The Unassessed Attack Surface: Channel-Based Provenance in HPC and Quantum-Computing Open-Source Software

Pengyin Shan
National Center for Supercomputing Applications
University of Illinois Urbana-Champaign
Urbana, USA
0009-0009-6309-380X

***Abstract*—Software trust is negotiated in communication channels such as the README, the registry listing, or the maintainer's account. AI agents now consume these channels directly, and attackers enter through them. Existing tools assess repositories, builds, and artifacts, but not this layer. We present a channel-provenance framework, to our knowledge the first for this layer, and measure 87 HPC and quantum-computing projects, hand-verifying a stratified 30-project panel. We find near-total provenance blindness. No missing signal requires new infrastructure, so the gap is adoption.**

***Index Terms*—software supply chain security, provenance, high-performance computing, quantum computing, AI agents**

I. Introduction

Every installation begins with an act of trust, negotiated in communication channels, such as the README that names the official Discord, the registry listing, or the maintainer's account. For humans, they were one input among many; for AI coding agents, the entire decision. 84% of developers use or plan to use AI tools [1] and package recommendation is intrinsic to code generation: of 2.23 million packages recommended by sixteen models, 19.7% were hallucinations and 43% of those recurred on every re-run [2].

Attackers found this layer first. The XZ Utils backdoor entered through a mailing list, where a sock-puppet pressure campaign installed the attacker as co-maintainer [3]. In 2025, the Shai-Hulud worms phished maintainer credentials, compromised over 1,000 npm packages across two waves, drawing a CISA alert [4]–[6]. In January 2026 the hallucinated package react-codeshift spread to 237 repositories from one commit of 47 unreviewed LLM-generated files, and agents kept downloading it after registration [7]. Each entry point was a channel that no widely-deployed tool assesses.

II. The Channel-Provenance Gap

Supply-chain security has marched toward the trust decision. OpenSSF Scorecard, SLSA, and Sigstore [8]–[10] assess repositories, builds, and artifacts, while Trusted Publishers [11] and pnpm's no-downgrade policy [12] carry identity provenance to publication. On the channels where trust is negotiated, it is hard to predict failure. After an attempted takeover of an OpenJS project, OpenSSF warned such attacks are difficult to detect programmatically [13]. We therefore call these channels' trust signals channel-based provenance: 18 signals asking who controls a project's voice (identity and access), which voice is authoritative (communication channels), how the channel corrects itself (knowledge infrastructure), and whether its assertions are machine-checkable (distribution metadata).

III. Method

We assessed 87 projects (44 HPC, 43 quantum computing) drawn from a popularity-thresholded frame of GitHub organizations and ecosystem topics. Nine automatically detectable signals were scored for all 87. The other nine require human judgment, so we verified them on 30 projects (˜35%) selected by seeded stratified sampling by domain and posture tier. Verification consulted the repository, releases, documentation site, and README-linked pages. MFA on channel administrators was unobservable for all 30 and is reported as unknown.

IV. Results

The 30-project panel is organizationally healthy: All 30 have multiple maintainers, 22 name official channels, and 94% of 87 publish stable releases. But the provenance layer is nearly absent. Table I maps each verified absence to a real attack. When a hijacked account published malicious Nx versions in August 2025, missing npm provenance was how the project confirmed they were not from its CI [14]. Phishing a maintainer can therefore attack a layer with no documented defenses and no assessment vocabulary.

V. Takeaways and Future Work

Table I's lone attestation came from slsa-github-generator [15], [16]. Setup is free and takes an afternoon, so the gap is adoption, not feasibility. For example, a maintainer can add official channels and a fake-reporting path in SECURITY.md, software stacks can surface these signals at install time, and facilities can request them in software plans.

Publishing a signal is only half of the chain. In a companion pre-registered study on six projects from this corpus (three HPC, three quantum computing), AI coding assistants opened

TABLE I
CHANNEL-PROVENANCE POSTURE, VERIFIED PANEL ($n = 30$)

Signal	Result	What its absence enables
Multiple maintainers	30/30	rules out solo-maintainer fragility
Official-channel statement	22/30	agents must guess which channel is real
Disclosure process	7 full / 3 basic / 20 none	attackers gain first-mover silence
Impersonation reporting	0/30	10 with SECURITY.md still lack fake account/channel/package instructions
Channel-admin succession	0/30	a departed maintainer leaves orphaned trust
Admin MFA	unknown, all 30	unverifiable externally; none states it
Signed releases	2/30	a hijacked account publishes indistinguishable artifacts [14]
SBOM	0/30	"are we affected?" requires manual archaeology
Provenance attestation	1/30	trust-downgrade blocking [12] protects almost no one

Auto tier ($n = 87$): 94% stable releases; 25% security policy; 20% archival DOI; 1/87 governance docs.

any provenance signal before installing in 9 of 1,920 registered trials (0.5%), and no trial ran a verification command, whether the signals were valid, forged, combined, or contradictory [17]. Supply and demand therefore fail together: maintainers rarely publish the signals, and assistants do not check them when they exist. Our next step re-measures both sides over time on an expanded panel. Two questions remain for the community: what counts as an official channel, and who attests to it?

ACKNOWLEDGMENT

Conducted under the author's 2025–2026 Trusted CI Fellowship. Drafted with writing assistance from Claude (Anthropic); all content and data were verified by the author.

REFERENCES

[1] S. E. Inc., "Ai — 2025 stack overflow developer survey," Stackoverflow.co, 2025. [Online]. Available: https://survey.stackoverflow.co/2025/ai/

[2] J. Spracklen, R. Wijewickrama, A. H. M. N. Sakib, A. Maiti, and B. Viswanath, "We have a package for you! a comprehensive analysis of package hallucinations by code generating {LLMs}," in *34th USENIX Security Symposium (USENIX Security 25)*. Seattle, WA: USENIX Association, 8 2025, pp. 3687–3706. [Online]. Available: https://www.usenix.org/conference/usenixsecurity25/presentation/spracklen

[3] CISA, "Reported supply chain compromise affecting xz utils data compression library, cve-2024-3094 — cisa," www.cisa.gov, 3 2024. [Online]. Available: https://www.cisa.gov/news-events/alerts/2024/03/29/reported-supply-chain-compromise-affecting-xz-utils-data-compression-library-cve-2024-3094

[4] Unit 42, ""shai-hulud" worm compromises npm ecosystem in supply chain attack (updated september 19)," Unit 42, 9 2025. [Online]. Available: https://unit42.paloaltonetworks.com/npm-supply-chain-attack/

[5] Cybersecurity & Infrastructure Security Agency, "Widespread supply chain compromise impacting npm ecosystem — cisa," Cybersecurity and Infrastructure Security Agency CISA, 9 2025. [Online]. Available: https://www.cisa.gov/news-events/alerts/2025/09/23/widespread-supply-chain-compromise-impacting-npm-ecosystem

[6] A. Kurmi, "ctrl/tinycolor and 40+ npm packages compromised - stepsecurity," Stepsecurity.io, 2025. [Online]. Available: https://www.stepsecurity.io/blog/ctrl-tinycolor-and-40-npm-packages-compromised

[7] C. Eriksen, "Agent skills are spreading hallucinated npx commands," Aikido.dev, 1 2026. [Online]. Available: https://www.aikido.dev/blog/agent-skills-spreading-hallucinated-npx-commands

[8] Open Source Security Foundation, "Openssf scorecard – open source security foundation," Openssf.org, 2020. [Online]. Available: https://openssf.org/projects/scorecard/

[9] The Linux Foundation, "Supply-chain levels for software artifacts," SLSA, 2026. [Online]. Available: https://slsa.dev/

[10] ——, "sigstore," sigstore, 2026. [Online]. Available: https://www.sigstore.dev

[11] S. M. L. (Python Software Foundation), "Trusted publishers for all package repositories," wg-securing-software-repos, 2024. [Online]. Available: https://repos.openssf.org/trusted-publishers-for-all-package-repositories.html

[12] R. Sobol, "How we're protecting our newsroom from npm supply chain attacks," Pnpm.io, 12 2025. [Online]. Available: https://pnpm.io/blog/2025/12/05/newsroom-npm-supply-chain-security

[13] OpenSSF, "Open source security (openssf) and openjs foundations issue alert for social engineering takeovers of open source projects – open source security foundation," Openssf.org, 4 2024. [Online]. Available: https://openssf.org/blog/2024/04/15/open-source-security-openssf-and-openjs-foundations-issue-alert-for-social-engineering-takeovers-of-open-source-projects/

[14] J. Strumpflohner, "S1ngularity - what happened, how we responded, what we learned," Nx.dev, 2025. [Online]. Available: https://nx.dev/blog/s1ngularity-postmortem

[15] kokkos, "Release kokkos v5.1.0 · kokkos/kokkos," GitHub, 3 2026. [Online]. Available: https://github.com/kokkos/kokkos/releases/tag/5.1.0

[16] slsa-framework, "Github - slsa-framework/slsa-github-generator: Language-agnostic slsa provenance generation for github actions," GitHub, 2 2025. [Online]. Available: https://github.com/slsa-framework/slsa-github-generator

[17] P. Shan, "Do AI coding assistants check before they install? a preregistered demand-side audit of trust signals in the research software supply chain," arXiv:2609.07754, 2026.