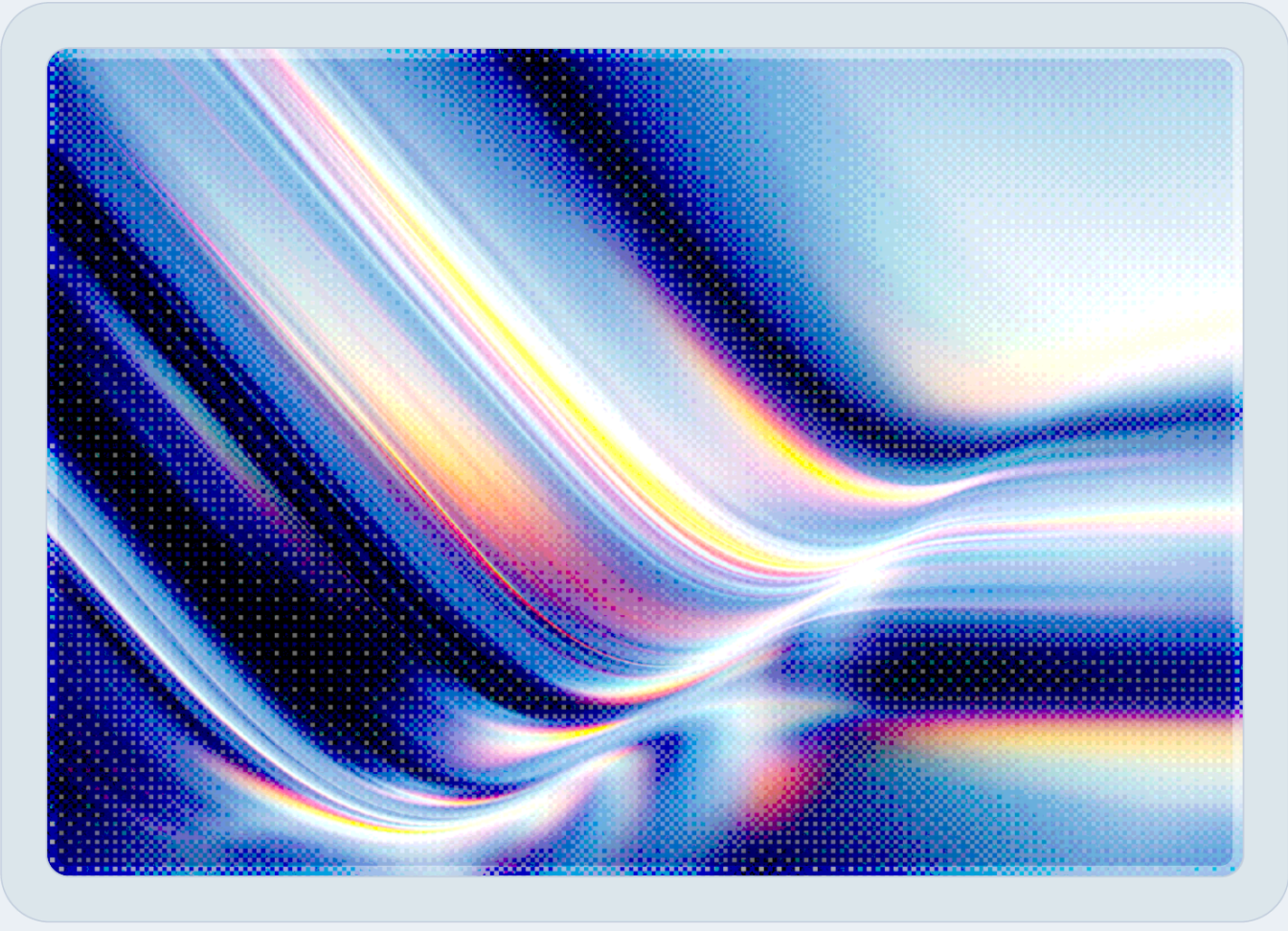
CSAI Foundation | Cloud Security Alliance

PromptMink: AI-Optimized DPRK Supply Chain Attack

How Famous Chollima Engineered Malicious npm Packages to Exploit AI Coding Agents

2026-04-30

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- The North Korean threat actor Famous Chollima (also tracked as Shifty Corsair) has deployed a sustained, AI-assisted npm supply chain campaign—codenamed PromptMink by ReversingLabs—that ran for at least seven months beginning in September 2025, publishing over 60 unique malicious packages across 300+ versions [1].
- PromptMink introduces a distinctive attack vector: packages are deliberately crafted using large language models (LLMs) to maximize their appeal to AI coding agents. Well-commented, polished code and convincing README files are engineered to pass the quick-glance scrutiny that characterizes AI-assisted dependency resolution—a technique ReversingLabs terms "LLM Optimization (LLMO) abuse" [1].
- On February 28, 2026, this approach succeeded in practice: a commit co-authored by Anthropic's Claude Opus model added the malicious package `@solana-launchpad/sdk` to an open-source autonomous crypto trading project, introducing a credential-stealing payload through a transitive dependency [1][4].
- The malware evolved rapidly over the campaign, progressing from JavaScript-based infostealers to Node.js Single Executable Applications (SEAs) exceeding 85 MB, and ultimately to compiled Rust payloads delivered as NAPI-RS Node.js add-ons—each iteration designed to evade the detection methods that neutralized its predecessor [1][2].
- PromptMink is part of a broader North Korean software supply chain offensive: Socket identified more than 1,700 malicious packages linked to this activity across npm, PyPI, Go, Rust, and Packagist since January 2025 [3].
- Organizations relying on AI coding assistants to generate or review dependency selections should treat those agents as an attack surface, not a verification layer.

Background

Famous Chollima is a financially motivated North Korean advanced persistent threat (APT) group operating under the broader Lazarus umbrella, with documented overlaps with BlueNoroff and the UNC1069 cluster tracked by Mandiant [2]. For more than three years, the group has run the Contagious Interview campaign, a persistent operation that lures software developers—particularly those working in

cryptocurrency and Web3—through fake job offers, trojanized coding assessments, and fraudulent technical interviews [6]. The campaign delivers a family of bespoke malware that has grown increasingly capable: BeaverTail (a JavaScript infostealer), InvisibleFerret (a Python-based RAT), and OtterCookie (a Node.js RAT now at version 5, with full keylogging, clipboard monitoring, browser credential theft, and remote mouse/keyboard control) [5][6].

PromptMink represents an evolution of this established playbook. Rather than relying solely on social engineering to trick individual developers into running malicious code, Famous Chollima appears to have identified a structural weakness in how AI coding agents can evaluate candidate packages—relying on surface-level signals such as naming conventions, documentation quality, and apparent code organization rather than deep static or behavioral analysis. By investing in LLM-generated package presentation, the group developed a mechanism for contaminating developer environments without requiring direct developer deception—one that demonstrably succeeded in at least one confirmed incident and shows evidence of iterative refinement across the campaign.

The timing suggests a deliberate strategic calculation. As AI pair-programming tools and autonomous coding agents expanded in use throughout 2025 [1][2], Famous Chollima appears to have identified and targeted the new attack surface they introduced. The group has a well-documented history of this kind of operational agility; after researchers exposed multiple phases of Contagious Interview, the actors regrouped with new infrastructure, new malware families, and new lure materials within weeks [6][7].

Security Analysis

The Two-Layer Architecture

PromptMink's central technical innovation is its layered package structure, designed to separate the entity that establishes trust from the entity that delivers harm. First-layer "bait" packages—such as `@solana-launchpad/sdk`, `@meme-sdk/trade`, `@validate-ethereum-address/core`, `@solmasterv3/solana-metadata-sdk`, `@pumpfun-ipfs/sdk`, and `@solana-ipfs/sdk`—contain no malicious code whatsoever [1]. They are published with well-structured documentation, realistic versioning histories, and package names that plausibly match legitimate tooling for Solana, Ethereum, and DeFi workflows. Their primary observable function is to declare second-layer payload packages as dependencies.

The actual malicious capability resides in packages such as `@validate-sdk/v2`, `@hash-validator/v2`, and `@slackgram/logger` [1]. When a developer—or an AI agent—installs any first-layer package, npm's dependency resolution automatically fetches and executes the payload without any additional user action. This separation appears designed to serve multiple operational purposes: detection of the first-layer package triggers no automated takedown since it contains no suspicious code; when second-layer packages are identified and removed from npm, they can be swiftly replaced while the first-layer packages—which are harder to remove once adopted—continue pointing developers toward new payloads.

LLM Optimization Abuse

ReversingLabs identified direct evidence of LLM involvement in PromptMink's development. The packages display unusually polished code structure, consistent inline comments, and professional README documentation—characteristics that ReversingLabs attributes to generative AI authorship and that they note were more effective at deceiving AI coding assistants than at fooling experienced human reviewers [1]. Forensic artifacts reinforced this attribution: researchers found residual LLM prompt text embedded in package metadata, including the instruction "also obfuscate the README a bit"—a fragment suggesting that threat actors were iterating on AI-generated content through prompt refinement [1].

The significance of this technique extends beyond PromptMink. Based on publicly available threat intelligence as of April 2026, this campaign represents among the first confirmed cases of a nation-state actor systematically optimizing malware presentation for the consumption patterns of AI coding tools. AI agents that suggest or auto-install dependencies may evaluate packages based on name relevance, documentation completeness, and apparent code quality—signals that Famous Chollima appears to have deliberately manufactured. The February 28, 2026 incident, in which Claude Opus co-authored a commit adding `@solana-launchpad/sdk` to the open-source `openpaw-graveyard` trading project, confirmed that this optimization produced real-world results [1][4].

Malware Capabilities and Evolution

ReversingLabs tracked PromptMink through four distinct development phases over approximately seven months [1]:

In the initial phase from September through January 2026, packages deployed obfuscated JavaScript that recursively scanned directories for `.env` and `.json` files—the standard locations for API keys, database credentials, and private keys in Node.js projects—and exfiltrated their contents to attacker-controlled infrastructure. Command-and-control communication ran through the domain

`validator[.]uno` via endpoints named `/api/validate/files`, `/api/validate/project-env`, and `/api/validate/system-info`, a naming scheme designed to blend with legitimate validation service traffic [1].

In February 2026, the campaign expanded to PyPI with the package `scraper-npm`, providing a Python implementation with equivalent credential-harvesting functionality and cross-platform coverage for developers who work across both ecosystems. That same month, attackers began bundling payloads as Node.js Single Executable Applications—binaries exceeding 85 MB that encapsulate the entire Node.js runtime alongside obfuscated JavaScript, substantially complicating static analysis [1].

By March 2026, the campaign transitioned to compiled Rust payloads distributed as NAPI-RS Node.js add-ons. These native modules are significantly more difficult to reverse-engineer than JavaScript, load directly into the Node.js process space, and can access system resources without the subprocess overhead that behavioral detection tools monitor. The Rust variant added capabilities beyond credential theft: it exfiltrates entire project source trees, installs attacker SSH keys for persistent remote access on both Linux and Windows systems, and collects detailed system metadata including OS version, IP addresses, and usernames [1][2].

Infrastructure and Scale

Across the seven-month campaign, ReversingLabs identified 60 unique malicious packages published across more than 300 versions, supported by more than 20 command-and-control domains and IP addresses [1]. Key infrastructure included `validator[.]uno`, `ipfs-url-validator.vercel.app`, and `api.mywalletsss.store`, alongside IP addresses including `45[.]61[.]161[.]146`, `45[.]8[.]22[.]144`, and `45[.]8[.]22[.]52` [1]. The use of Vercel-hosted infrastructure for early C2 stages—a pattern consistent with prior Famous Chollima operations—allowed the group to leverage trusted CDN infrastructure to reduce the likelihood of network-level blocking. Separately, researcher analysis of npm publisher account registrations revealed that threat actors inadvertently exposed their own IP addresses through publicly accessible disposable email inbox metadata, providing additional attribution intelligence on the campaign's operators [9].

The Parallel Graphalgo Campaign

PromptMink did not operate in isolation. Concurrently, Famous Chollima ran the Graphalgo campaign, which targeted developers through fraudulent companies with manufactured legitimacy. The group registered Blocmerce as a Florida LLC in August 2025, established Veltrix Capital and Bridgers Finance with credible web presences, and recruited developers through these entities for coding tests that

delivered malicious npm packages including `graph-dynamic`, `graphbase-js`, and `graphlib-js` [2]. The preparation required to establish corporate registrations, build out fake business websites, and create functional interview processes demonstrates the operational investment Famous Chollima is willing to make to access developer environments at scale.

Broader Supply Chain Footprint

Socket's analysis of the full Famous Chollima–linked supply chain activity identified more than 1,700 malicious packages distributed across npm, PyPI, Go, Rust, and Packagist since January 2025 [3]. The cross-ecosystem coordination—with parallel packages deployed across five registries simultaneously, including a Rust package (`logtrace`) that embedded malicious code inside a legitimate-appearing `Logger::trace()` method—indicates a coordinated, multi-registry campaign sustained across more than 15 months and five package ecosystems simultaneously. A parallel operation documented by Silent Push identified three additional DPRK front companies—BlockNovas LLC, Angeloper Agency, and SoftGlide LLC—using comparable developer recruitment tactics [8]. Between February 6 and April 7, 2026, the Security Alliance SEAL team blocked 164 domains tied to UNC1069, including infrastructure impersonating Microsoft Teams and Zoom used for developer recruitment [2][10].

Recommendations

Immediate Actions

Organizations using AI coding assistants or agentic development tools should audit recent dependency additions for packages matching the known PromptMink first-layer indicators: `@solana-launchpad/sdk`, `@meme-sdk/trade`, `@validate-ethereum-address/core`, `@solmasterv3/solana-metadata-sdk`, `@pumpfun-ipfs/sdk`, `@solana-ipfs/sdk`, and `@hash-validator/v2` [1]. Any environment that installed these packages should be treated as potentially compromised, and `.env` files, JSON-stored credentials, SSH authorized_keys entries, and crypto wallet files should be rotated or revoked immediately. Development machines that ran these packages should be examined for unauthorized SSH keys in `~/.ssh/authorized_keys` and unexpected outbound connections to `validator[.]uno` or the C2 IP ranges listed above.

Security teams should also block the known C2 infrastructure at the network perimeter: `validator[.]uno`, `api.mywalletsss.store`, and the IP addresses `45[.]61[.]161[.]146`, `45[.]8[.]22[.]144`, and `45[.]8[.]22[.]52` [1]. Given the campaign's pattern of rapidly cycling domains and payloads, these indicators should be treated as a representative sample rather than a complete blocklist.

Short-Term Mitigations

The architectural response to PromptMink requires treating AI coding agents as dependency-selection actors that require the same scrutiny applied to human developers. Organizations should implement or tighten npm package allowlisting policies so that AI agents can only add packages from a pre-approved registry or require human review before any new transitive dependency is introduced into a production codebase. Lock file hygiene—reviewing changes to `package-lock.json` for new transitive entries as part of code review—provides a detection opportunity for the specific mechanism PromptMink exploited: the malicious payload installed via `@validate-sdk/v2` was not a direct project dependency, only a transitive one [1][4].

Software composition analysis (SCA) tooling should be integrated at the CI/CD level to flag any dependency added by an AI-assisted workflow before the package is committed. This creates a chokepoint that re-introduces human review for the specific scenario PromptMink exploited—AI agent selection of unvetted packages. Development teams building or operating autonomous coding agents should configure those agents with explicit constraints on dependency management: agent-generated code that introduces new packages should be treated as a distinct review category requiring human confirmation.

Developer awareness should explicitly cover the LLMO abuse vector. The traditional indicators of a trustworthy package—well-organized code, professional documentation, consistent commenting—can now be artificially generated by the same tooling developers use every day. The presence of these signals can no longer be treated as reliable evidence of legitimacy in a threat environment where adversaries are using LLMs to manufacture convincing-looking malicious packages at scale.

Strategic Considerations

The PromptMink campaign may represent a meaningful inflection point in supply chain threat strategy—the first documented case where adversarial use of AI generation was demonstrably optimized for AI tool consumption. Famous Chollima has demonstrated that nation-state actors can adapt attack techniques faster than the ecosystem's trust infrastructure evolves. While npm has implemented code signing, automated malware scanning, and publisher verification controls, its trust signals were not designed to

surface packages that are polished, well-documented, and technically functional except for a single malicious payload buried in a transitive dependency. The security community needs to develop new trust signals appropriate for an environment where adversaries use AI to generate convincing-looking malicious packages at scale.

At the organizational level, enterprises should evaluate their AI coding agent governance policies to determine whether those tools have unconstrained access to install or recommend packages from public registries. The question is not whether to use AI coding assistants but whether the processes governing their interactions with the software supply chain are commensurate with the threats now documented against them.

CSA Resource Alignment

CSA's MAESTRO framework for agentic AI threat modeling directly addresses the attack surface exploited by PromptMink. MAESTRO's Layer 3 (Agent Frameworks) and Layer 7 (Ecosystem and Infrastructure) map to the conditions Famous Chollima exploited: an AI coding agent with unsupervised access to public package registries, operating in an ecosystem where supply chain trust signals have not kept pace with adversarial adaptation. MAESTRO's threat modeling guidance addresses the conditions under which autonomous tool-use by agents—including package installation—warrants explicit human-in-the-loop controls for high-consequence actions, consistent with its Layer 3 and Layer 7 threat mappings.

CSA's AI Controls Matrix (AICM) provides supply chain security controls across Application Provider, Orchestrated Service Provider, and AI Customer roles that are directly applicable here. AICM supply chain domain controls address third-party component vetting, dependency integrity verification, and software bill of materials (SBOM) requirements—all of which would have provided detection or prevention opportunities across the PromptMink kill chain. Organizations building or deploying AI coding agents should map their supply chain controls against AICM to identify gaps analogous to those Famous Chollima exploited.

The CSA Cloud Controls Matrix (CCM) supply chain management controls (STA domain) apply to the organizational governance question: how development environments running AI agents are provisioned, what network access they have, and what approval workflows govern package installation in those environments. The STAR assessment program can be used to evaluate the supply chain security posture of AI coding tool vendors themselves, given that the trustworthiness of agent-generated dependency selections is now a material security question.

More broadly, PromptMink underscores the Zero Trust principle that no tool or process—including AI coding assistants—should have implicit trust to modify the software supply chain without verification. Trust must be continuously validated, and the package selection behavior of AI agents should be treated as an action requiring the same authorization controls applied to human developer commits.

References

[1] ReversingLabs. "Claude adds PromptMink malicious dependency to crypto agent." ReversingLabs Blog, April 2026.

[2] The Hacker News. "New Wave of DPRK Attacks Uses AI-Inserted npm Malware, Fake Firms, and RATs." The Hacker News, April 2026.

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

[4] Cybersecurity News. "Claude-Generated Commit Adds PromptMink Malware to Crypto Trading Agent." Cybersecurity News, April 2026.

[5] Infosecurity Magazine. "Malicious npm Dependency Linked to AI Assisted Commit Targets Crypto Wallets." Infosecurity Magazine, April 2026.

[6] Microsoft Security Blog. "Contagious Interview: Malware delivered through fake developer job interviews." Microsoft, March 11, 2026.

[7] Threat Intelligence Report. "FAMOUS CHOLLIMA: DPRK employment fraud and developer-lure intrusion set." TIR, February 28, 2026.

[8] Silent Push. "Contagious Interview (DPRK) Launches a New Campaign Creating Three Front Companies to Deliver a Trio of Malware: BeaverTail, InvisibleFerret, and OtterCookie." Silent Push Blog, 2026.

[9] Threat Intelligence Report. "DPRK FAMOUS CHOLLIMA OPSEC failure exposes npm publisher IPs through public disposable inboxes." TIR, February 28, 2026.

[10] Security Alliance. "SEAL Radar Advisory: DPRK UNC1069 Fake Microsoft Teams and Zoom Calls." Security Alliance SEAL, 2026.