VSX before the March flaw's disclosure, which is why

GlassWorm found the registry a productive distribution channel across multiple waves. The Eclipse Foundation confirmed remediation of the flaw following the March 2026 disclosure [7].

Security Analysis

The Zig Native Addon Technique

The choice to implement the first-stage dropper in Zig, compiled as a Node.js native addon, suggests deliberate attention to the defensive landscape. Node.js native addons are shared libraries (`.node` files) that load directly into the V8 runtime and execute at the operating system level, entirely outside JavaScript's sandboxed execution model. The extension delivers `win.node` on Windows (a PE32+ DLL) and `mac.node` on macOS (a universal Mach-O binary compiled for both x86_64 and ARM64) [1]. When the extension activates, the native binary runs with the same permissions as the IDE process itself — which on most developer workstations means read/write access to the home directory, access to credential stores and SSH keys, and network access to internal services.

The choice of Zig as the implementation language compounds the evasion advantage. In this campaign, the Zig-compiled binary was built without a runtime dependency on libc — a compilation mode Zig supports natively — which reduces the well-known library patterns that static analysis tools use as behavioral signals. The broader security community has documented growing adoption of Zig among threat actors for precisely this reason: the Zig Strike offensive toolkit, documented by KPMG in late 2024, explicitly advertises the ability to generate shellcode and DLL payloads that evade modern endpoint detection and response (EDR) stacks including Microsoft Defender for Endpoint [8]. GlassWorm's use of Zig for a targeted supply chain operation follows the same anti-detection logic [1] [11].

Cross-IDE Propagation as a Force Multiplier

Once the Zig binary executes, it enumerates the developer's system for every IDE that supports VS Code-compatible extensions. The enumeration scope extends beyond VS Code itself to include Cursor, Windsurf, and any other editor that has adopted the VS Code extension API — a category that has expanded substantially as AI-native IDEs proliferated through 2025 and early 2026 [1]. For each discovered IDE, the binary retrieves a `.VSIX` installer from GitHub, writes it to a temporary directory,

and invokes the target IDE's own command-line installer to perform a silent, trusted installation [1]. The installer is inherently trusted because it is the IDE's own tooling; no system prompt, elevated privilege request, or security warning accompanies the operation.

The second-stage extension is named "floktokbok.autoimport," a deliberate impersonation of "steoates.autoimport," a legitimate extension with more than five million installs on the official Visual Studio Marketplace [1]. The impersonation relies not on identical naming but on typographic similarity and the expectation that developers reviewing their installed extensions list will not scrutinize publishers carefully. This social engineering layer is the final line of defense the campaign relies on going unnoticed after installation.

Payload Capabilities and the Solana C2 Architecture

The second-stage extension implements a multi-capability framework. Upon initialization, it checks whether the system is configured for a Russian locale; if so, execution halts without further action — a geofencing technique commonly associated with threat actors believed to operate from or adjacent to the Commonwealth of Independent States, though the technique can also be adopted to simulate that origin or is widely copied among unrelated malware families [1][4]. On all other systems, the extension retrieves the active C2 server address by querying a Solana blockchain transaction memo — a technique first documented in the GlassWorm March 2026 wave that has since appeared in other campaigns [4]. The use of a public blockchain as a resolver makes the C2 infrastructure highly resilient to conventional takedown: there is no domain to sinkhole, no IP address to block through standard mechanisms, and no registration record to disrupt.

From the resolved C2 address, the extension downloads a remote access trojan (RAT) that installs a browser extension disguised as "Google Docs Offline" across Chrome, Edge, Brave, Opera, Opera GX, Vivaldi, and Firefox [4]. The browser extension logs keystrokes, dumps cookies and session tokens, captures screenshots, and receives operator commands over a WebSocket channel. Aikido researchers documented the extension's capability to bypass Chrome's app-bound encryption (ABE) protections, which Google introduced specifically to prevent credential extraction by local processes [4][9]. On the credential harvesting side, the framework actively targets authentication tokens for GitHub, npm, and Open VSX accounts — the same credentials that, if compromised, enable the campaign to push new poisoned updates into the registries it has already exploited. It also enumerates 49 cryptocurrency wallet browser extensions, including MetaMask, Phantom, and Coinbase Wallet [4].

The campaign's self-perpetuating credential abuse deserves particular attention. By using the developer's own credentials to publish malicious updates, GlassWorm can sustain itself for extended periods without maintaining its own publishing infrastructure, and each newly compromised maintainer account extends the campaign's reach into additional packages and user bases.

Elevated Risk in AI-Assisted Development Environments

AI-native IDEs such as Cursor and Windsurf have design characteristics that amplify the impact of IDE-level compromise. These environments grant their AI components read access to the entire working repository, write access for code generation, and in many configurations the ability to execute shell commands and run tests. An adversary with code-execution capability inside such an IDE is not merely positioned to steal credentials — they can inject malicious code into whatever the developer is actively writing and persist across all future development sessions without any additional infection step. The GlassWorm dropper's propagation to every installed IDE ensures that switching editors does not remediate the infection. Beyond what the GlassWorm campaign has demonstrated, adversaries with this level of access could in principle also influence AI-generated code suggestions to include attacker-controlled packages or endpoints — a capability that has not been observed in this campaign but follows directly from the execution access the dropper achieves.

Recommendations

Immediate Actions

Any developer or organization that has encountered "specstudio.code-wakatime-activity-tracker" or "floktokbok.autoimport" in their extension list — across any IDE — should treat the affected workstation as compromised. The first priority is credential containment: all API tokens, SSH keys, and cloud access keys accessible from the affected machine must be revoked and rotated immediately, with GitHub, npm, and cloud provider tokens receiving highest priority given the campaign's documented credential harvesting capabilities. All repositories and packages published from the affected account should then be audited for unauthorized commits, version bumps, or configuration changes in the past 90 days. All VS Code-compatible extensions should be removed from every IDE on the affected system and reinstalled only from verified sources after re-imaging, and browser extension lists across all installed browsers should be reviewed and cleaned of any extension not explicitly recognized.

Run the glassworm-hunter open-source scanner against the local extension directory, npm package cache, and any Python environments before declaring the system clean [6]. The tool outputs SARIF-formatted results suitable for integration with GitHub Code Scanning, enabling organizations to incorporate GlassWorm detection into pull-request pipelines and CI/CD gates.

Short-Term Mitigations

Organizations should establish formal extension governance policies for developer workstations. This means defining an approved list of IDE extensions, requiring that any new extension be reviewed before installation, and preferring extensions from the official Visual Studio Marketplace over Open VSX where functional equivalents exist — given the documented difference in vetting maturity between the two registries. Where Open VSX extensions are necessary, organizations should verify that the Eclipse Foundation's mandated pre-publish scanning has been applied and that the remediated version of the pipeline (patched in Open VSX 0.32.0) is in use [7].

Endpoint controls should be configured to alert on `.node` file creation in IDE extension directories and on IDE CLI processes spawning child processes that write to temporary directories and invoke additional CLI installers. These behavioral patterns are reliable indicators of the GlassWorm dropper's lateral propagation step. Organizations using browser enterprise management should enforce extension allowlists in Chrome and other browsers, preventing force-installation of unauthorized extensions even when the installer has code-execution access on the system.

Given GlassWorm's use of blockchain and Google Calendar as C2 channels, network-layer blocking of Solana RPC endpoints and calendar API calls is impractical for most organizations and would generate excessive false positives. The better control is endpoint behavioral detection rather than network filtering.

Strategic Considerations

The GlassWorm campaign demonstrates that the software supply chain extends directly into developer workstations and that IDEs are not passive tools but active participants in the security perimeter. Organizations that have invested in securing their CI/CD pipelines and artifact registries but have not applied equivalent scrutiny to developer endpoint tooling have a material gap. Developer workstations should be brought under the same endpoint detection, secrets management, and access control disciplines applied to production infrastructure.

Extension registries must move from reactive to proactive security postures. The Open VSX vetting gap — in which scanner failures silently permitted publication even after a mandatory scanning mandate was in place — illustrates a broader pattern in which open-source registry projects lack the engineering resources to implement the layered vetting that commercial registries employ. CSA recommends that organizations contributing to or depending upon open-source registries engage with those projects' security working groups to accelerate adoption of mandatory pre-publish scanning, anomalous behavior detection, and publisher identity verification.

The trend toward Zig and other newer systems languages for malware development warrants tooling investment. Traditional signature-based detection is slower to develop for binaries compiled from languages with small install bases and distinct toolchain fingerprints. Organizations should evaluate whether their endpoint security vendors have developed Zig-specific detection heuristics and request roadmap commitments if not.

CSA Resource Alignment

The GlassWorm Zig dropper campaign engages several dimensions of CSA's guidance frameworks.

CSA's AI Controls Matrix (AICM) addresses supply chain integrity for AI systems and developer tooling under its Application Provider controls category. The campaign demonstrates that the attack surface for AI-assisted development environments extends well beyond model security to include the extension ecosystems within which AI coding assistants operate. AICM's guidance on application provider security responsibilities is directly applicable to organizations that have adopted AI-native IDEs as part of their development workflow.

CSA's Cloud Controls Matrix (CCM) Supply Chain Management and Transparency domain (STA-09 through STA-12) addresses third-party component vetting and software bill of materials requirements. The GlassWorm extensions' propagation relied in part on the absence of effective supply chain controls at the extension registry level. Organizations applying CCM STA controls to their cloud deployments should evaluate whether equivalent rigor has been applied to their developer tooling supply chains, including IDE extensions and package managers.

CSA's Software Transparency: Securing the Digital Supply Chain guidance is directly relevant to organizations assessing their exposure to this campaign. That publication addresses CI/CD pipeline security and open-source software risk management in terms directly applicable to the package ecosystem and repository compromise vectors GlassWorm has exploited throughout its campaign lifecycle.

MAESTRO (CSA's Multi-Agent Environment Security Threat Reasoning and Operational framework) applies at the layer where AI-native IDEs execute agentic code-generation and tool-execution tasks. A compromised AI coding assistant that has been manipulated through an infected extension constitutes a MAESTRO Layer 3 (execution environment) threat: the adversary has subverted the environment within which the AI agent operates, enabling downstream manipulation of all agent outputs without direct access to the model itself. Organizations using agentic development environments should review their MAESTRO threat models to include IDE extension compromise as an initial access vector.

CSA's Zero Trust guidance applies to the question of developer workstation access. Zero Trust principles — verify explicitly, least privilege, assume breach — argue against the current common posture in which developer workstations are trusted network participants with broad access to internal services. The GlassWorm RAT's credential harvesting capabilities make the case that developer workstations should be treated as untrusted endpoints requiring continuous verification of identity and behavior.

References

[1] Aikido Security. "GlassWorm goes native: New Zig dropper infects every IDE on your machine." Aikido Security Blog, April 2026.

[2] Aikido Security. "Glassworm Returns: Invisible Unicode Malware Found in 150+ GitHub Repositories." Aikido Security Blog, March 2026.

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

[4] The Hacker News. "GlassWorm Malware Uses Solana Dead Drops to Deliver RAT and Steal Browser, Crypto Data." The Hacker News, March 2026.

[5] The Hacker News. "GlassWorm Attack Uses Stolen GitHub Tokens to Force-Push Malware Into Python Repos." The Hacker News, March 2026.

[6] AFINE. "Hunting GlassWorm: Open-Source Detection for Invisible Supply Chain Payloads." AFINE Blog, 2026. Source code available at github.com/afine-com/glassworm-hunter.

[7] The Hacker News. "Open VSX Bug Let Malicious VS Code Extensions Bypass Pre-Publish Security Checks." The Hacker News, March 2026.

[8] KPMG Netherlands. "Zig Strike: The ultimate toolkit for payload creation and evasion." KPMG Insights, December 2024.

[9] Aikido Security. "GlassWorm Hides a RAT Inside a Malicious Chrome Extension." Aikido Security Blog, March 2026.

[10] Bleeping Computer. "GlassWorm malware hits 400+ code repos on GitHub, npm, VSCode, OpenVSX." Bleeping Computer, March 2026.

[11] Security Affairs. "GlassWorm evolves with Zig dropper to infect multiple developer tools." Security Affairs, April 2026.

[12] The Hacker News. "Eclipse Foundation Mandates Pre-Publish Security Checks for Open VSX Extensions." The Hacker News, February 2026.

