CSAI Foundation

Cloud Security Alliance AI Safety Initiative

UNC1069 Axios Supply Chain Attack: AI Vendor Code-Signing at Risk

How North Korean Social Engineering of an Open Source Maintainer Cascaded to AI Infrastructure

Unofficial AI-assisted Research

2026-04-15

Key Takeaways

On March 31, 2026, the North Korea-nexus threat actor UNC1069 published two backdoored versions of the Axios npm package — one of the most downloaded JavaScript libraries in the world — after a weeks-long social engineering campaign compromised the account of its lead maintainer. The attack delivered the WAVESHAPER.V2 remote access trojan to any system that installed the affected packages during a roughly three-hour exposure window. The downstream consequences reached AI vendor infrastructure directly: a GitHub Actions workflow used in OpenAI's macOS application code-signing process executed the malicious Axios version and had access to certificates used to sign ChatGPT Desktop, Codex, Codex-cli, and Atlas. This incident illustrates how adversaries are now treating open source maintainers as a privileged access vector into AI company build pipelines.

- UNC1069 spent approximately two weeks on targeted, AI-assisted social engineering before compromising the Axios maintainer's system and npm credentials, indicating a deliberate, high-patience campaign rather than opportunistic exploitation [9].
- The malicious versions (axios 1.14.1 and 0.30.4) were detected and removed within three hours on March 31, 2026, yet with approximately 100 million weekly downloads, the potential exposure was substantial given the download volume [2][6].
- OpenAI was a confirmed downstream casualty: its macOS app-signing workflow executed the malicious axios version and had access to code-signing material. OpenAI subsequently rotated certificates and published new builds of all affected applications [3].
- A critical structural weakness enabled the attack: the Axios project had adopted OIDC-based publishing with SLSA provenance attestations, but a legacy long-lived npm token remained active alongside those credentials. When npm receives both, it defaults to the token — effectively nullifying the provenance infrastructure [4].
- This incident is not isolated. UNC1069-linked activity is estimated to have seeded approximately 1,700 malicious packages across npm, PyPI, Go, and Rust since January 2025, reflecting an industrialized approach to malicious package insertion across open source registries [5].

Background

UNC1069 is a financially motivated threat actor with high-confidence North Korea nexus, tracked by Mandiant (Google Threat Intelligence Group, or GTIG) since 2018 and overlapping with designations including BlueNoroff, Sapphire Sleet, and Stardust Chollima [6]. The group's original focus centered on cryptocurrency exchanges and financial services, but its tactics have evolved materially over the past two years. Since at least 2023, UNC1069 has shifted from conventional spear-phishing toward more elaborate social engineering operations targeting software developers, venture capital professionals, and individuals at high-technology firms — particularly in the Web3 ecosystem [7].

The group's incorporation of AI-generated video and audio into its social engineering tradecraft marks a significant escalation. Rather than relying on text-based impersonation, UNC1069 now constructs convincing video personas and cloned organizational identities — complete with realistic Slack workspaces populated with synthesized LinkedIn content — to build rapport with targets over days or weeks before deploying any malicious payload [8]. This patience-and-precision model makes the group's operations considerably more difficult to detect than traditional credential phishing and may overwhelm conventional awareness training designed for lower-fidelity lures.

The Axios npm package sits at the foundation of the modern JavaScript ecosystem. It functions as the de facto HTTP client for a broad range of applications, from browser-based single-page applications to server-side Node.js services and CI/CD pipelines. With approximately 100 million weekly downloads as of March 2026 [6], Axios is present across a vast range of organizations that build or deploy JavaScript-based software. AI companies, which rely heavily on JavaScript and TypeScript for desktop applications, web interfaces, and build automation, are among its most significant users. This ubiquity made it a high-value target: a compromise of Axios, even for hours, represents access to a supply chain node touching millions of downstream systems.

Security Analysis

The Social Engineering Campaign

The compromise of the Axios npm package began not with a vulnerability or a credential-stuffing attack but with a carefully constructed false identity. According to the Axios maintainer's own post-mortem account, attackers approached Jason Saayman impersonating the founder of a legitimate, well-known company [9]. The impersonation was not superficial: the threat actor had constructed an AI-cloned

video likeness of the executive and established a convincing Slack workspace bearing the company's branding, including populated channels sharing synthesized LinkedIn activity to simulate an active organization.

After establishing trust through approximately two weeks of low-pressure interaction, the attackers invited Saayman to a meeting through Microsoft Teams. When he joined the call, a ClickFix-style lure appeared — a fake error message claiming the user's system was not functioning correctly and directing them to download a "corrected" Teams or Zoom SDK [10]. This technique exploits the natural instinct to resolve a technical obstacle in the moment. Depending on the victim's operating system, the malicious payload executed as an AppleScript on macOS or a PowerShell script on Windows, installing a remote access trojan that gave the attackers persistent access to Saayman's system. From there, they extracted the npm credentials stored on his machine.

The sophistication of this campaign reflects what several security researchers have described as the "industrialization" of social engineering [8]. The barriers to building credible executive video deepfakes have fallen substantially; the infrastructure to clone a corporate Slack workspace requires minimal effort. What once demanded nation-state resources to execute against a single target can now be replicated across many targets concurrently.

The Supply Chain Mechanism

With access to Saayman's npm account, the attackers executed a deliberate, staged injection. On March 30, 2026 at approximately 05:57 UTC, they published a clean version of a previously unknown package — plain-crypto-js@4.2.0 — to the npm registry [4]. The attackers published this clean version approximately eighteen hours before inserting the malicious payload; this sequencing likely served as a trust-building measure, giving the package a brief registry history to reduce anomaly scores for age-based automated scanning tools. A few hours before midnight on March 30, they published plain-crypto-js@4.2.1 containing the malicious payload. At 00:21 UTC on March 31, they published the backdoored axios versions 1.14.1 and 0.30.4, each declaring plain-crypto-js as a dependency. The malicious versions were removed at approximately 03:20 UTC — a window of just under three hours [4] [11].

The payload delivery mechanism relied on npm's postinstall hook, which executes automatically during package installation with no user interaction required. The plain-crypto-js setup.js script used a two-layer encoding scheme to conceal its intent: string reversal combined with Base64 decoding, followed by an XOR cipher using a hardcoded key [4]. Upon execution, the script communicated with a command-and-control server at the domain sfrclak.com (also associated with IP address 142.11.206.73), using a second-stage payload delivery mechanism to install WAVESHAPER.V2 on the victim's machine.

WAVESHAPER.V2 is a cross-platform remote access trojan attributed to UNC1069 by GTIG based on code overlap with previously observed WAVESHAPER variants [6]. The backdoor collects system telemetry including hostname, username, boot time, time zone, OS version, and a running process list, then beacons to its C2 server every 60 seconds using Base64-encoded JSON. A particularly diagnostic indicator is the trojan's hardcoded User-Agent string, which spoofs Internet Explorer 8 on Windows XP — an anachronism that functions as a server-side routing key but renders the C2 traffic detectable via standard User-Agent string filtering on any modern network monitoring stack [4][11].

Downstream Impact: AI Vendor Code-Signing Infrastructure

The most significant downstream casualty of the Axios supply chain compromise was OpenAI's macOS application build pipeline. On March 31, 2026, a GitHub Actions workflow used in OpenAI's code-signing process executed as part of its normal CI/CD operation and, in doing so, downloaded and ran axios version 1.14.1 — the compromised release [3]. This workflow had access to the certificate and notarization credentials used to sign OpenAI's macOS desktop applications, including ChatGPT Desktop, Codex, Codex-cli, and Atlas [20].

OpenAI's subsequent forensic analysis, conducted with a third-party incident response firm, concluded that the signing certificate was likely not successfully exfiltrated by the malicious payload. The conclusion was based on the timing of payload execution, the sequencing of certificate injection into the GitHub Actions job, and other mitigating factors [3]. However, the operative word is "likely." OpenAI chose to treat the potential exposure as a worst-case scenario and responded accordingly: it rotated its macOS code-signing certificate entirely, published new signed builds of all affected applications, and initiated a process with Apple to ensure that applications signed under the prior certificate could no longer be newly notarized. OpenAI has announced that beginning May 8, 2026, applications signed under the prior certificate will be blocked from launch following Apple's revocation action [3][12].

OpenAI's response — certificate rotation, application rebuilds, and revocation initiation — addresses the worst-case exposure scenario regardless of whether exfiltration occurred. The underlying exposure is nonetheless significant for the AI security community. Code-signing certificates are a foundational trust anchor for software distribution. A compromised AI vendor certificate, in an adversary's hands, could be used to distribute malware signed with a trusted identity — a capability that would undermine the integrity of AI model delivery, CLI tooling distribution, and desktop application update mechanisms simultaneously. The fact that this outcome was narrowly avoided does not diminish the structural lesson: AI vendors' CI/CD pipelines are reachable through open source dependencies, and any workflow with signing privileges is a high-value target.

Structural Failure: Provenance Attestations Without Revocation of Legacy Tokens

A critical architectural gap enabled the attack to succeed despite the Axios project having adopted modern supply chain security practices. Legitimate Axios releases were published through a GitHub Actions OIDC publisher flow that generated SLSA provenance attestations — cryptographic metadata linking each npm package to a specific, verifiable GitHub Actions run [4]. These attestations are a meaningful supply chain integrity control: they allow consumers to verify that a given package was built from a specific commit in a known repository rather than assembled offline by an unknown party.

The malicious versions lacked any SLSA provenance metadata. They were published directly via the npm CLI using the compromised maintainer's long-lived npm token. The absence of provenance on new versions of a high-visibility package should, in principle, have triggered automated alerts. But it did not — in part because tooling to enforce provenance verification at install time remains immature and inconsistently deployed, and in part because the legacy token gave the attacker a legitimate path around the OIDC flow entirely.

The core problem is that the project's security posture was only as strong as its weakest credential. The long-lived npm token had not been revoked when the OIDC-based flow was introduced. When npm received both forms of authentication, it defaulted to the token — making the classic token the actual authentication method for every publish, regardless of OIDC configuration [4]. This misconfiguration pattern may be common among projects that have adopted provenance workflows, where legacy tokens remain active alongside newer OIDC flows without owners realizing those tokens represent an active bypass. The Axios case illustrates a risk that likely exists across many similarly structured projects.

Per post-incident analysis [4][18], GitHub has announced an accelerated timeline for mandatory short-lived tokens for packages with more than one million weekly downloads, with enforcement planned for June 2026. GitHub also committed to a "critical package" designation for the top 500 most-depended-upon packages, requiring additional identity verification and mandatory two-person publish approval. These are meaningful structural improvements, but they will not be fully in place before the next high-value package comes under similar pressure.

Broader Campaign Context

The Axios compromise is best understood as a targeted operation within a much larger, sustained campaign. Since January 2025, researchers have attributed the publication of approximately 1,700 malicious packages across npm, PyPI, Go, and Rust to UNC1069-linked activity [5]. This campaign — variously described as an evolution of the "Contagious Interview" operation — uses typosquatting, dependency confusion, and, as in the Axios case, account takeover to introduce malicious packages into

widely used registries. The consistent payload across these packages is a credential harvester or remote access trojan designed to steal cryptocurrency wallet keys, developer credentials, and corporate access tokens [19].

The Security Alliance (SEAL) reported blocking 164 UNC1069-linked domains impersonating services such as Microsoft Teams and Zoom between February 6 and April 7, 2026 alone [5]. This infrastructure scale indicates that the social engineering operation is not artisanal — individual targeting of a single maintainer like Jason Saayman represents the high end of effort, but the group simultaneously runs lower-fidelity lures at volume across LinkedIn, Telegram, and Slack [18]. The combination of precision targeting for high-value supply chain positions and mass-scale lure distribution reflects a mature, division-of-labor operational model.

Recommendations

Immediate Actions

Security teams with JavaScript/TypeScript in their development stack should audit recent `npm install` logs and dependency trees for axios versions 1.14.1 and 0.30.4, as well as the presence of the plain-crypto-js package in any installed version [11]. Any system that installed these versions during the window between March 31, 2026 00:21 UTC and 03:20 UTC should be treated as potentially compromised: isolate the system, initiate an incident investigation, rotate all credentials stored in environment variables or shell profiles on that machine, and examine outbound connections to sfrclak.com and 142.11.206.73. Indicators of compromise, including file hashes and network IOCs, have been published by Elastic Security Labs [11]; companion detection rules are available through Elastic's detection engineering resources. These IOCs are also maintained in a community GitHub Gist [13].

Organizations running AI vendor software on macOS should verify that they are running versions signed after OpenAI's certificate rotation. OpenAI has published guidance on which versions carry the new certificate and has indicated that apps signed under the prior certificate will be blocked from launch following Apple's revocation action on May 8, 2026 [3][12].

Short-Term Mitigations

All projects that publish npm packages should immediately audit the relationship between their OIDC-based publishing workflows and any legacy long-lived tokens. The coexistence of both represents a provenance bypass that can be exploited even when OIDC is correctly configured: revoke all classic npm

tokens for accounts where OIDC or granular access tokens have been fully adopted. Revoking the legacy token would likely have prevented this specific compromise vector, since the malicious publish was authenticated via that token rather than via the OIDC flow.

Downstream consumers should evaluate whether their CI/CD pipelines enforce provenance verification at install time. The npm `--provenance` flag and tooling from OpenSSF's Supply Chain Security project can validate that packages carry attestations before installation proceeds. For high-criticality pipelines — particularly those with access to code-signing credentials, deployment keys, or model weights — absence of provenance on a new version of a major package should be treated as a blocking condition rather than a warning.

Social engineering awareness programs should be updated to reflect the current threat model for developer-facing campaigns. The UNC1069 approach does not rely on obvious pretexts: it involves multi-week relationship building through realistic AI-generated personas in professional communication channels. Training that focuses on "don't click suspicious links" is inadequate against an adversary who schedules a calendar meeting, joins a Teams call, and then surfaces a plausible-seeming error message. Developers with npm publish rights, repository admin access, or signing credentials should receive targeted training that covers ClickFix-style lures specifically.

Strategic Considerations

AI companies should formally classify their code-signing workflows as critical security infrastructure and apply controls commensurate with that classification. This means, at minimum: storing signing certificates in hardware security modules or a dedicated signing service rather than as GitHub Actions secrets, enforcing two-person approval for any workflow step that accesses signing material, and maintaining an air-gapped copy of the signing workflow that can be substituted if the primary workflow is suspected of compromise. The OpenAI incident demonstrated that these workflows are reachable through ordinary npm dependencies — treating them as privileged infrastructure, not just automation, is the appropriate posture.

The broader lesson of the Axios attack is that human-layer compromise has emerged as a leading initial access vector against software supply chains, as evidenced by this and related incidents. Technical controls on package integrity — SLSA, Sigstore, provenance attestations — are necessary but insufficient if the human account holding publish rights can be socially engineered into installing a RAT. Supply chain security programs must pair artifact integrity verification with privileged account hygiene, maintainer security assistance, and insider threat detection for accounts with publish access to high-criticality packages.

At an industry level, the AI sector would benefit from coordinating with open source foundations and registries to establish an "AI vendor dependency" designation for packages whose compromise could reach AI application code-signing, model delivery, or agent tool execution pipelines. Packages carrying this designation would be candidates for the enhanced security requirements GitHub is developing for its "critical package" program. CSA's AI Safety Initiative intends to develop criteria for this designation in collaboration with the Open Source Security Foundation (OpenSSF) and the broader AI Safety working community.

CSA Resource Alignment

This incident maps directly to several layers of CSA's MAESTRO agentic AI threat modeling framework [14]. Layer 5 (External Integrations) captures the threat surface that the Axios attack exploited: AI vendor build pipelines depend on external package registries, and a compromise at that integration layer cascades inward to signing infrastructure and, potentially, to deployed agent capabilities. Layer 6 (Deployment and Infrastructure) covers the code-signing workflows themselves — the pipelines through which AI vendor software reaches end users. MAESTRO's framework explicitly identifies CI/CD pipeline compromise and supply chain poisoning as high-priority threat scenarios for AI systems and provides a structured approach to threat modeling these attack surfaces in the context of agentic deployments [14] [15].

CSA's AI Controls Matrix (AICM) addresses supply chain risk through its control domains covering third-party dependency management and software integrity verification. The AICM's control guidance on artifact integrity aligns with the SLSA provenance framework and the principle that AI development pipelines require verified build provenance at every stage from training data through deployed application. The Axios incident demonstrates that this guidance applies not only to ML model artifacts but to all software components in an AI application's dependency graph.

The CSA Cloud Controls Matrix (CCM) domain on Change Control and Configuration Management (CCC) is directly applicable to the credential hygiene failure that enabled this attack. CCM controls governing revocation of unused credentials and enforcement of least-privilege publishing access would, if applied, have removed the legacy npm token that served as the attack vector once the social engineering campaign succeeded.

CSA's ongoing research on securing non-human identities in AI agent environments is also relevant here [16]. The npm publish token is a non-human identity: it represents a machine-level credential that, when compromised, grants software publishing rights without triggering the human authentication controls

that might otherwise detect an intrusion. Treating publish tokens with the same rigor applied to service account credentials — including time-limited issuance, automatic rotation, and usage auditing — is a direct application of the principles that work addresses.

Finally, CSA's prior research note on the TeamPCP CI/CD Infrastructure Campaign [17] identified a contemporaneous wave of attacks targeting developer infrastructure in March 2026. The Axios compromise and the TeamPCP campaign represent parallel vectors that may reflect a coordinated adversary posture against the developer supply chain, or at minimum a sustained campaign focus on developer supply chain targets. Organizations responding to either incident should evaluate their exposure to both.

References

[1] Mandiant / Google Cloud. "UNC1069 Targets Cryptocurrency Sector with New Tooling and AI-Enabled Social Engineering". Google Cloud Blog, February 2026.

[2] The Hacker News. "UNC1069 Social Engineering of Axios Maintainer Led to npm Supply Chain Attack". The Hacker News, April 2026.

[3] OpenAI. "Our response to the Axios developer tool compromise". OpenAI, April 2026.

[4] Dev|Journal. "Securing the npm Supply Chain: Lessons from the 2026 Axios Attack – npm Provenance and SLSA". Dev|Journal, April 4, 2026. (Secondary technical analysis drawing on primary forensic data from Elastic Security Labs [11] and community analysis.)

[5] The Hacker News. "N. Korean Hackers Spread 1,700 Malicious Packages Across npm, PyPI, Go, Rust". The Hacker News, April 2026.

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

[7] Hive Pro. "UNC1069's Social Engineering Operations Focused on Crypto Sector". Hive Pro Threat Advisory, 2026.

[8] Dark Reading. "Axios Attack Shows Complex Social Engineering Is Industrialized". Dark Reading, April 2026.

[9] SC Media. "Axios maintainer's post mortem confirms social engineering by UNC1069". SC Media, April 2026.

[10] BleepingComputer. "Axios npm hack used fake Teams error fix to hijack maintainer account". BleepingComputer, April 2026.

[11] Elastic Security Labs. "Inside the Axios supply chain compromise – one RAT to rule them all". Elastic Security Labs, April 2026.

[12] BleepingComputer. "OpenAI rotates macOS certs after Axios attack hit code-signing workflow". BleepingComputer, April 2026.

[13] GNURub. "Detector script for the WAVESHAPER.V2 / axios npm supply chain attack". GitHub Gist, April 2026.

[14] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO". CSA Blog, February 6, 2025.

[15] Cloud Security Alliance. "Applying MAESTRO to Real-World Agentic AI Threat Models: From Framework to CI/CD Pipeline". CSA Blog, February 11, 2026.

[16] Cloud Security Alliance. "Securing Non-Human Identities in the Age of AI Agents". CSA, RSAC 2025.

[17] Cloud Security Alliance Labs. "TeamPCP Supply Chain Cascade: When Security Tools Become Attack Infrastructure". CSA Labs Research Note, April 2, 2026.

[18] Microsoft Security Blog. "Mitigating the Axios npm supply chain compromise". Microsoft Security Blog, April 1, 2026.

[19] Unit 42 / Palo Alto Networks. "Threat Brief: Widespread Impact of the Axios Supply Chain Attack". Unit 42, April 2026.

[20] SecurityWeek. "OpenAI Impacted by North Korea-Linked Axios Supply Chain Hack". SecurityWeek, April 2026.