CSAI Foundation

Cloud Security Alliance AI Safety Initiative

Axios npm Compromised: UNC1069 Deploys Cross-Platform RAT

North Korean Supply Chain Attack Targets 100M+ Weekly Downloads via WAVESHAPER.V2 Backdoor

Unofficial AI-assisted Research

2026-04-01

Key Takeaways

- Between 00:21 and 03:29 UTC on March 31, 2026, two trojanized versions of the axios npm package — `1.14.1` and `0.30.4` — delivered a cross-platform remote access trojan to any environment that ran `npm install` during the roughly three-hour exposure window [1][2][4].
- Google Threat Intelligence Group (GTIG) attributes the campaign to UNC1069, a financially motivated North Korea-nexus threat cluster active since 2018 and previously linked to BlueNoroff, a Lazarus Group subunit specializing in cryptocurrency theft [3][12].
- The attack vector was account takeover: the attacker compromised the npm account of the primary axios maintainer (`jasonsaayman`), changed the associated email to an attacker-controlled ProtonMail address, and used a stolen long-lived classic access token to publish directly to the registry — bypassing the legitimate GitHub Actions OIDC Trusted Publisher workflow [4][5].
- The malicious payload, WAVESHAPER.V2, is a fully functional cross-platform RAT supporting arbitrary command execution, filesystem enumeration, in-memory PE injection, and lateral distribution of a secondary Go-based downloader (HYPERCALL). It beacons to its command-and-control server every 60 seconds and self-destructs after initial execution to evade forensic analysis [3][6].
- Wiz estimates axios is present in approximately 80% of cloud and code environments and receives roughly 100 million downloads per week; the firm observed confirmed payload execution in approximately 3% of affected environments [7].
- Organizations that ran `npm install` or `npm ci` between March 30, 23:59 UTC and March 31, 03:29 UTC against any project with an axios dependency — direct or transitive — should treat affected systems as compromised and initiate full credential rotation and system rebuild [2][4].

Background

The axios HTTP client library is one of the most widely depended-upon packages in the JavaScript ecosystem. Available on npm under the package name `axios`, it provides a promise-based interface for making HTTP requests in both Node.js and browser environments, and serves as a direct or transitive dependency for a substantial portion of modern web and cloud-native applications. According to Wiz, the library is present in approximately 80% of cloud and code environments scanned by the firm and receives in excess of 100 million downloads weekly across its current and legacy release branches [7]. This scale makes axios an exceptionally high-value target: a successful compromise does not affect a single application but rather propagates instantly through automated build pipelines across tens of thousands of organizations.

The threat actor behind this campaign, UNC1069, is a financially motivated North Korean-nexus cluster that Google Threat Intelligence has tracked since at least 2018. The group operates within the constellation of DPRK-linked cyber units and shares tooling and operational characteristics with BlueNoroff, a Lazarus Group subunit with an established focus on cryptocurrency exchanges, decentralized finance platforms, and financial technology companies [3][8]. Prior to the axios incident, UNC1069 had drawn attention in early 2026 for deploying AI-generated social engineering lures against cryptocurrency organizations — a tactic shift reflecting the group's ongoing adaptation of its intrusion methods to leverage emerging technology [8]. The group's use of supply chain compromise is not unprecedented; North Korean-affiliated actors have previously exploited the npm registry and other open source distribution channels to target software developers, whose build environments typically contain high-value secrets including cloud credentials, signing keys, and repository access tokens.

The axios attack itself unfolded with a degree of operational planning that security analysts described as among the most sophisticated ever documented against a top-ten npm package [9][11][13]. The threat actor pre-staged a decoy package, `plain-crypto-js`, on the npm registry approximately 18 hours before the attack, first publishing a clean version (`4.2.0`) to establish a brief registry history and reduce automated detection friction. The malicious version (`4.2.1`) was then published at 23:59:12 UTC on March 30, 2026. Shortly thereafter, the attacker used the compromised `jasonsaayman` credentials to publish two backdoored axios releases — `1.14.1` at 00:21 UTC and `0.30.4` at 01:00 UTC on March 31 — covering both the current and legacy release branches simultaneously [4][5]. Neither malicious release had a corresponding commit or tag in the legitimate axios GitHub repository, a discrepancy that should in principle have been detectable but was not actionable in time to prevent widespread exposure.

Socket's automated detection system flagged the malicious `plain-crypto-js@4.2.1` package at 00:05:41 UTC — within minutes of publication — but npm did not complete removal of both compromised axios versions until 03:29 UTC, leaving a window of approximately three hours during which the packages were available for automated download [4]. During this window, any developer running `npm install`, any CI/CD pipeline triggered by a commit, or any scheduled build job that resolved the axios dependency could have installed the trojanized version and silently executed the dropper payload.

Security Analysis

Attack Chain: From Registry to Remote Access

The attack's technical construction exploited a standard npm feature — the `postinstall` hook in `package.json` — that is widely used by legitimate packages for post-installation setup tasks. The malicious `plain-crypto-js@4.2.1` included a hook that automatically triggered execution of an obfuscated JavaScript dropper named `setup.js` during the `npm install` process, before the installing environment had any indication that something was wrong. This means payload execution occurred as an implicit side effect of a routine developer or CI/CD action, with no user interaction beyond the install command itself [4][5].

The dropper decoded its payload using XOR and base64 operations, then selected a platform-specific delivery mechanism. On macOS, an AppleScript download retrieved a binary disguised as an Apple system daemon, storing it at `/Library/Caches/com.apple.act.mond`. On Windows, a hidden VBScript copied the PowerShell interpreter to `%PROGRAMDATA%\wt.exe` and executed a second-stage PowerShell RAT with execution policy bypass flags set. On Linux, a `curl` and Python command chain downloaded a script to `/tmp/ld.py` and launched it via `nohup python3` to ensure persistence across session termination [4][6]. Across all three platforms, the initial C2 contact occurred within approximately two seconds of the `npm install` invocation — before the broader installation process completed [6]. After successfully staging the payload, `setup.js` deleted itself and replaced the malicious `package.json` with a clean stub reporting version `4.2.0`, making post-installation inspection of `node_modules` unreliable as a detection method [4][5].

WAVESHAPER.V2: Capabilities and Architecture

The payload deployed by this dropper chain is WAVESHAPER.V2, an updated version of the WAVESHAPER backdoor previously attributed to UNC1069. WAVESHAPER.V2 represents a meaningful capability expansion over its predecessor: where the original WAVESHAPER used a lightweight raw binary C2 protocol, WAVESHAPER.V2 communicates using JSON, collects a broader set of system telemetry, and supports an expanded command set [3]. Upon execution, the implant begins collecting and transmitting system reconnaissance data — hostname, username, boot time, time zone, OS version, and a detailed enumeration of running processes — to a C2 server at `sfrclak.com` on port 8000 (IP: `142.11.206.73`), beaconing at 60-second intervals using a fake Internet Explorer 8 / Windows XP User-Agent string to blend with aged traffic profiles [3][6].

The implant supports four operational commands. The `kill` command terminates the malware process. The `rundir` command enumerates directories with full file metadata including paths, sizes, and timestamps — enabling the operator to map the victim's filesystem. The `runscript` command executes arbitrary AppleScript, PowerShell, or shell commands depending on the operating system, providing comprehensive arbitrary code execution capability. The `peinject` command decodes and executes arbitrary binaries in memory, facilitating lateral movement, credential harvesting, and further implant staging without writing additional executables to disk [3][6]. In addition, WAVESHAPER.V2 distributes HYPERCALL, a Go-based downloader that serves as a flexible staging mechanism for additional payloads selected by the operator based on post-compromise assessment of the victim environment [3].

The capability profile of WAVESHAPER.V2 is consistent with UNC1069's historically documented objectives. Developer build environments are rich targets: they typically contain npm and cloud provider access tokens, SSH private keys, repository credentials, code signing certificates, environment variable files with database passwords and API secrets, and CI/CD platform credentials. A single compromised developer machine or build pipeline can provide lateral access to production infrastructure, customer data, and financial accounts — outcomes directly aligned with UNC1069's cryptocurrency and financial theft focus [3][8].

Scale and Observed Impact

The scope of potential exposure in this incident is exceptionally broad. With axios receiving over 100 million downloads per week and present in approximately 80% of cloud and code environments, even a three-hour exposure window represents significant real-world reach [1][7]. Wiz reported observing confirmed WAVESHAPER.V2 execution in approximately 3% of affected environments it monitored — a figure that, applied against the full scale of axios's deployment footprint, implies a substantial number of

compromised systems [7]. Security researchers estimated that approximately 600,000 downloads occurred during the three-hour exposure window itself [11][13], and endpoint telemetry from Huntress identified at least 135 confirmed instances of C2 contact in monitored environments — providing a concrete lower bound on confirmed compromises independent of the broader execution-rate estimate [15]. The Register characterized the incident as "among the most operationally sophisticated supply chain attacks ever documented against a top-10 npm package" and noted that security leaders estimated tens of thousands of organizations could have received the malware on a typical day [9][14].

The transitive dependency dimension compounds the risk further. Axios is itself a dependency of many other widely used packages, meaning organizations whose direct dependency manifests did not explicitly reference axios could still have been affected through an indirect chain. This transitive exposure is characteristically difficult to audit without software composition analysis tooling, and it substantially expands the set of organizations that must evaluate their exposure [7].

Detection Signals

Several detection approaches have proven effective. StepSecurity's Harden-Runner identified the compromise through anomalous outbound network behavior — flagging C2 contact to `sfrclak.com:8000` during a routine CI run, a domain that had never appeared in any prior workflow execution [6]. Process tree analysis similarly exposed the dropper, which launched background processes detached to PID 1 with C2 contact occurring within seconds of `npm install` [6]. Snyk published advisory entries `SNYK-JS-AXIOS-15850650` and `SNYK-JS-PLAINCRYPTOJS-15850652` covering both compromised packages [10]. The absence of GitHub commits or tags corresponding to versions `1.14.1` and `0.30.4` in the legitimate axios repository also serves as a reliable post-hoc indicator of malicious publication.

Recommendations

Immediate Actions

Organizations should immediately audit their npm lock files, build logs, and node_modules directories for any reference to `axios@1.14.1`, `axios@0.30.4`, or `plain-crypto-js` at any version. The presence of these identifiers in lock files does not necessarily confirm payload execution — if `npm install --ignore-scripts` was in use, the dropper would not have fired — but their presence warrants investigation [2][4].

Any environment in which these packages were installed without the `--ignore-scripts` flag, particularly during the window from March 30, 23:59 UTC to March 31, 03:29 UTC, should be treated as potentially compromised. Credential rotation is non-negotiable in this scenario: npm access tokens, AWS and cloud provider credentials, SSH private keys, GitHub and GitLab tokens, code signing certificates, and any environment variables or secrets present in the build environment at the time of compromise should be considered exposed and rotated immediately [2][4][5]. Organizations should also block the known C2 infrastructure – `sfrclak.com` and IP `142.11.206.73` on port 8000 – via firewall and DNS controls, and monitor endpoint telemetry for filesystem artifacts at `/Library/Caches/com.apple.act.mond` (macOS), `%PROGRAMDATA%\wt.exe` (Windows), and `/tmp/ld.py` (Linux) [6].

Systems on which payload execution is confirmed or strongly suspected should be rebuilt from known-good state. Given WAVESHAPER.V2's in-memory PE injection capability and its distribution of the HYPERCALL secondary loader, persistence mechanisms beyond the initial file artifacts may have been established and may not be discoverable through artifact scanning alone [3].

Short-Term Mitigations

For ongoing operations, organizations should adopt `npm ci --ignore-scripts` as the standard install command in all CI/CD pipelines. While this will break packages that legitimately depend on postinstall hooks for compilation or setup, the security benefit – preventing automatic execution of untrusted code during installation – outweighs the integration friction, which is typically limited to a small number of native-module dependencies [2][4]. Package managers including npm, pnpm, Yarn, and Bun all support minimum release-age policies that delay installation of newly published versions; a policy of seven or more days significantly reduces exposure to attacks that rely on narrow publication windows, as the axios attack did [2]. Organizations should pin axios to safe versions – `1.14.0` or `0.30.3` – and enforce those pins via `npm overrides` or `resolutions` fields in package manifests to prevent transitive upgrades from reinstating vulnerable versions [4][5].

Software composition analysis tooling that continuously monitors the registry for malicious packages – such as Socket, Snyk, or Wiz's supply chain scanning – provides earlier warning than manual audits. Socket's detection of the malicious `plain-crypto-js` package within six minutes of publication demonstrates that automated scanning can identify these incidents rapidly, even if the window between detection and removal is too short to prevent all exposure [4].

Strategic Considerations

The axios incident illustrates a structural vulnerability in the software supply chain that extends beyond any single package: the trust model of package registries is predicated on the integrity of maintainer credentials, and that trust can be defeated through credential theft without any compromise of the package's source code or build infrastructure. The absence of a corresponding GitHub commit for the malicious axios releases was observable in principle, but no automated enforcement gate prevented those versions from being published or consumed. Requiring that npm packages for high-criticality packages be signed and published exclusively through OIDC-authenticated Trusted Publisher workflows — which tie publication to verifiable CI/CD runs rather than long-lived personal access tokens — would have materially raised the bar for this attack [4][5].

Organizations that depend heavily on open source ecosystem components should formalize their software bill of materials (SBOM) practices, enabling rapid triage when supply chain incidents occur. An SBOM with current transitive dependency resolution allows an organization to answer the question "are we affected?" in minutes rather than hours. At the vendor level, npm has faced renewed calls from the security community to enforce granular token scoping, mandate two-factor authentication for publishers of high-download packages, and provide real-time alerting to downstream consumers when a package's checksum or publisher metadata changes between versions [9].

The use of `plain-crypto-js` as a phantom dependency — a package added to `package.json` but never imported in source code — is a detection evasion technique that automated dependency auditing tools may not surface unless they specifically compare declared dependencies against import graph analysis. Tooling that flags declared-but-unused dependencies, particularly newly introduced ones, represents an underexplored detection surface for this class of attack.

CSA Resource Alignment

This incident maps directly to several threat categories and control domains documented in CSA's published frameworks.

The MAESTRO framework for agentic AI threat modeling identifies **Supply Chain & Dependency Manipulation** as a first-tier threat against AI agent infrastructure. Axios is a dependency of agent frameworks including LangChain, AutoGPT components, and numerous LLM SDK wrappers; environments running AI agents are therefore disproportionately likely to have been in-scope for this attack. MAESTRO's controls for dependency verification, runtime network isolation, and build environment credential segmentation apply directly to the remediation steps outlined above.

Within the AI Infrastructure Control Matrix (AICM) — the CSA framework that supersedes and extends CCM for AI workloads — the relevant control domains are **Supply Chain Risk Management** (controls requiring SBOM generation, integrity verification, and registry provenance), **Identity and Access Management** (controls governing the scope and lifecycle of CI/CD credentials), and **Threat Detection and Response** (controls requiring runtime network monitoring in build environments). The AICM emphasizes that build pipelines are themselves a critical attack surface requiring the same access controls and observability as production systems, a principle directly demonstrated by this incident.

The CSA Zero Trust principles of **verify explicitly**, **use least privilege access**, and **assume breach** map to the concrete controls recommended here: verifying package provenance before installation (not merely trusting the registry), scoping npm tokens to publish-only and enforcing short TTLs, and treating any environment that executed the malicious package as breached pending investigation. CSA's Zero Trust guidance specifically addresses CI/CD environments as trust boundaries requiring explicit policy enforcement rather than implicit trust inheritance from developer machines.

The CSA STAR (Security Trust Assurance and Risk) program provides a framework for organizations to assess and disclose their supply chain security posture. In the context of incidents like this one, STAR Level 2 assessments that incorporate software supply chain controls give enterprise consumers a structured mechanism for evaluating whether vendors and SaaS providers are operating build environments with the controls necessary to detect and respond to this class of attack.

References

[1] Axios.com News. "North Korean hackers implicated in major supply chain attack." Axios, March 31, 2026.

[2] Help Net Security. "Axios npm packages backdoored in supply chain attack." Help Net Security, March 31, 2026.

[3] Google Threat Intelligence Group. "North Korea-Nexus Threat Actor Compromises Widely Used Axios NPM Package in Supply Chain Attack." Google Cloud Blog, April 2026.

[4] Aikido Security. "axios compromised on npm: maintainer account hijacked, RAT deployed." Aikido Dev Blog, March 31, 2026.

[5] Ox Security. "Axios Compromised by Malicious NPM Dependency." Ox Security Blog, March 2026.

[6] StepSecurity. "axios Compromised on npm - Malicious Versions Drop Remote Access Trojan." StepSecurity Blog, March 31, 2026.

[7] Wiz. "Axios NPM Distribution Compromised in Supply Chain Attack." Wiz Blog, March 2026.

[8] Google Threat Intelligence Group. "UNC1069 Targets Cryptocurrency Sector with New Tooling and AI-Enabled Social Engineering." Google Cloud Blog, 2026.

[9] The Register. "Supply chain blast: Top npm package backdoored to drop dirty RAT on dev machines." The Register, March 31, 2026.

[10] Snyk. "Axios npm Package Compromised: Supply Chain Attack Delivers Cross-Platform RAT." Snyk Blog, March 2026.

[11] CyberScoop. "Attack on axios software developer tool threatens widespread compromises." CyberScoop, March 2026.

[12] The Hacker News. "Google Attributes Axios npm Supply Chain Attack to North Korean Group UNC1069." The Hacker News, April 2026.

[13] Malwarebytes. "Axios supply chain attack chops away at npm trust." Malwarebytes Blog, March 2026.

[14] Recorded Future / The Record. "Google links axios supply chain attack to North Korean group." The Record, March 2026.

[15] Huntress. "Supply Chain Compromise of axios npm Package." Huntress Blog, 2026.