CSAI Foundation

Cloud Security Alliance AI Safety Initiative

Axios Poisoned: UNC1069's npm Supply Chain Playbook

Social Engineering of a Critical npm Maintainer Exposes the Open-Source Dependency Trust Model

Unofficial AI-assisted Research

2026-04-03

Key Takeaways

- On March 31, 2026, the npm package `axios` – the most downloaded JavaScript HTTP client library with over 100 million weekly downloads – was compromised after a North Korea-nexus threat actor designated UNC1069 socially engineered the project's lead maintainer [1] [2].
- The attack introduced a malicious transitive dependency (`plain-crypto-js@4.2.1`) that deployed a cross-platform remote access trojan (RAT) identified as WAVESHAPER.V2, a tool previously attributed to BlueNoroff, a financially focused sub-unit of the Lazarus Group [3][4].
- Approximately 600,000 downloads of the poisoned versions occurred during the roughly three-hour exposure window before the packages were removed [5].
- The attacker bypassed GitHub Actions OIDC-based trusted publishing protections by using a long-lived classic npm access token harvested from the maintainer's session after RAT deployment – a technique that circumvents many modern supply chain hardening measures [6].
- Automated scanning by Socket.dev detected the malicious version within approximately six minutes of publication, underscoring the value of continuous, real-time dependency monitoring over point-in-time audits [1].
- The incident represents a tactical evolution for UNC1069: the group moved from targeting cryptocurrency executives with AI-generated deepfakes to directly targeting the open-source maintainers whose trusted credentials provide a scalable route into enterprise software pipelines [3].

Background

UNC1069 and the BlueNoroff Connection

UNC1069 is a financially motivated advanced persistent threat group tracked by Google Mandiant since at least April 2018, with assessed links to North Korean state-directed operations [3]. The group is also identified in industry reporting as CryptoCore, MASAN, Dangerous Password, and Leery Turtle; Microsoft tracks overlapping activity as Sapphire Sleet (formerly CageyChameleon) [3][7]. Between 2018 and 2020, the group stole an estimated minimum of $200 million from cryptocurrency exchanges, primarily in the United States and Japan [3]. Since at least 2023, UNC1069's observed targeting has expanded toward developer communities building Web3 infrastructure and centralized exchanges, suggesting a deliberate strategic evolution [3].

What distinguishes UNC1069 from generically opportunistic actors is its sustained investment in social engineering sophistication. By early 2026, Google Threat Intelligence Group documented the group's deployment of AI-generated deepfakes and real-time audio impersonation during live video calls, use of generative AI tools including Google Gemini to produce lure materials, and operation of a seven-family malware arsenal including WAVESHAPER, HYPERCALL, HIDDENCALL, DEEPBREATH, SUGARLOADER, CHROMEPUSH, and SILENCELIFT [3]. This arsenal is consistent with an organization conducting sustained research and development, rather than an opportunistic criminal group repurposing commodity tooling [3].

The Axios Ecosystem and Its Attack Surface

Axios is the most widely used JavaScript HTTP client library in the npm ecosystem, providing a promise-based interface for making HTTP requests in both Node.js server environments and web browsers. As of March 2026, the 1.x branch recorded over 100 million weekly downloads, the legacy 0.x branch an additional 83 million, and the library counted approximately 174,000 dependent npm packages in its upstream dependency graph [1][5]. Wiz Research estimates that Axios is present in roughly 80 percent of cloud and code environments [5]. These numbers make a successful Axios compromise qualitatively different from the typical supply chain attack: rather than targeting niche packages to reach a handful of specific applications, an adversary targeting Axios reaches an estimated 80 percent of cloud and code environments in a single operation [5].

Security Analysis

Phase 1: Maintainer Compromise via Social Engineering

The attack began approximately two weeks before package publication with a carefully constructed social engineering campaign against Jason Saayman (`@jasonsaayman`), the lead Axios maintainer [1]. Attackers cloned the identity and company profile of a legitimate, well-known company founder and constructed a fraudulent Slack workspace that convincingly replicated a corporate CI/CD environment, complete with plausible channel naming and shared LinkedIn artifacts to establish authenticity. Saayman was invited under the guise of a collaboration opportunity and subsequently directed to a Microsoft Teams meeting where a fabricated system error message prompted a download that delivered WAVESHAPER.V2 to his machine [1].

In his own post-mortem account, Saayman described the campaign as "a coordinated social engineering campaign involving cloned company identities, a convincing Slack workspace, staged meetings, and ultimately a malicious install that provided remote access" [1]. The RAT enabled session and cookie hijacking from his browser, allowing the attackers to lift his npm credentials and a long-lived classic npm access token without triggering two-factor authentication protections. The attackers then altered the maintainer account's registered email from `jasonsaayman@gmail.com` to `ifstap@proton.me` in npm registry metadata before proceeding to publication [6].

The choice of approach merits attention. A direct credential-stuffing or phishing attack against the npm account would likely trigger anomaly detection. By compromising the maintainer's device through social engineering and harvesting a session token that already represented an authenticated state, UNC1069 operated largely within what the registry's trust model treats as legitimate activity.

Phase 2: Package Introduction and Payload Delivery

Rather than modifying the axios source code directly — a change that would be visible to community code review — the attackers introduced a new dependency into two poisoned releases: `axios@1.14.1` (published March 31 at 00:21 UTC as the `latest` tag) and `axios@0.30.4` (published at 01:00 UTC) [2][4]. The malicious dependency, `plain-crypto-js@4.2.1` , had been staged the previous day as a typosquat-style package mimicking the well-known `crypto-js` library, with a clean decoy version (`4.2.0`) published first, apparently to establish plausibility [4].

On installation, `plain-crypto-js@4.2.1`'s `postinstall` hook executed `node setup.js`, triggering a dropper that applied two obfuscation layers — string reversal with Base64 decoding and a byte-wise XOR cipher using key `OrDeR_7077` and constant `333` — before deploying platform-specific payloads [4]. Following execution, the dropper performed aggressive anti-forensic cleanup: it deleted `setup.js`, removed the `postinstall` hook entry, and replaced the tampered `package.json` with a clean version that had been staged as `package.md` [4]. The resulting installation state appeared clean to post-hoc inspection.

The full timeline from first malicious package publication to complete removal was approximately three hours and nineteen minutes [2][4]. During that window, Socket.dev's automated scanner detected the anomaly within roughly six minutes of the first malicious version appearing on the registry [1].

Phase 3: WAVESHAPER.V2 — Cross-Platform RAT with Attribution Links

The deployed payload, identified by Mandiant as WAVESHAPER.V2, produced platform-specific remote access trojans with varying levels of completeness [3][4]. On macOS, the dropper placed a Mach-O universal binary supporting both x86_64 and ARM64 architectures at `/Library/Caches/com.apple.act.mond`, deliberately spoofing Apple daemon naming conventions and launched via AppleScript to blend with expected system behavior [4]. The macOS binary carried embedded build artifacts at the developer path `/Users/mac/Desktop/Jain_DEV/client_mac/macWebT/`, which Datadog Security Labs identified as directly linking to BlueNoroff's documented RustBucket webT module from 2023 — the most technically specific attribution link beyond infrastructure overlap [4].

On Windows, the payload copied the PowerShell binary to `%PROGRAMDATA%\wt.exe` to impersonate Windows Terminal and established persistence via a registry Run key at `HKCU\Software\Microsoft\Windows\CurrentVersion\Run\MicrosoftUpdate` [4]. Datadog Security Labs confirmed, however, that the Windows variant's `work()` function is never called, meaning the Windows RAT establishes persistence but does not beacon to its C2 infrastructure — whether due to a code defect or deliberate staging for later activation [4]. This ambiguity does not eliminate the need for Windows forensic review. The Linux variant was similarly incomplete, with `os.getlogin()` crashing in containerized or CI environments and a `peinject` function referencing an undefined variable [4].

Command-and-control communications were directed to `sfrclak[.]com` (registered March 30, 2026 — one day before the attack) at IP `142.11.206.73` on port 8000, with a secondary domain at `callnrwise[.]com` [4]. The RAT's User-Agent string was spoofed as Internet Explorer 8 running on

Windows XP — an anomalous signature that Elastic Security Labs identified as a reliable detection indicator in environments where such strings are no longer expected [8]. The RAT supported four commands: `kill` (self-terminate), `runscript` (execute platform-native code), `peinject` (drop and execute a binary payload), and `rundir` (filesystem enumeration) [4].

Scope and Impact Assessment

Wiz Research telemetry indicated that approximately 3 percent of affected environments showed confirmed RAT execution [5], a figure that, when applied against an estimated 600,000 malicious downloads, implies approximately 18,000 environments with confirmed RAT execution. The macOS variant was the only fully functional payload; the Windows and Linux variants, despite establishing persistence, did not achieve operational C2 communication due to the payload defects described above. The incident's harm appears to have been concentrated in macOS developer environments — the population most likely to have installed a new axios version within the brief exposure window.

Three additional packages were identified as distributing the compromised versions beyond the direct axios installs: `@qqbrowser/openclaw-qbot@0.0.130` vendored a tampered `axios@1.14.1`, while `@shadanai/openclaw@2026.3.31-1` and `@2026.3.31-2` vendored `plain-crypto-js` directly [6]. These packages represent a secondary distribution channel that persisted briefly after the primary axios versions were removed.

The GitHub Advisory Database assigned identifier `GHSA-fw8c-xr5c-95f9` with critical severity under CWE-506 (Embedded Malicious Code), covering both `axios@1.14.1` and `axios@0.30.4` [9]. Snyk issued four advisories: `SNYK-JS-AXIOS-15850650`, `SNYK-JS-PLAINCRYPTOJS-15850652`, `SNYK-JS-QQBROWSEROPENCLAWQBOT-15850776`, and `SNYK-JS-SHADANAIOPENCLAW-15850775` [6].

Tactical Evolution and Broader Significance

The UNC1069 Axios operation is not an isolated event but an extension of a documented operational pattern. The group's February 2026 campaigns used AI-generated deepfakes and Calendly or Zoom invitation lures to target cryptocurrency executives [3]. The Axios operation adapted this social engineering infrastructure — fabricated identities, controlled meeting environments, malicious downloads dressed as routine system activity — to target a software maintainer rather than a financial actor. The distinction matters: a compromised executive yields access to one organization's funds or data, while a compromised maintainer yields a distribution vector into every organization that installs their package.

This targeting shift also signals adversary awareness of the developer ecosystem's trust model. The npm registry, like most package ecosystems, treats maintainer credentials as the primary trust anchor. Continuous integration pipelines in most organizations are configured to automatically pull updated dependencies, meaning a malicious `latest` tag on a widely used package becomes operational in production environments within hours without human review. UNC1069's decision to poison a transitive dependency rather than modify axios source directly further reflects an understanding of when code review attention is highest: direct changes to axios would attract scrutiny from contributors and security researchers; a new transitive dependency added by the maintainer would not.

The XZ Utils incident of 2024 demonstrated that a patient, multi-year social engineering campaign could insert a backdoor into critical open-source infrastructure [10]. The Axios operation demonstrated that the same objective can be achieved in a matter of weeks when the target is a sole maintainer of a heavily depended-upon package rather than a committee-governed project. Together, the two incidents represent among the most significant recent demonstrations of the threat to open-source supply chain trust, and they illustrate the trajectory that sophisticated actors follow when exploiting maintainer trust models.

Recommendations

Immediate Actions

Organizations should audit their software bills of materials (SBOMs) and dependency lock files for any reference to `axios@1.14.1`, `axios@0.30.4`, or `plain-crypto-js@4.2.1`. Any environment that installed these versions should be treated as potentially compromised and investigated for the presence of the IOCs documented in published advisories, including the filesystem artifacts at `/Library/Caches/com.apple.act.mond` (macOS), `%PROGRAMDATA%\wt.exe` (Windows), and `/tmp/ld.py` (Linux), as well as network activity to `sfrclak[.]com` or `callnrwise[.]com` [4][9]. Because the Windows and Linux payloads did not achieve operational C2 communication, macOS developer environments warrant the highest priority in forensic review — though the ambiguity around the Windows payload's design intent means Windows systems should not be excluded from investigation.

Dependency lock files (`package-lock.json`, `yarn.lock`, `pnpm-lock.yaml`) should be regenerated against clean versions of axios — `1.14.0` (1.x branch) or `0.30.3` (0.x branch), the last clean releases before the compromise — and pipeline caches should be fully invalidated [9][13].

Organizations relying on vendored or mirrored package registries should verify that the malicious versions have not been cached internally.

Short-Term Mitigations

Among the architectural failures this incident exposes is the absence of cryptographic commit signing and trusted publishing requirements for high-impact packages. Organizations should adopt `npm audit signatures` to verify published package integrity against registry-provided signatures, enforce `provenance` metadata checks where available, and require that any package exceeding a defined download threshold or transitive dependency count implement OIDC-based trusted publishing through GitHub Actions or an equivalent attestation mechanism [11][14].

Real-time dependency monitoring should be treated as a mandatory control rather than an optional enhancement. Socket.dev's detection of the malicious version within six minutes of publication represents an order-of-magnitude improvement over periodic SCA scanning, which would have missed the attack entirely given its sub-four-hour exposure window [1]. Security teams should evaluate continuous monitoring tooling — including Socket.dev, Snyk Container, or GitHub Dependency Review — and ensure these checks are integrated into CI gating rather than advisory reporting only.

Maintainer accounts for packages meeting an organizational criticality threshold — defined by weekly download count, number of transitive dependents, or presence in production services — should be subject to hardware security key enrollment on the npm registry. npm's support for hardware tokens provides a meaningful barrier against the session-hijacking technique UNC1069 employed, as harvesting a session cookie does not yield a hardware token's response to a challenge [11][14].

Strategic Considerations

The Axios incident makes the case for organizational policies that treat open-source dependency intake as a trust boundary analogous to vendor software procurement. This means requiring SBOMs from all deployed applications, establishing automated alerts on any dependency version change without prior internal review, and defining explicit policies for how quickly automated dependency update pull requests from tools such as Dependabot or Renovate should be merged for different package criticality tiers.

At the maintainer community level, there is a structural question about the sustainability of sole-maintainer stewardship for packages of Axios's scale. A package with 100 million weekly downloads and 174,000 dependent packages is functionally critical infrastructure, yet its security posture depends entirely on a single individual's resistance to sophisticated state-sponsored social engineering. The

open-source community, ecosystem stewards such as npm and the OpenSSF, and organizations that derive commercial value from widely used packages should assess whether this model is consistent with the risk profile it implies. Initiatives such as the OpenSSF's Securing Critical Projects working group and the Alpha-Omega project represent partial responses to this problem, but they have not yet scaled to cover the breadth of high-impact sole-maintainer packages [12].

For security operations teams, the anomalous Internet Explorer 8 User-Agent string identified by Elastic Security Labs as WAVESHAPER.V2's C2 signature (`mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)`) provides a high-fidelity detection rule for network inspection and endpoint detection platforms [8]. Any process making outbound HTTP POST requests with this User-Agent in a contemporary environment should be treated as an immediate incident indicator.

CSA Resource Alignment

This incident maps directly to several active research and framework areas within the Cloud Security Alliance AI Safety Initiative.

MAESTRO (Multi-Agent Environment and System Threat, Risk, and Operational analysis) provides the relevant threat modeling lens for agentic AI systems that consume npm packages at runtime. When AI coding agents, autonomous deployment pipelines, and CI/CD orchestration tools automatically resolve and install dependencies without human review, they become direct vectors for supply chain compromise. MAESTRO's Layer 5 (Agent Workflow Orchestration) and Layer 6 (Infrastructure and Tooling) controls apply to the governance of what packages agentic systems are permitted to install and under what validation conditions.

CSA AI Controls Matrix (AICM) maps supply chain integrity controls under its Infrastructure Security and Application Security domains. The Axios incident specifically implicates controls governing software component verification, dependency integrity attestation, and privileged credential management. Organizations using the AICM as their AI security control baseline should review whether their implementation coverage extends to the npm publishing token lifecycle for packages consumed by AI systems.

STAR (Security Trust Assurance and Risk) Program assessments of AI vendors and SaaS providers should include explicit questions about the provenance and integrity verification of JavaScript dependencies, particularly for web-delivered AI products and browser-based agents that may include axios or similar packages in their client-side bundles.

CSA Zero Trust Guidance applies at the package registry boundary. Zero Trust principles require that no entity — including a package signed by a trusted maintainer — be automatically granted execution privilege. Translating Zero Trust into the dependency management context means treating each dependency update as an untrusted input requiring verification before promotion to production, rather than relying on registry trust anchors as a sufficient control.

The incident also reinforces themes from CSA's prior research on AI-enabled social engineering threats. UNC1069's use of AI-generated deepfakes, fabricated corporate identities, and generative AI lure materials aligns with threat patterns documented in CSA guidance on AI-augmented phishing and insider threat scenarios, and suggests that developer communities — not only executive targets — require dedicated security awareness programs addressing these techniques.

References

[1] Socket.dev. "Axios Maintainer Confirms Social Engineering Behind npm Compromise." Socket Security Blog, March 2026.

[2] Sonatype. "Axios Compromise on npm Introduces Hidden Malicious Package." Sonatype Blog, March 2026.

[3] Google Cloud / Mandiant. "North Korea Threat Actor Targets Axios NPM Package." Google Cloud Blog, April 2026.

[4] Datadog Security Labs. "Axios npm Supply Chain Compromise." Datadog Security Labs, April 2026.

[5] Wiz Research. "Axios NPM Distribution Compromised in Supply Chain Attack." Wiz Blog, April 2026.

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

[7] Malpedia. "UNC1069 Actor Profile." Fraunhofer FKIE, accessed April 2026.

[8] Elastic Security Labs. "Axios: One RAT to Rule Them All." Elastic Security Labs, April 2026.

[9] GitHub Advisory Database. "GHSA-fw8c-xr5c-95f9: Embedded Malicious Code via Compromised Maintainer Account." GitHub, March 2026.

[10] Freund, Andres. "backdoor in upstream xz/liblzma leading to ssh server compromise." Openwall oss-security mailing list, March 29, 2024.

[11] Tenable. "FAQ About the Axios npm Supply Chain Attack by North Korea-Nexus Threat Actor UNC1069." Tenable Blog, April 2026.

[12] Open Source Security Foundation (OpenSSF). "Alpha-Omega Project." OpenSSF, accessed April 2026.

[13] Microsoft Security. "Mitigating the Axios npm Supply Chain Compromise." Microsoft Security Blog, April 2026.

[14] npm. "Generating provenance statements." npm Documentation, accessed April 2026.