

jscrambler npm Package Compromised in Supply Chain Attack

 socket.dev/blog/jscrambler-supply-chain-attack

Socket Research Team

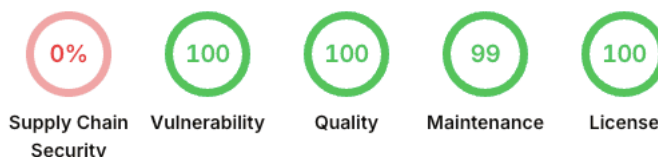
July 11, 2026

npm > jscrambler

 **jscrambler**

Jscrambler Code Integrity API client.

[latest](#) [Source](#) [npm](#) [Copy purl](#) [Socket](#)  0



Known malware SUPPLY CHAIN RISK



This package version is identified as malware. It has been flagged either by Socket's AI scanner and confirmed by our threat research team, or is listed as malicious in security databases and other sources.
Found 2 instances

Install scripts SUPPLY CHAIN RISK



Install scripts are run when the package is installed or built. Malicious packages often use scripts that run automatically to execute payloads or fetch additional code.

Shell access SUPPLY CHAIN RISK



This module accesses the system shell. Accessing the system shell increases the risk of executing arbitrary code.

[Blog](#)

[ResearchSecurity News](#)

A compromised jscrambler npm release added a malicious preinstall hook that runs hidden native binaries on Linux, macOS, and Windows.

Socket Research Team

Summarize with AI

A compromised release of the popular [jscrambler npm package](#) introduced hidden native binaries that execute automatically during `npm install`, exposing users to a supply chain attack before any application code runs.

The malicious **8.14.0** release, published on **July 11**, adds an undocumented `preinstall` hook that invokes `dist/setup.js`. It also introduces new files, including `dist/setup.js` and `dist/intro.js`, along with platform-specific binaries for Linux, macOS, and Windows

embedded in an obfuscated CSI container. None of these files or the install hook exist in the previous release, **8.13.0**.

Socket detected the compromised package 6 minutes after publication.

The package selects and executes one of three bundled binaries depending on whether it is installed on Windows, macOS, or Linux.

At a glance

- **Compromised package:** `jscrambler@8.14.0`
- **Published:** July 11, 2026
- **Weekly downloads:** ~15,800
- **Time to detection:** 6 minutes

Key findings

- Undocumented `preinstall` hook executes code automatically during `npm install`.
- Hidden native binaries added for Linux, macOS, and Windows.
- Payloads are embedded in an obfuscated CSI container.
- Install-time components (`dist/setup.js` and `dist/intro.js`) are entirely new in version 8.14.0.
- These behaviors are absent from the previous release, 8.13.0.

Impact

The compromised `jscrambler` package is used to integrate Jscrambler's JavaScript code-protection tooling into application build pipelines. Developers commonly install it as a development dependency or invoke it through CI systems to process production builds.

Because the malicious code runs through a `preinstall` hook, simply installing `jscrambler@8.14.0` is enough to trigger the bundled platform-specific binary. Users do not need to import the package or run the Jscrambler CLI.

This creates potential exposure across developer workstations, automated build systems, and CI environments. Depending on where the package was installed, the malicious binary may have executed with access to source code, environment variables, build credentials, deployment tokens, and other secrets available to the `npm` process.

The package receives approximately 15,800 weekly downloads, although the number of users who installed the compromised version is not yet known. Socket detected and flagged version 8.14.0 six minutes after it was published.

Recommendation

Users should remove `jscrambler@8.14.0`, rotate any credentials accessible to affected development or CI environments, review installation logs for execution of `dist/setup.js`, and revert to a verified clean release. Pin to version **8.13.0** or another verified clean release until the maintainers publish a remediation.

Socket has reported the compromised release to the Jscrambler maintainers, and a public tracking issue is available on GitHub: <https://github.com/jscrambler/jscrambler/issues/322>

Update: the campaign expanded and moved off the install hook (18:34:32 UTC)

Since our initial report, the same threat actor published four more malicious releases of jscrambler — 8.16.0, 8.17.0, 8.18.0, and 8.20.0 — over roughly three hours, interleaved with remediation attempts. The payload did not change: the bundled `intro.js` blob and the three platform binaries it drops are byte-for-byte identical across every malicious release. What changed is the delivery.

Versions 8.14.0, 8.16.0, and 8.17.0 ran the dropper from a preinstall hook (`"preinstall": "node dist/setup.js"`). Starting with 8.18.0 the install hook is gone entirely — the identical dropper is instead injected as a self-executing function at the top of `dist/index.js` and `dist/bin/jscrambler.js`, so it fires when the package is imported or its CLI is run rather than at install time. This is a deliberate evasion: it defeats scanners that only inspect `preinstall/postinstall` scripts and survives `npm install --ignore-scripts`. Versions 8.18.0 and 8.20.0 also declare a self-dependency (`"jscrambler": "^8.17.0"`), pulling a compromised release transitively.

The latest published version, **8.22.0**, is clean and safe to use (8.15.0 was a clean interim release as well). Users should upgrade to 8.22.0 or pin to it, and audit any machine that installed one of the affected releases.

Jscrambler has since confirmed the unauthorized publication in its [security advisory](#). The company said its investigation found that “the attacker was able to publish the package using an npm publishing credential.” Jscrambler revoked and rotated its publishing credentials and related secrets, deprecated the affected releases, and added further controls around its publishing process.

Technical Analysis

The compromised version 8.14.0 ships two malicious files under `dist/`.

[setup.js](#) is a small loader that runs on install via a preinstall hook such that it gets executed whether or not the package is ever actually imported. `intro.js`, despite its name and `.js` extension, is not JavaScript at all. It is an approximately 7.8 MB large binary container marked with a custom five-byte header (1b 43 53 49 01, i.e. `\x1bCSI\x01`), whose sixth byte is a payload count.

The container packs three gzip-compressed native executables, one per operating system:

- A Linux x86-64 ELF
- A Windows x86-64 PE32+
- An Apple Silicon (arm64) macOS Mach-O

On install or run, [setup.js](#):

1. Reads the container.
2. Selects the single blob matching `process.platform`.
3. Decompresses it to a randomly named hidden file in the system temp directory (appending `.exe` on Windows).
4. Marks it executable.
5. Launches it with `spawn(..., { detached: true, stdio: 'ignore', windowsHide: true })` followed by `unref()`.

In plain terms: the loader silently drops and executes a native binary in the background, tailored to the victim's operating system.

The embedded executables are Rust-built, cross-platform infostealers (Linux x86-64, Windows x86-64, macOS arm64). Analysis of the payloads reveals a broad, developer-focused credential and secret harvester. Its sensitive configuration strings are individually encrypted with ChaCha20-Poly1305; we recovered approximately 2,400 of them.

Capabilities and Target Surface

The malware stores its sensitive strings encrypted (see String Obfuscation below). Once decrypted, the target surface is extensive and clearly oriented toward developer and cloud-operator machines — a rational choice for an npm-delivered payload, since the victims are developers.

Cryptocurrency wallets and seed phrases

Browser-extension wallets are targeted by their extension IDs:

- MetaMask — `nkbihfbeogaeaoehlefnkodbefgpgknn`

- Trust Wallet — `egjidjbpglichdcondbcbdnbeepgdph`
- Coinbase Wallet — `hnfanknocfeofbddgcijnmhnfnkdnaad`
- Phantom — `bfnaelmomeimhlpmgjnjojphpkkoljpa`

The Exodus wallet (`server.exodus.io`) is also targeted. The config contains vault- and seed-extraction keys — HD Key Tree, mnemonic, seedPhrase, recoveryPhrase, and seed — plus scrypt KDF parameters (salt, iterations, N, r, p). This indicates an attempt to decrypt wallet vaults, not merely copy files.

AI coding assistants and MCP server configs

A distinctive and current focus: the malware enumerates configuration for AI developer tooling, which frequently holds API keys and Model Context Protocol (MCP) server credentials. Targets include:

- Claude Desktop — `.config/Claude/claude_desktop_config.json`, `.claude.json`
- Cursor — `.cursor/mcp.json`
- Windsurf — `.codeium/windsurf/mcp_config.json`
- Factory — `.factory/mcp.json`
- Zed — `.config/zed/settings.json`, `context_servers`
- VS Code / VS Code Insiders — `settings.json`, `.mcp.json`, `mcpServers`
- opencode

Cloud credentials

The payload targets all three major clouds:

- **GCP** — `metadata.google.internal`, the compute-metadata service-account token endpoint, `GOOGLE_APPLICATION_CREDENTIALS`, `.config/gcloud`, `credentials.db`, `access_tokens.db`, `application_default_credentials.json`, and Secret Manager access.
- **AWS** — the ECS task-metadata endpoint `169.254.170.2`, `secretsmanager.GetSecretValue / ListSecrets`, and SSM Parameter Store `AmazonSSM.GetParameters / DescribeParameters`.
- **Azure** — the IMDS endpoint `169.254.169.254` and `management.azure.com`.

Messaging and collaboration

- **Discord** — stable, PTB, and Canary bundle IDs; `/api/v9/users/@me` and guild enumeration.
- **Slack** — `.slack.com`, `/api/auth.test`.

- **Telegram Desktop** — `tdata`, `key_datas`.

Browsers, gaming, and OS keyrings

- **Chromium-family browsers** — Chrome, Chromium, Edge, Brave, Vivaldi, and Opera (both Linux profile paths and macOS bundle identifiers), read via embedded LevelDB and SQLite.
- **Firefox** — `profiles.ini`, `cookies.sqlite`, `prefs.js`.
- **Steam** — session theft via `steamLoginSecure`, `loginusers.vdf`, `ConnectCache`.
- **KDE KWallet** — OS keyring access.

Privilege escalation, persistence, and reconnaissance

- **Local privilege escalation** is attempted via `sudo -S -p` (password on stdin) and `systemd-run --system --no-ask-password`.
- **Persistence** references include `systemd` user and system units, `crontab`, and macOS `LaunchAgents`.
- **Host reconnaissance and anti-analysis** strings include `check.torproject.org` with `/api/ip`, the public resolvers `1.1.1.1` and `8.8.8.8`, and machine fingerprinting via `/etc/machine-id`, `/var/lib/dbus/machine-id`, and `/sys/class/dmi/id/board_serial`.

String Obfuscation: Per-String ChaCha20-Poly1305#

The malware's sensitive strings are not stored in cleartext. Each is individually encrypted with ChaCha20-Poly1305 (IETF, 12-byte nonce) and decrypted on demand into lazily initialized statics. Per string, the binary embeds:

- A 32-byte key in `.rodata` (loaded in the decompiled code as a `movups/movupd` pair of adjacent 16-byte reads).
- A 12-byte nonce materialized inline from immediates (a `movabs imm64` followed by a `mov dword ptr [...], imm32`).
- The ciphertext in `.rodata`, whose trailing 16 bytes are the Poly1305 authentication tag.
- An empty AAD.

In the Linux binary, the per-string decrypt helper is at `0x1782f9`, and a large lazy-initializer at `0x25c5e5` decrypts the wallet-targeting block into `.bss` statics. The standalone ChaCha20 core is at `0x425340`; the config AEAD's Poly1305 clamp constant is at `0x2e2c0`, distinct from the rustls TLS instance whose clamp is at `0xbaee0`.

Because Poly1305 is an authenticator, a wrong key, nonce, or ciphertext boundary fails tag verification — decryption is self-validating. We recovered ~2,421 strings this way, cross-validated with two independent AEAD implementations (OpenSSL and libsodium) yielding byte-identical plaintext with zero tag mismatches. A reproduction outline is in the appendix.

Network exfiltration

Static analysis confirmed outbound exfiltration, carried over TLS via rustls. The payload contains a literal POST /upload HTTP/1.1 request with a multipart/form-data body, consistent with uploading harvested data to a drop server. It also constructs numerous POST/PUT requests that query cloud and orchestration APIs using stolen credentials: cloud metadata services, Kubernetes (/api/v1/namespaces), AWS Secrets Manager and SSM, and others.

Indicators of Compromise

Malicious npm package

- [jscrambler@8.14.0](#)
- [jscrambler@8.16.0](#)
- [jscrambler@8.17.0](#)
- [jscrambler@8.18.0](#)
- [jscrambler@8.20.0](#)

SHA-256 Hashes

- dist/setup.js —
a742de963f14a92d24ebcbc7b44ac867e23a20d31d1b0094a13a4f83287f4e60
- dist/intro.js — a41a523ef9517aab37ed6eea0ec881821bdb7aefcb5c5f603adc7907f868c86
- package.json —
bba32ddeab075a5e5015eec50f5d2af364c95b848732c714aea6b6baf78f49f0

Decompressed native payload SHA-256

- Linux ELF — fbbcf4d8f98168f78f5c0c47a9ae56d59ec8ac84a7c9ca6b797fedfb8d62d2bd
- Windows PE —
b7ca95d1b23c8e67416a25cedf741de0917c2096bbc9d24649eea7853d054903
- macOS Mach-O —
c8fd47d36bdf7c825378593ab82ed8c24d1dc52e26b507812393e24e1d5201fd

Subscribe to our newsletter

Get notified when we publish new security blog posts!

Enter your email