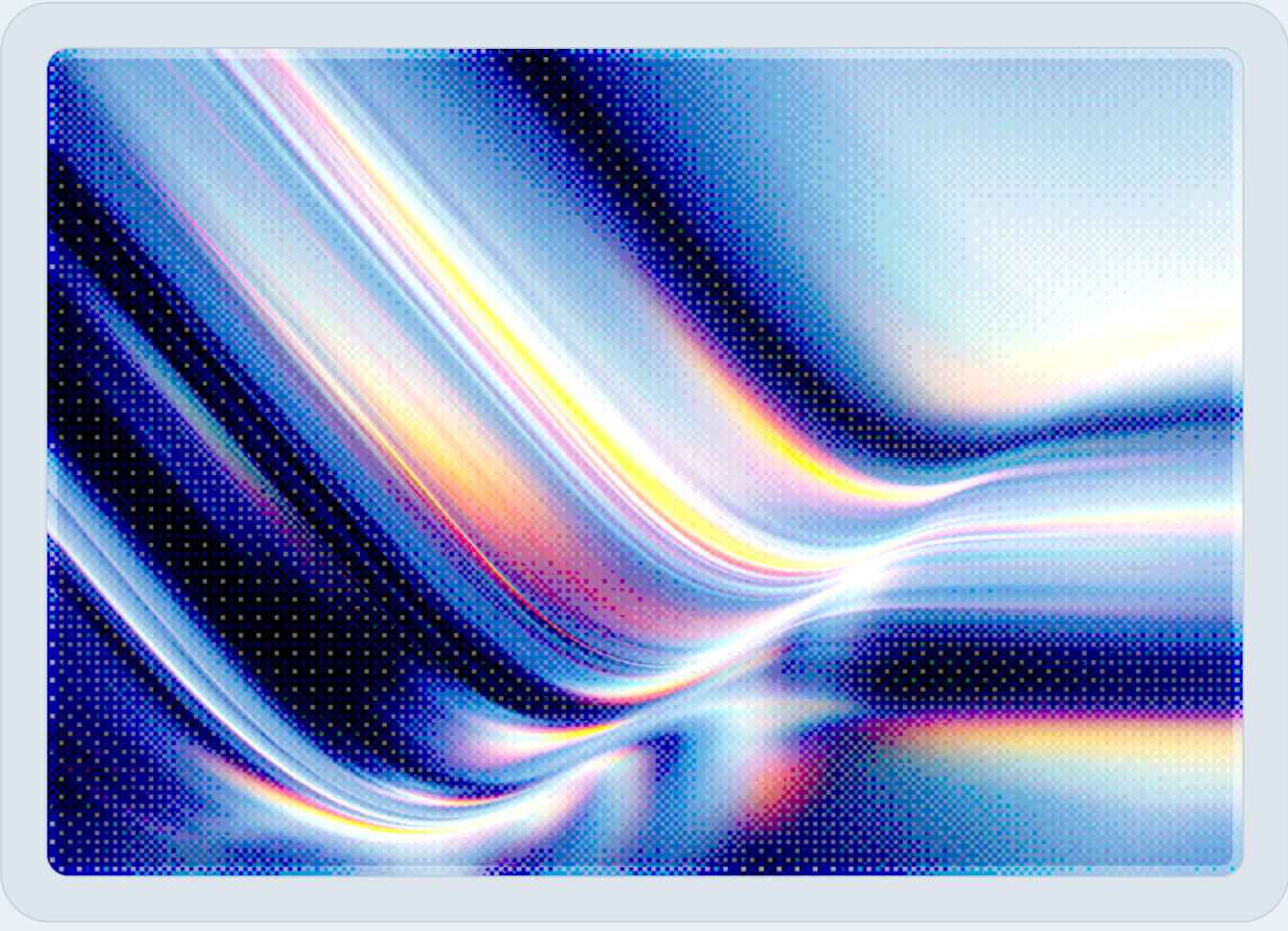
CSAI Foundation | Cloud Security Alliance

Slopsquatted npm Packages Deliver RAT and Infostealer at Scale

The WEL1DROPPER / Flooding Dropper Campaign and Its Implications for Package Hygiene

2026-08-10

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- Between roughly August 3 and August 5, 2026, a threat actor published nearly 800 malicious packages to the npm registry — a figure that researchers report has since grown beyond 1,000 — in a campaign tracked as "WEL1DROPPER" by the researcher group OpenSourceMalware and as "Flooding Dropper" (advisory sonatype-2026-005660) by Sonatype [1][2][3].
- Rather than impersonating specific legitimate libraries through classic typosquatting, the packages relied on randomly generated, plausible-sounding names — a pattern researchers describe as "AI slopsquatting," echoing the AI-hallucinated package names that large language models sometimes invent when asked to recommend dependencies [1][2].
- The packages avoided npm's `preinstall` / `postinstall` lifecycle hooks entirely, instead instructing developers via README files to load them with the built-in `require()` function — a technique that has the effect of evading tooling that specifically monitors install-time script execution [1].
- Once loaded, a downloader component fetches a cross-platform payload from Cloudflare Workers infrastructure, with a DNS-TXT-record fallback channel, and delivers operating-system-specific persistence, credential theft, and remote access capabilities to Windows, macOS, and Linux hosts alike [1][2][3].
- Infrastructure naming, targeting of Russian financial institutions, and tradecraft overlap with the earlier "Moika" npm campaign point to a financially motivated actor with a Russian-language footprint, though attribution remains circumstantial [1][3][4].

Background

The npm registry has seen a series of supply chain attacks throughout 2026, and this campaign extends a pattern CSA has documented in prior research on cross-platform remote-access-trojan delivery through compromised or abused npm packages, including the Mastra scope takeover discussed later in this note [6]. What distinguishes the current wave is scale paired with a new evasion strategy. On August 5, 2026, the open source threat intelligence group OpenSourceMalware, led by researcher Paul McCarty, reported that a cluster of packages published over a roughly 48-hour window numbered in the hundreds and shared a consistent malware delivery chain [1][2]. The Hacker News subsequently reported the count at nearly 800 packages, and Sonatype's own tracking — published under advisory sonatype-2026-

005660 and the label "Flooding Dropper" — put the confirmed figure at 846 malicious components as of its analysis (Sonatype's own headline rounds this to 850), with OpenSourceMalware noting the campaign had grown past 1,000 packages by the time later reporting went to press [1][3].

One of the campaign's most significant departures from prior npm attacks is its naming strategy. Traditional typosquatting mimics a specific, well-known package name with a subtle misspelling (for example, substituting "electorn" for "electron") to catch a developer's typo. This campaign instead generated large volumes of plausible but entirely fictitious package names by interpolating common but generic terms — Sonatype's analysis cites recurring fragments such as "bigops" and "bnpl" combined with other words, producing names like `bigops-api` and `dolyame-boxy-desktop-bnpl-card-gallery`, often published under a recurring version pattern beginning around 35.x.y [3]. Researchers have labeled this approach "AI slopsquatting" because the resulting names resemble the kind of plausible-but-nonexistent package names that generative AI coding assistants are known to hallucinate when asked to suggest a dependency, and because the sheer volume and generic-sounding structure of the names suggests automated, likely AI-assisted, name generation rather than manual crafting [1][2][4]. Whether a developer discovers one of these packages through a hallucinated AI coding-assistant suggestion, a search-engine result, or simple chance, the packages are built in a way that is consistent with an attempt to look unremarkable enough to survive a cursory glance.

The campaign's execution technique also differs from the install-time script hooks that have defined most npm malware to date. Security tooling built over the past two years has increasingly focused on monitoring or blocking `preinstall` and `postinstall` scripts, since those lifecycle hooks are the most common vector for automatic code execution during `npm install`. The packages in this campaign instead ship with a README that instructs the developer to import the package with the standard `require()` call, at which point the malicious logic executes as ordinary module code — a channel that lifecycle-hook-focused scanners and `--ignore-scripts` flags do not cover [1]. A structurally similar bypass surfaced one month earlier in the AsyncAPI npm compromise, where a trojanized package executed its payload at import time — Socket.dev's analysis of that incident noted plainly that "the malicious code does not need `preinstall`, `postinstall`, or any package lifecycle script. It runs when the infected module is loaded by Node.js," specifically defeating organizations relying on `--ignore-scripts` as a control [5]. The recurrence of import-time execution across unrelated campaigns within a matter of weeks suggests threat actors have identified lifecycle-hook monitoring as a control gap that is now being deliberately targeted rather than incidentally bypassed.

Security Analysis

The malware chain begins with a downloader OpenSourceMalware has named WEL1DROPPER, after the fallback command-and-control domain `wel1.ru`. Upon execution, the dropper fingerprints the host operating system and processor architecture and requests a matching payload from one of three Cloudflare Workers subdomains — `oob-worker.cf103-070.workers[.]dev`, `oob-worker.cf102-baf.workers[.]dev`, and `oob-worker.cf99-9b3.workers[.]dev` [1]. Using Cloudflare's own infrastructure as a delivery front typically lets malware blend its command-and-control traffic into ordinary HTTPS requests to a widely trusted cloud provider, complicating network-based blocking that relies on domain reputation. If those primary channels are unreachable, the dropper falls back to reconstructing its payload from DNS TXT records hosted under platform-specific subdomains of `wel1[.]ru`: `sdk.dl.wel1[.]ru` for Linux x64 hosts, `ext.dl.wel1[.]ru` for Linux ARM64, `pkg.dl.wel1[.]ru` for macOS, and `net.dl.wel1[.]ru` for Windows [1]. This kind of DNS-based fallback is a resilience technique that can let an operator keep a campaign alive even after its primary web infrastructure is taken down, since DNS resolution is far harder for defenders to block wholesale without breaking legitimate traffic.

The final payload's behavior diverges meaningfully by platform. CSA assesses that this platform-specific divergence reflects greater operational investment than is typical of credential-theft-only npm malware. On Windows, the malware patches Event Tracing for Windows (ETW) and the Antimalware Scan Interface (AMSI) in memory to blind endpoint security tooling before establishing persistence through a Registry Run key and a scheduled task, then retrieves an encrypted secondary payload disguised under a filename such as `update_win.exe` or, in some observed samples, names mimicking legitimate .NET diagnostic tooling [1][2]. On macOS, the malware performs anti-debugging and sandbox-detection checks before installing a LaunchAgent for persistence and pulling a beacon binary; researchers also observed references to Russian financial services including TCSBank and CloudPayments embedded in the sample, which may indicate either targeting of those institutions' customers or developers, or the reuse of fraud-oriented tooling built for a different purpose [1]. On Linux, the payload is a UPX-packed ELF binary that ultimately deploys Sliver, an open source command-and-control framework originally built for legitimate red-team use but now a common building block for criminal and state-linked operators because it is well-documented, actively maintained, and freely available [1].

To further complicate detection during manual code review, the packages bundle a file named `lib/telemetry.js` that presents itself as ordinary application telemetry or usage-profiling code, while in fact housing the downloader logic itself [1]. A developer or automated reviewer skimming the

package's file list for anything alarming is more likely to dismiss a file named "telemetry" as benign instrumentation than to flag it as the payload delivery mechanism — a piece of social engineering aimed at the reviewer rather than the end user, exploiting the assumption that a "telemetry" filename is benign.

Attribution signals point toward a financially motivated, Russian-language-affiliated actor, though CSA treats this as an assessment rather than a confirmed finding. The `wel1.ru` domain registration, the Russian financial institution references embedded in the macOS payload, and tradecraft overlap with an earlier campaign known as "Moika" all support this assessment without proving it definitively [1][3][4]. The Moika campaign, first observed in late May 2026, published more than 200 malicious npm packages across several waves — with some public reporting placing the cumulative total above 250 — that used a `postinstall` hook to exfiltrate a victim's full environment variables to an external endpoint before fetching an OS-specific second-stage payload [4]. OpenSourceMalware links the two campaigns based on shared "oob"-prefixed infrastructure naming, the reuse of fake-telemetry camouflage, and comparable kill-switch mechanisms, framing WEL1DROPPER/Flooding Dropper as a tradecraft evolution rather than a wholly new actor [1][4]. Separately, Palo Alto Networks Unit 42 has documented concurrent, apparently unrelated npm and PyPI campaigns in the same general window — including a set of ten npm packages, distributed in late July 2026, that download an obfuscated cryptocurrency stealer and a remote access trojan from an external server, and other cross-registry packages combining cloud credential exfiltration with blockchain-based command-and-control and wallet theft — underscoring that this campaign is one strand in a broader surge of open source package abuse rather than an isolated event [1][8].

Recommendations

Immediate Actions

Organizations should treat any host that resolved and imported a package matching the naming and version patterns associated with this campaign as compromised, not merely infected, given the malware's persistence mechanisms and anti-forensic behavior. Security teams should search build logs, lockfiles, and internal package mirrors or caches for unfamiliar dependencies published under the recurring 35.x.y version pattern or containing a `lib/telemetry.js` file, and should hunt for the specific indicators of compromise documented by OpenSourceMalware and Sonatype: outbound connections to the three named Cloudflare Workers subdomains, DNS TXT lookups against `wel1[.]ru` subdomains, disguised file names such as `update_win.exe` or `dotnet_diag_<hex>.exe` on Windows, and `.cache_<hex>`-style artifacts on Linux and macOS [1][3]. Any confirmed match should trigger host isolation, removal of persistence mechanisms

(Registry Run keys, scheduled tasks, LaunchAgents, and any Sliver implant configuration), and a full rebuild rather than an in-place cleanup, followed by rotation of developer and CI/CD credentials only after the environment is verified clean — rotating credentials on a still-compromised host simply hands the new credentials to the same attacker [3]. CSA extends this same full-rebuild posture to the related Moika campaign discussed above: given that campaign's full `process.env` exfiltration technique, CSA assesses that any CI runner with live cloud credentials that installed a Moika package should be treated as compromised outright, and rebuilt rather than cleaned in place, rather than assuming partial exposure [4].

Short-Term Mitigations

Because this campaign specifically bypasses install-time lifecycle-hook monitoring, organizations should not rely on `--ignore-scripts` or postinstall-hook scanning as a sufficient control against npm supply chain risk; any dependency review process needs to also account for payloads that execute at ordinary `require()` / `import()` time. Software composition analysis tooling should be configured to flag brand-new packages with generic, templated names, unusually recent publication dates, and low or zero download history before they are pulled into a build, since this campaign's packages had no organic adoption history to lend them false credibility. Developers using AI coding assistants to suggest or scaffold dependencies should independently verify that a suggested package actually exists on the registry with a plausible history before installing it, since slopsquatting campaigns are explicitly designed to catch exactly this workflow. Given Sonatype's explicit recommendation, teams should also verify the identity and history of any replacement package before installing it as a substitute for a removed malicious dependency, since attackers in this campaign space have shown a pattern of quickly reusing or adjacent-squatting cleared names [3].

Strategic Considerations

The recurrence of import-time payload execution across multiple unrelated 2026 npm campaigns, paired with the emergence of AI-assisted or AI-inspired name generation at scale, suggests that npm-registry defenses built primarily around lifecycle-hook monitoring and manual typosquat detection may be losing ground against an increasingly automated adversary. Organizations should weight dependency vetting toward provenance and behavioral signals — publication velocity, maintainer history, and runtime network behavior in a sandbox — rather than name-similarity heuristics alone, since slopsquatting by design produces names with no legitimate look-alike to compare against. The reuse of legitimate open source red-team tooling such as Sliver as a final-stage implant also argues for extending endpoint

detection coverage to recognize dual-use frameworks regardless of whether their presence appears to originate from a penetration test, and for treating "environment appears to already have C2 tooling installed" as an anomaly worth investigating rather than assuming it reflects sanctioned internal testing.

CSA Resource Alignment

This campaign's use of npm to deliver a cross-platform remote access trojan is directly comparable to the incident documented in CSA's Sapphire Sleet Poisons Mastra AI npm Supply Chain [6], which analyzed how a hijacked contributor account was used to backdoor roughly 145 packages in the @mastra npm scope with a cross-platform RAT and cryptocurrency wallet stealer delivered through a bait dependency. Both incidents illustrate that npm's trust model — whether exploited through a stolen publishing credential or through the sheer volume of newly registered, never-before-seen packages — can be a weak point for cross-platform malware distribution, and both underscore that credential rotation and dependency-tree auditing need to happen immediately upon discovery rather than after formal confirmation of compromise.

More broadly, both the immediate detection needs and the strategic dependency-governance gaps this campaign exposes map to the Threat and Vulnerability Management and Application and Interface Security domains of CSA's AI Controls Matrix (AICM) v1.1 [7]. AICM's third-party and supply chain control objectives call for continuous monitoring of external component provenance and behavior rather than one-time vetting at first install — a posture directly responsive to a threat actor publishing packages by the hundreds specifically to outrun manual review.

References

[1] The Hacker News. "Nearly 800 Malicious npm Packages Deliver Cross-Platform RAT and Infostealer." The Hacker News, August 7, 2026.

[2] Paul McCarty / OpenSourceMalware. "Russian AI Slopsquatting Publishes 700+ Malicious NPM Packages." OpenSourceMalware, August 6, 2026.

[3] Sonatype. "Flooding Dropper Hits npm With 850 Malicious Packages." Sonatype Blog, August 2026.

[4] SafeDep. "oob-moika-tech-depconf-2026 — Campaign." SafeDep Threat Intelligence, 2026.

[5] Socket.dev. "Compromised npm Packages in the AsyncAPI Namespace Deliver Miasma Botnet Loader." Socket.dev, July 14, 2026.

[6] Cloud Security Alliance. "Sapphire Sleet Poisons Mastra AI npm Supply Chain." Cloud Security Alliance, June 22, 2026.

[7] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, June 22, 2026.

[8] Palo Alto Networks Unit 42. "Obfuscated JavaScript Crypto Stealer — Timely Threat Intel." Palo Alto Networks Unit 42, August 6, 2026.

