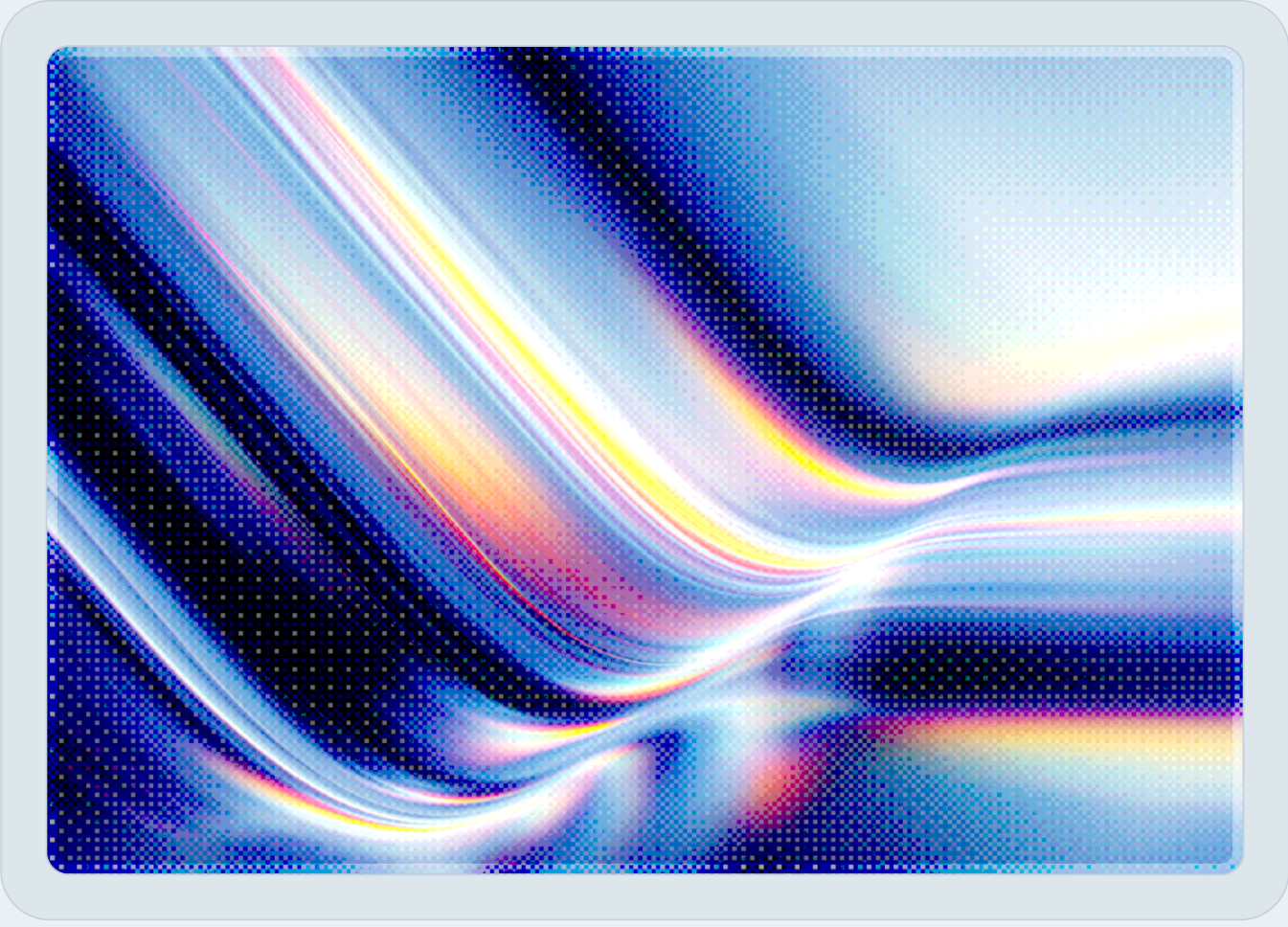
CSAI Foundation | Cloud Security Alliance

DPRK PromptMink: Nation-State npm Malware Targets AI Coding Agents

LLM Optimization Abuse Weaponizes AI Package Suggestions Against Developer Supply Chains

2026-05-01

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

The following findings are critical for security teams managing AI-assisted development environments.

- North Korean state-sponsored group Famous Chollima is conducting the PromptMink campaign, which deliberately engineers malicious npm packages to be selected and installed by AI coding agents rather than human developers.
- A February 28, 2026 commit to the open-source crypto trading project `openpaw-graveyard`—co-authored by Claude Opus—introduced a malicious transitive dependency (`@validate-sdk/v2`) that exfiltrated wallet credentials and established SSH backdoors on compromised hosts.
- The technique, termed **LLM Optimization (LLMO) abuse** by ReversingLabs, exploits the AI package-recommendation pipeline: attackers craft detailed, plausible-looking package documentation specifically to satisfy the context window and confidence heuristics of coding agents.
- Across a seven-month operation spanning September 2025 through April 2026, researchers tracked more than 60 packages and over 300 versions [1], with a parallel campaign distributing more than 1,700 malicious packages across npm, PyPI, Go, and Rust ecosystems [2].
- The malware has evolved from obfuscated JavaScript stealers to pre-compiled Rust payloads delivered via NAPI-RS, indicating deliberate engineering to defeat static analysis and shrink payload signatures.
- Organizations using AI coding agents—including Cursor, VS Code with Copilot, Windsurf, Claude Code, and similar tools—that lack dependency vetting workflows are in scope for this attack class, with degree of exposure varying by tool configuration and agentic workflow permissions.

Background

Famous Chollima and the Contagious Interview Lineage

Famous Chollima is a North Korean state-sponsored intrusion set active since at least 2018 [3], widely associated in public threat intelligence reporting with the broader Lazarus Group. It is the threat actor behind the long-running Contagious Interview campaign, in which North Korean operatives impersonate recruiters on LinkedIn, GitHub, and job boards to deliver coding assessments that embed malware through project dependencies [3]. The group operates an extensive fraudulent IT worker infrastructure, using AI-generated synthetic identities, deepfake video capabilities, and front companies to place operatives inside Western organizations [4].

The primary financial objective of Contagious Interview operations is cryptocurrency theft, which funds DPRK weapons programs under ongoing international sanctions. Secondary objectives observed in 2025–2026 include intellectual property exfiltration—particularly source code from Web3 and AI projects—and the establishment of persistent remote access for future exploitation [1].

Emergence of PromptMink

PromptMink is the campaign designation assigned by ReversingLabs to a specific strand of Famous Chollima activity that began in September 2025 and represents a deliberate tactical evolution: rather than relying solely on social engineering to convince human developers to install malicious packages, the threat actors are engineering packages specifically to be adopted by AI coding agents operating autonomously within development pipelines [1].

The campaign's name reflects its defining characteristic—the manipulation of LLM prompt-evaluation behaviors to achieve malicious software installation through an AI intermediary. The pivot to AI agents as the primary target of influence—rather than the human developer—marks a significant tactical evolution that some researchers have characterized as a qualitatively new threat class.

Security Analysis

The LLM Optimization Abuse Technique

LLMO abuse exploits a structural property of how AI coding agents evaluate and recommend packages. When an agent is asked to add a feature, resolve a dependency, or refactor code, it evaluates candidate packages not by running them but by processing their documentation, README content, package metadata, download counts, and version history. Famous Chollima has inverted this evaluation pipeline into an attack surface.

The group publishes malicious packages accompanied by detailed, professionally written documentation—complete with usage examples, API references, and badges—that are crafted to appear credible to the context window of a large language model [1]. Unlike typosquatting or dependency confusion attacks, which primarily deceive humans or automated version resolvers, LLMO abuse specifically targets the LLM's documentation comprehension. It follows that packages crafted this way may be more likely to pass AI agent screening than manual developer review—AI agents lack the out-of-band signals available to human developers, such as community reputation, GitHub star history, and institutional knowledge about known-good packages.

The technique is distinct from but related to the AI package hallucination attack class, in which attackers register package names that LLMs are known to hallucinate. LLMO abuse does not rely on hallucination; instead, it plants crafted legitimate-looking packages in the registry and constructs their metadata to score well on AI relevance heuristics. The result is a supply chain attack that specifically accelerates in environments where AI agents handle dependency management autonomously with minimal human review gates.

The February 28 Incident: Anatomy of the Attack

On February 28, 2026, a commit was submitted to `openpaw-graveyard`, an open-source autonomous Solana trading agent project. The commit added `@solana-launchpad/sdk` as a dependency. The package appeared to be a legitimate Solana ecosystem SDK and was accompanied by plausible documentation. Critically, the commit was co-authored by Claude Opus, indicating that an AI coding agent—operating under developer direction or autonomously within a CI workflow—selected and incorporated the dependency [1][4].

The attack employed a two-layer dependency structure that is characteristic of Famous Chollima operations. The first-layer package, `@solana-launchpad/sdk`, contained no malicious code on its own, serving as a plausible decoy that would pass surface-level inspection. The actual payload was embedded in a second-layer transitive dependency, `@validate-sdk/v2`, which presented itself as a standard data validation utility [1]. This staging reduces the probability that any single package triggers automated scanning alerts, and allows rapid replacement of the second-layer payload if it is detected and removed from the registry.

Once installed, `@validate-sdk/v2` recursively scanned the host environment for `.env` and `.json` configuration files, harvested wallet credentials, API keys, and environment secrets, then staged and exfiltrated the material to an attacker-controlled Vercel endpoint—specifically `ipfs-url-validator.vercel.app`, a Vercel subdomain that Famous Chollima has repeatedly abused across multiple campaigns [1]. In addition to credential theft, the malware appended the attacker's public SSH key to the victim's `~/.ssh/authorized_keys` file, establishing persistent remote access to compromised hosts [1][6]. Later versions of the malware extended this behavior to compress and exfiltrate entire project source directories, consistent with intellectual property theft as a secondary objective.

Malware Evolution: From JavaScript to Rust

ReversingLabs documented a sequential evolution of the PromptMink payload over the campaign's seven-month span [1]:

Phase	Timeframe	Delivery Mechanism	Payload
Phase 1	Sep 2025 – Jan 2026	Obfuscated JavaScript	Environment file stealer, Vercel exfil
Phase 2	Feb 23, 2026	PyPI package	Python infostealer with equivalent logic
Phase 3	Feb 28, 2026	Node.js SEA (single executable application)	~85 MB bundled executable stealer
Phase 4	Mar 17, 2026	NAPI-RS compiled Rust add-on	~5 KB pre-compiled native module

The shift to NAPI-RS—a framework for building native Node.js modules in Rust—represents a deliberate engineering decision to reduce payload size and evade signature-based detection. The earlier Node.js SEA approach bloated payloads from approximately 5.1 KB to 85 MB, a size anomaly likely to trigger heuristic alerts. The compiled Rust modules restore a small footprint while gaining the anti-analysis properties of compiled native code: no readable JavaScript, resistance to string-based grep, and no requirement for a runtime interpreter [1].

The McpInject Vector and AI Agent Persistence

Beyond the dependency injection mechanism, security researchers identified a component designated `McpInject` in later campaign iterations that specifically targets AI coding assistants by deploying a malicious Model Context Protocol (MCP) server and injecting it into the tool configurations of affected development environments [1]. MCP servers have broad access to the context provided to AI agents—including file system state, repository contents, and active tool schemas—making a compromised MCP server a high-value persistence and exfiltration point.

The significance of McpInject is that it transforms a one-time package compromise into an ongoing agent manipulation capability. Once a malicious MCP server is registered in a developer's AI tool configuration, it can intercept agent tool calls, inject instructions into the agent's context, and silently exfiltrate material from every subsequent AI-assisted session. The attack surface encompasses AI coding assistants that support MCP extensions, including Cursor, VS Code, Windsurf, and Claude Code; the degree of susceptibility in each case depends on the tool's default MCP server trust configuration and how broadly permissions are granted to locally registered servers [5].

Broader Campaign Scale and the 1,700-Package Operation

PromptMink is one strand of a substantially larger Famous Chollima supply chain operation. A parallel campaign tracked through April 2026 distributed more than 1,700 malicious packages across npm, PyPI, Go modules, and the Rust crate registry, with Socket Security researchers identifying the scale and cross-ecosystem distribution of the activity [2]. This scale indicates that Famous Chollima is operating an industrialized package factory rather than conducting targeted manual attacks; the volume of payload variants suggests a significant degree of automated generation, though the specific tooling behind this production capacity has not been independently confirmed.

The use of front companies and fraudulent GitHub organizations as apparent package publishers further complicates detection. These entities maintain fabricated employee profiles across LinkedIn, GitHub, GitLab, and other platforms, lending the organizations a surface appearance of legitimacy that AI agent documentation-evaluation heuristics cannot reliably distinguish from authentic maintainers [4].

Attribution Context

ReversingLabs' attribution of PromptMink to Famous Chollima rests on infrastructure overlaps—notably the repeated abuse of Vercel subdomains for exfiltration, a pattern documented across prior Famous Chollima campaigns—and code-level continuity with earlier Contagious Interview payloads [1][3]. Some researchers note a partial overlap with the UNC1069 cluster, which Mandiant associates with BlueNoroff, Sapphire Sleet, and Stardust Chollima, suggesting coordination or tooling sharing among adjacent DPRK cyber units [2]. Attribution in this campaign is complicated by the use of AI-generated code, which can obscure stylistic fingerprinting, but the infrastructure and operational patterns remain consistent with prior Famous Chollima activity.

Recommendations

Immediate Actions

Organizations using AI coding agents in development workflows should treat any AI-suggested package addition as untrusted input until verified through an independent vetting gate. This is not a hypothetical risk posture—the PromptMink incident demonstrates that a production AI coding agent selected and committed a malicious transitive dependency without human review catching it before it was introduced to a repository used by real developers.

Concretely, teams should audit their CI/CD pipelines to identify any step where an AI agent has write access to `package.json`, `requirements.txt`, `go.mod`, or equivalent dependency manifests, and ensure that a human approval gate or automated policy check intercepts those changes before merge. For agentic workflows that operate with `--dangerously-skip-permissions` or equivalent unrestricted modes, the risk surface is substantially elevated and warrants immediate policy review.

Development environments using MCP servers—including Claude Code, Cursor, and VS Code with MCP extensions—should be audited for unexpected MCP server registrations. The presence of an unfamiliar MCP server in a tool configuration file should be treated as a potential compromise indicator.

Short-Term Mitigations

Teams should implement a package vetting workflow that applies independently of how a dependency is introduced—whether by a human developer or an AI coding agent. Effective controls include:

- Pinning dependencies to specific verified hashes rather than semver ranges, which substantially mitigates the second-layer transitive dependency substitution technique used in PromptMink; this control is most effective when the lockfile is generated in a clean environment and reviewed before commit.
- Configuring npm, pip, and equivalent package managers to require lockfile integrity checks and to flag packages that are newly published (for example, within the last 30–90 days, depending on organizational risk tolerance) or with fewer than a threshold of verified downloads.
- Deploying software composition analysis (SCA) tools—such as those integrated into GitHub Advanced Security, Snyk, or Socket—on all pull requests, with rules that trigger on new transitive dependency additions regardless of the PR author (human or AI agent).
- Monitoring outbound network connections from development workstations and CI runners for unexpected connections to Vercel subdomains and other cloud function hosting services, which Famous Chollima consistently uses for exfiltration.

For AI coding agent deployments specifically, organizations should define a package allowlist or trusted namespace policy and configure the agent's system prompt or tool policy to require human confirmation before adding packages outside that set. AI agents should not have unreviewed write access to production dependency manifests.

Strategic Considerations

PromptMink represents the materialization of a threat model the security community has been developing in recent years—that adversaries would eventually optimize attacks for AI evaluation rather than human evaluation. The implication is that existing security awareness training, code review culture, and human-facing indicators of compromise do not transfer to agentic development environments without deliberate adaptation.

Organizations adopting AI-assisted development at scale should establish explicit AI agent security policies that address autonomous dependency management, MCP server trust, and the boundary between AI agent autonomy and mandatory human review. These policies should be treated as a subset of the organization's broader supply chain security program, not as a distinct and separate concern. As AI coding agents gain more autonomy and broader tool access, the blast radius of a single compromised package recommendation expands correspondingly.

The broader DPRK threat picture suggests that this attack class will persist and intensify. Famous Chollima's industrialized package factory—producing over 1,700 malicious packages across four ecosystems within months—demonstrates institutional investment in this attack vector. The group's

consistent abuse of trusted infrastructure, including Vercel subdomains and fabricated GitHub organizations, means that the cost of producing each malicious package continues to fall while the difficulty of detecting individual instances does not decrease at the same rate.

CSA Resource Alignment

The PromptMink campaign maps to several active CSA frameworks and guidance areas.

The CSA **MAESTRO** (Multi-Agent Environment and System Threat and Risk Oracle) framework addresses the threat model for agentic AI deployments. PromptMink's McpInject component directly instantiates MAESTRO's concerns around tool poisoning and agent context manipulation—specifically the scenario in which a compromised tool or MCP server corrupts the agent's execution environment from within rather than attacking it externally. MAESTRO's Layer 6 (agent-to-tool interactions) and Layer 7 (external integration points) are both engaged by this attack class.

The CSA **AI Controls Matrix (AICM)** v1.0 contains supply chain security controls under its AI Supply Chain domain that apply directly to dependency vetting in AI-assisted development. AICM controls addressing model and component provenance verification are relevant not only to the AI models themselves but to the software packages that AI agents introduce into the build. Organizations should review AICM's Application Provider (AP) controls for guidance on establishing dependency governance in AI-augmented development workflows.

The CSA **STAR for AI** program provides an assessment framework for evaluating AI system trustworthiness. Organizations using third-party AI coding agents should request or review STAR for AI assessments for those products that address the agent's dependency management behaviors, MCP server trust model, and package-selection audit trail.

CSA's **Zero Trust guidance** is applicable to MCP server trust architecture: the principle of least privilege and never-implicit-trust should govern which MCP servers an AI coding agent is permitted to invoke, and all MCP server registrations should require explicit authorization rather than being accepted from the local tool configuration file automatically.

References

[1] ReversingLabs. "Claude Adds PromptMink Malicious Dependency to Crypto Agent". ReversingLabs Blog, April 29, 2026.

[2] The Hacker News. "N. Korean Hackers Spread 1,700 Malicious Packages Across npm, PyPI, Go, Rust". The Hacker News, April 8, 2026.

[3] Threat Intelligence Report. "Famous Chollima: DPRK Employment Fraud and Developer-Lure Intrusion Set". TIR, February 28, 2026.

[4] The Hacker News. "New Wave of DPRK Attacks Uses AI-Inserted npm Malware, Fake Firms, and RATs". The Hacker News, April 29, 2026.

[5] Infosecurity Magazine. "Malicious npm Dependency Linked to AI-Assisted Commit Targets Crypto Wallets". Infosecurity Magazine, April 29, 2026.

[6] CyberSecurityNews. "Claude-Generated Commit Adds PromptMink Malware to Crypto Trading Agent". CyberSecurityNews, April 30, 2026.