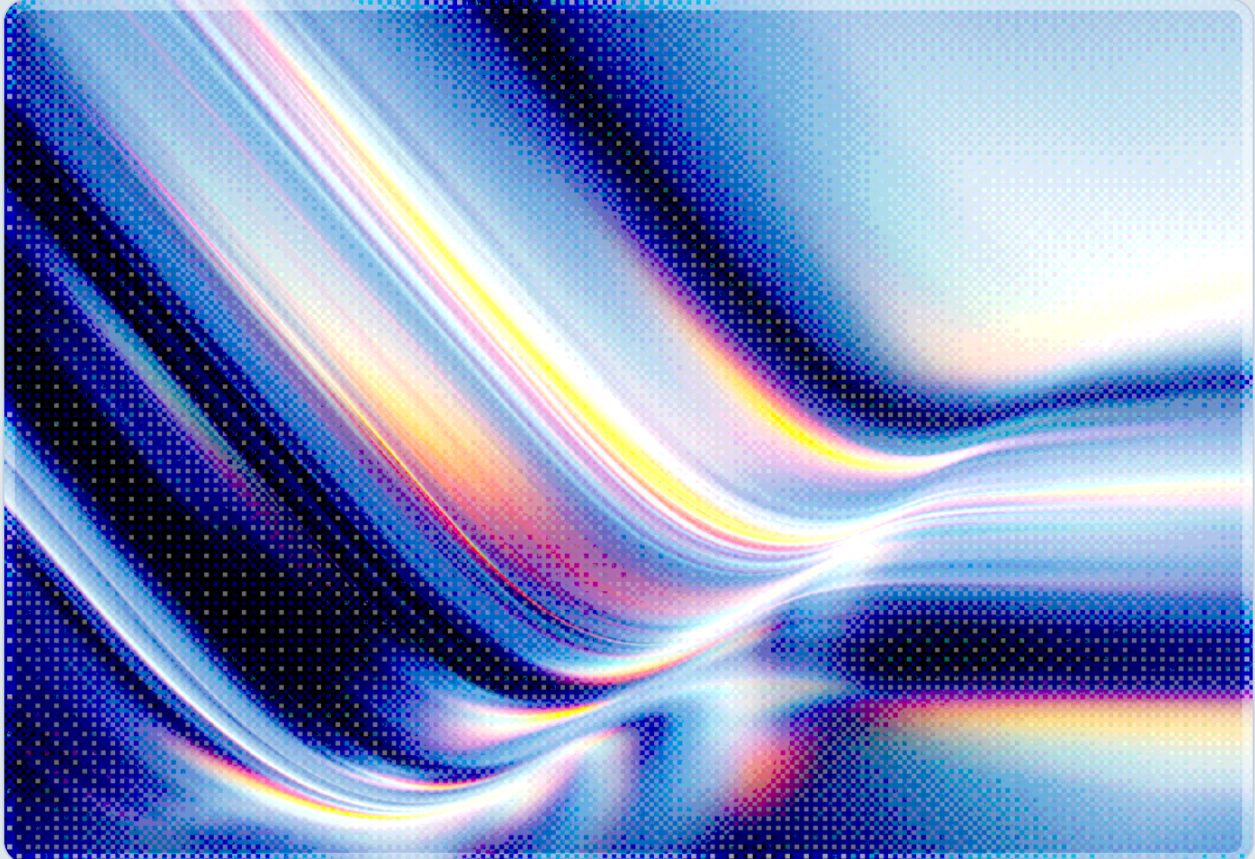


MALFEX: Native Node.js Addons as a Shellcode Loader Path

Security Implications and Guidance for npm Consumers

2026-10-17

 AI-assisted Rapid Research



CSAI

cloud
CSA security
alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

The Malfex npm campaign has moved part of its loader logic out of readable JavaScript and into a precompiled Windows native addon. The package `imgdecode` ships a nine-line JavaScript entry point that loads a prebuilt `.node` DLL, which downloads encrypted shellcode from GitHub and injects it into a new `dllhost.exe` process when the module is first required [1][2]. Because there is no install script and no `binding.gyp`, controls that look for `postinstall` hooks or build steps do not see it.

The addon runs only when standard output is an interactive Windows console, so CI runners and sandboxes with redirected output do not trigger it [1]. This gating, together with the use of a compiled binary, appears designed to avoid the automated analysis that registries and scanners depend on [1]. The operator has repeatedly published new packages after earlier ones were removed, most recently `imgbundle` and `runhelper` after the 28 September removals, and the move to a native addon is consistent with an effort to evade detection of the earlier JavaScript loaders [2][3].

Organizations should treat any Windows developer workstation that loaded `imgdecode`, `imgbundle`, `runhelper`, or the other packages named in this note as potentially compromised. The final payload is reported to steal browser credentials, session cookies, SSH keys, and cryptocurrency wallet data, and to tamper with Windows event logs [1]. Remediation therefore needs to include credential rotation and independent log comparison, not only package removal.

Background

Malfex is described by Checkmarx as an npm malware campaign that appears to be the work of a single operator who has published to the registry since August 2023. Checkmarx reports twelve packages published by the actor, eight of them malicious, delivering three payload paths to Windows systems: a loader for the open-source Overlord remote access trojan written in Go, a chain that installs a Node.js stealer called `movinlike`, and a long-running downloader hidden in the package `function-flag` [3]. Public reporting attributes the activity to a Portuguese-speaking operator using Brazilian hosting, though attribution rests on language and infrastructure artifacts and should be read as an assessment rather than a confirmed identification [1][4].

The scale of the earlier activity is notable mainly because of one package. Checkmarx and press coverage put the campaign's recorded downloads at 40,767, of which `function-flag` accounts for 37,419; reporting on that package suggests it has carried malicious code for an extended period, though we could not confirm the start date in the sources reviewed [3][4]. As of 29 September 2026, three of the malicious packages (`function-flag`, `function-color`, and `cdn-img-fetch`) were reported as still installable, and the advisories that existed covered only part of the malicious versions [3] [5]. The gap between registry takedown and advisory coverage is relevant to defenders because many software composition analysis tools rely on advisory feeds rather than behavioral analysis.

The native addon variant came later. SafeDep reports that `imgdecode` was first published on 30 September 2026 by the publisher account `malfexteste8`, with versions 1.0.0 and 1.0.1 appearing that evening, 1.0.2 on 1 October, and 1.0.3 on 11 October [1]. According to The Hacker News coverage, the operator also published `imgbundle` and `runhelper` around 30 September, after npm removed `native-runner` and `img-to-native` on or about 28 September [2]; we could not confirm these specific dates independently. SafeDep links `imgdecode` to the earlier campaign through a shared GitHub repository (`cavecrew/proj`), an identical encryption passphrase, and a matching stage-two hash from the Checkmarx report, and assesses the connection with high confidence [1].

Download counts for `imgdecode` differ by source: SafeDep reports 929 downloads between 30 September and 12 October, while The Hacker News reports 979 at the time of its 17 October article [1] [2]. The difference is small and may reflect the reporting window, but both figures are modest next to `function-flag`. Download counts on npm include mirrors, scanners, and researchers, so they are an upper bound on exposure and not a count of victims.

Native addons are a legitimate and widely used part of the Node.js ecosystem. Packages such as database drivers and cryptographic libraries ship compiled `.node` files, and Node's N-API provides a stable interface for loading them [6]. A compiled module executes native code with the full privileges of the Node.js process the moment it is loaded, and its initialization function runs at load, before the calling JavaScript regains control, so runtime auditing from within the process cannot intercept it. That property is what makes the format useful to an attacker, and it is also the reason the ecosystem has no single widely adopted signal that separates a benign addon from a hostile one.

Security Analysis

Loading Mechanism and Evasion

According to SafeDep, `imgdecode` contains a precompiled DLL at `prebuilds/win32-x64/imgdecode.node`, loaded by a nine-line JavaScript file [1]. The package has no install script and no `binding.gyp` file, so it neither compiles code at install time nor runs a lifecycle hook. The malicious logic lives in the N-API module initialization function and executes when `require('imgdecode')` is called. This shifts the trigger from install time to first use, which is the same runtime-execution trend CSA has described in other npm campaigns [7].

Three checks in the addon limit exposure to analysis. The addon calls `GetConsoleMode()` on standard output and proceeds only when it is attached to an interactive Windows console, which means that pipes, files, and most CI configurations do not trigger it [1]. After decrypting the downloaded payload, the loader inspects the first bytes for JPEG or PNG signatures and exits if it finds an image, which suggests the host file is a decoy that serves an image to casual viewers [1]. Sensitive strings inside the DLL are obfuscated with a single-byte XOR [1]. We assess that these measures are likely inexpensive and, on the vendor's account, effective against sandbox detonation, but they would not prevent static inspection of the binary by a motivated analyst.

Payload Retrieval and Injection

The addon requests a file from `raw.githubusercontent.com` in the `cavecrew/proj` repository, named `banner.jpg`. Using GitHub's raw content host means that outbound traffic blends with ordinary developer activity and that many egress policies permit it, although this varies by organization. The response is AES-256-CBC encrypted, with the first 16 bytes serving as the initialization vector and the key derived from a SHA-256 hash of a hardcoded passphrase [1]. SafeDep also reports that the `banner.jpg` endpoint returned HTTP 404 during its analysis, so the staged payload may not have been retrievable at that time [1]. Because the payload is fetched at runtime from what appears to be an operator-controlled repository, it can presumably be changed without publishing a new package version, so analysis of the package alone does not establish what will run.

After decryption, SafeDep reports that the addon starts `dllhost.exe` suspended from `System32`, allocates memory in it with `VirtualAllocEx`, writes the shellcode with `WriteProcessMemory`, changes protection to execute-read with `VirtualProtectEx`, and starts a remote thread [1]. If injection fails, the addon falls back to running the shellcode inside the

Node.js process using `VirtualAlloc` and `CreateThread`. Process injection into a signed Windows binary is a well-understood technique, but its presence in an npm dependency has no obvious legitimate use in most packages, which makes it a useful behavioral signal, though defenders should expect to tune for false positives. A `dllhost.exe` process that is a child of `node.exe` and lacks the `/Processid:` argument that COM Surrogate instances normally carry is, according to SafeDep, a useful detection indicator [1].

Downstream Payload Chain

The shellcode is reported to be a Go downloader that retrieves a roughly 71.5 MB executable, `setup.exe`, from a remote server and saves it as `svc_update.exe` [1]. That executable is the `movinlike` stealer and remote access tool, a Node.js 18 application packaged with `pkg`. SafeDep and The Hacker News describe capabilities that include theft of Chromium-browser passwords and cookies, Discord tokens, wallet extensions, seed phrases, and files such as `.pem`, `.ppk`, `.env`, and `.kdbx`, as well as shell execution, file transfer, screenshots, and input logging [1][2]. The malware reportedly persists through a scheduled task and Run key named `MicrosoftEdgeUpdateService`, and tampers with Security and Sysmon event logs [1].

For organizations, the practical consequence is credential exposure on developer machines. SSH keys, `.env` files, npm and GitHub tokens, and browser sessions are the access that lets an attacker move from one workstation to source repositories and publishing pipelines. We infer that a compromised developer endpoint could therefore become a stepping stone to further supply chain compromise, although the reporting reviewed does not document such downstream abuse in this campaign. The claim of log tampering also means that endpoint telemetry from an affected host should not be treated as complete.

Indicators and Detection Opportunities

The table below summarizes indicators and detection ideas drawn from the SafeDep analysis and the Hacker News coverage [1][2]; all rows derive from these two sources, and the second largely restates the first. Defenders should validate them against their own telemetry, since the operator has changed packages and infrastructure repeatedly.

Category	Indicator or signal
npm packages	<code>imgdecode</code> 1.0.0-1.0.3, <code>imgbundle</code> 1.0.0-1.0.2, <code>runhelper</code> 1.0.0; earlier <code>cdn-img-fetch</code> , <code>img-to-native</code> , <code>native-runner</code> , <code>function-</code>

Category	Indicator or signal
	<code>flag</code> , <code>function-color</code>
Publisher	<code>malfexteste8</code> (additional handles include <code>malfexkkj</code> , <code>malfex_user</code> , <code>malfexteste2</code> to <code>malfexteste4</code>)
Network	<code>raw.githubusercontent.com/cavecrew/proj</code> ; <code>104.234.65[.]75</code> on port 700 (port 7007 also reported)
Host artifacts	<code>%TEMP%\svc_update.exe</code> ; <code>MicrosoftEdgeUpdateService</code> Run key and scheduled task; <code>%APPDATA%\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe</code>
Process behavior	<code>dllhost.exe</code> spawned by <code>node.exe</code> without <code>/Processid:</code> ; <code>node.exe</code> fetching from <code>raw.githubusercontent.com</code> followed by <code>dllhost.exe</code> connecting to an unknown host
Package structure	Precompiled <code>.node</code> file under <code>prebuilds/</code> with no native source or <code>binding.gyp</code> , importing <code>WinINet</code> , <code>WinHTTP</code> , or <code>bcrypt.dll</code> with no related functionality

The last row deserves attention because it generalizes beyond Malfex. A native addon for image decoding that imports network and cryptography APIs but no image codec functions is inconsistent with its stated purpose. Registry and enterprise scanners that unpack tarballs and inspect import tables of bundled binaries could flag this class of package without knowing any specific indicator.

Limits of the Available Evidence

The reporting reviewed comes from a small number of vendors, with SafeDep as the primary source for most loader behavior and several other outlets largely restating the same research, and several details, including download counts, package removal dates, and the exact set of eight malicious packages, vary slightly between sources. Dates and package status may also have changed since publication, and `imgdecode` was reported as still available on npm at the time of The Hacker News article [2]. Readers should check the current registry state and the primary vendor analyses before acting on any single indicator. We also have not independently analyzed the binaries, so the technical descriptions here reflect the vendors' findings.

Recommendations

Immediate Actions

Search lockfiles, build caches, and private registry mirrors for the package names listed above, including transitive references, and block them at the registry proxy. On Windows endpoints where any of them was loaded, isolate the host, collect memory before rebooting, and look for the persistence artifacts and `dllhost.exe` parentage described earlier [1]. Rotate browser-stored passwords, session cookies, SSH keys, and npm, GitHub, and cloud tokens from a known-clean device, and move any locally stored cryptocurrency to new wallets. Because the malware is reported to alter Security and Sysmon logs, compare local logs against copies forwarded to a central collector before concluding that nothing occurred [1].

Short-Term Mitigations

Extend dependency review beyond install scripts. Policies that block or flag `postinstall` hooks do not cover a package whose code runs on `require`, so organizations should add checks for prebuilt binaries in newly introduced dependencies, particularly those with no corresponding source or build file. Where a project does not need native addons, tools such as npm's `--ignore-scripts` setting reduce some risks, but they do not prevent a prebuilt `.node` file from executing once required, so they are not a sufficient control on their own [1][11]. We infer this from the reported behavior, in which the addon executes on `require` rather than at install time. Where feasible, route installs through a registry proxy with a cooldown period for newly published packages, since `imgdecode` and its siblings were new, low-download packages with sequential versions published within days.

On endpoints, add detections for `node.exe` spawning `dllhost.exe`, for `node.exe` or its children making requests to raw content hosts followed by unusual process creation, and for cross-process memory writes originating from developer tooling. CSA recommends that developer workstations receive the same endpoint detection coverage as servers, since they hold publishing credentials. Where practical, run first-time evaluation of unfamiliar packages in an isolated environment with an interactive session, since this campaign's console check means that headless sandboxes may report a clean result.

Strategic Considerations

Although the trend rests on a single campaign, the shift toward native code suggests that scanning approaches built around JavaScript source and install-time scripts will cover a shrinking share of malicious behavior. Programs should plan for binary-aware package analysis, provenance requirements for native artifacts, and allow-lists for the small number of addons that a given organization genuinely needs. Registries and organizations alike would benefit from a way to check that a prebuilt binary corresponds to published source, although no such mechanism is universal today. Finally, the gap between package removal and advisory publication reported for this campaign argues for controls that do not depend solely on advisories, such as behavioral monitoring, least-privilege developer environments, and short-lived credentials that limit what a single stolen workstation session can reach [3][5].

CSA Resource Alignment

CSA's recent research note on npm's `indexed-btree` campaign is the closest prior work to this topic. It describes malware that moves from install-time scripts to runtime execution in npm packages, and the MALFEX native addon is another instance of that shift, with execution triggered at `require` and gated on environment conditions [7]. The detection and review recommendations in that note on runtime behavior apply directly to the prebuilt-binary case described here.

CSA's research note on slopsquatted npm packages that deliver a RAT and infostealer at scale covers a related delivery pattern in which developer-facing packages carry remote access tooling and credential theft [8]. The `movinlike` stealer's targeting of browser sessions, wallets, and developer secrets parallels the objectives described there. CSA's note on the Mastra npm scope takeover adds context on how compromise of developer ecosystems can reach organizations building AI applications, which is relevant because developer workstations that hold model provider keys and repository access are a common target [9].

For control mapping, the AI Controls Matrix v1.1 provides the baseline. Its supply chain, application and interface security, and threat and vulnerability management domains are the most relevant to dependency vetting, binary provenance, and endpoint detection on development systems [10].

References

- [1] SafeDep. "[MALFEX Moves Its npm Loader into a Native Windows DLL](#)." SafeDep, October 2026.
- [2] The Hacker News. "[MALFEX npm Package 'imgdecode' Hides Shellcode Loader in Windows Native Addon](#)." The Hacker News, October 17, 2026.
- [3] Checkmarx Zero. "[MALFEX npm Malware Campaign: Three Payloads and an Adversary That Signs Their Work](#)." Checkmarx, October 2026.
- [4] PBX Science. "[MALFEX: Eight Malicious npm Packages Behind 40,767 Downloads Deliver a RAT and a Stealer to Windows Developers](#)." PBX Science, October 17, 2026.
- [5] SecurityWeek. "[Long-Running npm Malware Campaign Accumulates 40,000 Downloads](#)." SecurityWeek, October 2026.
- [6] OpenJS Foundation. "[Node-API Documentation](#)." Node.js Documentation.
- [7] Cloud Security Alliance. "[npm's indexed-btree Campaign Moves Malware to Runtime](#)." CSA Labs, September 2026.
- [8] Cloud Security Alliance. "[Slopsquatted npm Packages Deliver RAT and Infostealer at Scale](#)." CSA Labs, August 10, 2026.
- [9] Cloud Security Alliance. "[Mastra npm Scope Takeover: AI Framework Supply Chain Backdoored](#)." CSA Labs, June 2026.
- [10] Cloud Security Alliance. "[AI Controls Matrix v1.1](#)." Cloud Security Alliance, 2026.
- [11] npm. "[npm-config: ignore-scripts](#)." npm Docs.