

31 Red Hat npm packages backdoored in 72 seconds

 reversinglabs.com/blog/red-hat-cloud-service-npm-packages-backdoored-in-72-seconds

RL Research Team

June 1, 2026

RL has discovered a new supply chain attack affecting 9.8M total downloads across Red Hat's Hybrid Cloud Console JavaScript ecosystem.



RL researchers have uncovered a large-scale, coordinated supply chain attack targeting the [@redhat-cloud-services](#) npm scope. On June 1, an attacker published malicious versions of 31 packages (update) in a 72-second window, injecting obfuscated preinstall malware into every one.

The affected packages collectively represent approximately 9.8 million total downloads and cover the full breadth of Red Hat's Hybrid Cloud Console JavaScript ecosystem — UI components, API clients, build tooling, configuration utilities, and MCP servers.

Scale of the attack

Package	Malicious Version	Published (UTC)	Total Downloads
@redhat-cloud-services/tsc-transform-imports	1.2.2	10:54:09	335,889
@redhat-cloud-services/types	3.6.1	10:54:09	1,281,228

@redhat-cloud-services/eslint-config-redhat-cloud-services	3.2.1	10:54:20	487,137
@redhat-cloud-services/entitlements-client	4.0.11	10:54:20	80,714
@redhat-cloud-services/frontend-components-testing	1.2.1	10:54:21	13,405
@redhat-cloud-services/javascript-clients-shared	2.0.8	10:54:26	507,613
@redhat-cloud-services/config-manager-client	5.0.4	10:54:25	12,448
@redhat-cloud-services/hcc-kessel-mcp	0.3.1	10:54:26	736
@redhat-cloud-services/frontend-components-remediations	4.9.2	10:54:26	112,034
@redhat-cloud-services/hcc-feo-mcp	0.3.1	10:54:27	1,324
@redhat-cloud-services/hcc-pf-mcp	0.6.1	10:54:27	3,415
@redhat-cloud-services/quickstarts-client	4.0.11	10:54:31	5,729
@redhat-cloud-services/chrome	2.3.1	10:54:42	56,869
@redhat-cloud-services/frontend-components-config	6.11.3	10:54:36	788,862
@redhat-cloud-services/frontend-components-config-utils	4.11.2	10:54:36	874,609
@redhat-cloud-services/integrations-client	6.0.4	10:54:36	7,762
@redhat-cloud-services/frontend-components-translations	4.4.1	10:54:47	112,768

@redhat-cloud-services/frontend-components-notifications	6.9.2	10:54:53	701,097
@redhat-cloud-services/frontend-components-advisor-components	3.8.2	10:54:58	28,616
@redhat-cloud-services/notifications-client	6.1.4	10:54:58	24,738
@redhat-cloud-services/frontend-components	7.7.2	10:55:00	1,353,473
@redhat-cloud-services/frontend-components-utilities	7.4.1	10:54:59	1,491,890
@redhat-cloud-services/remediations-client	4.0.4	10:54:59	38,925
@redhat-cloud-services/rule-components	4.7.2	10:54:59	157,274
@redhat-cloud-services/sources-client	3.0.10	10:55:04	44,801
@redhat-cloud-services/host-inventory-client	5.0.3	10:55:09	226,677
@redhat-cloud-services/patch-client	4.0.4	10:55:09	15,442
@redhat-cloud-services/vulnerabilities-client	2.1.8	10:55:10	48,286
@redhat-cloud-services/compliance-client	4.0.3	10:55:21	11,302
@redhat-cloud-services/insights-client	4.0.4	10:55:21	31,116
@redhat-cloud-services/rbac-client	9.0.3	10:55:21	942,556

Total: 31 packages, ~9.8 million downloads

The 72-second window

The malicious versions span from 10:54:09 to 10:55:21 UTC — 72 seconds from first to last publish. This is not a human typing at a terminal; it is automated scripted publishing. The attacker prepared all

31 poisoned archives in advance and executed a single batch push.

The packages span two different GitHub source repositories — *RedHatInsights/frontend-components* and *RedHatInsights/javascript-clients* — confirming that the attacker had access to the *@redhat-cloud-services* npm scope credentials directly, not to any individual repository. This is a scope-level account compromise.

What changed in every package

The modification is identical across all 31 packages: exactly two files changed compared to the prior clean version.

package.json — a *preinstall* script entry is added, pointing execution at *index.js*:

```
json
"scripts": {
  "preinstall": "node index.js"
}
```

index.js — the package's main entry point is completely replaced with a malicious payload. In the clean referential versions, *index.js* contains the package's normal JavaScript exports. In the malicious versions, it is a single-line obfuscated dropper.

No other files are touched. The surgical precision — two files, same modification pattern, 31 packages — is another indicator of automated tooling.

The obfuscation technique: ROT-N + AES-128-GCM + obfuscator.io

Each malicious *index.js* follows the same outer structure — but what it conceals is a three-layer attack pipeline.

Outer wrapper:

```
javascript
try{eval(function(s,n){
  return s.replace(/[a-zA-Z]/g, function(c) {
    var b = c <= "Z" ? 65 : 97;
    return String.fromCharCode((c.charCodeAt(0) - b + n) % 26 + b)
  })
})([/* ~500,000 integer char codes */].map(function(c){return String.fromCharCode(c)}).join(""),
}catch(e){console.log("wrapper:",e.message|e)}}
```

Layer 1 — ROT-N caesar cipher. The outer call encodes ~4 MB of character codes.

String.fromCharCode reconstructs a string, then the *function(s,n)* applies a ROT-N rotation to all alphabetic characters. The rotation value N differs per package — making each blob visually distinct while using the same decode engine. For the *chrome* package, N = 3.

Layer 2 — AES-128-GCM decryption (two blobs). The ROT-decoded string is an async IIFE that decrypts two encrypted blobs using Node.js's *crypto.createDecipheriv (aes-128-gcm, ...)*:

```

javascript
(async()=>{try{
const _c = await import("node:crypto");
const _d = (k, i, a, c) => {
  const d = _c.createDecipheriv("aes-128-gcm", Buffer.from(k,"hex"), Buffer.from(i,"hex"), {aut
    d.setAuthTag(Buffer.from(a,"hex"));
    return Buffer.concat([d.update(Buffer.from(c,"hex")), d.final()])
};
const _b = _d(key_b, iv_b, tag_b, ciphertext_b); // bun downloader
const _p = _d(key_p, iv_p, tag_p, ciphertext_p); // main payload

```

GCM (Galois/Counter Mode) provides authenticated encryption: any tampering with the ciphertext causes decryption to fail, protecting the payload from modification. Keys, IVs, and auth tags are unique per package.

_b — bun runtime downloader. The first blob decrypts to JavaScript that downloads the bun runtime (v1.3.13) from github.com/oven-sh/bun/releases into a temp directory, makes it executable, and stores the path as `globalThis.getBunPath`. This is a legitimate binary from a legitimate source — a deliberate choice to avoid suspicious network traffic.

Layer 3 — Obfuscator.io payload (_p, 634 KB). The second blob — the actual payload — is written to a temp file and executed with bun. If bun is already present in the environment, it is used directly. *Otherwise* _b is eval'd first. After execution the temp file is deleted. The 634 KB payload is heavily obfuscated using the obfuscator.io toolchain (string array encoding, control flow flattening, identifier mangling).

The *try/catch* at every layer ensures silent execution: if any stage fails for any reason, no error surfaces in the build log.

Critically, every one of the 31 index.js files carries a unique SHA-256 hash. The attacker generated individually tailored payloads — rotation values, AES keys, IVs, auth tags, and ciphertexts all differ across packages. This defeats hash-based deduplication and means each package must be analyzed independently.

What the chrome payload does

Full decryption of the *chrome* package payload reveals a cloud credential stealer with worm propagation capabilities. The `_p` blob is a 634 KB bundled JavaScript program that performs several distinct functions when run with the bun runtime.

Credential harvesting. The payload reads and exfiltrates credentials from the developer's build environment via multiple collectors. Confirmed targets include:

Source	Environment Variables / Files
--------	-------------------------------

AWS	AWS_REGION, AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, AWS_SESSION_TOKEN, profile files
Azure / ARM	ARM_TENANT_ID, ARM_CLIENT_ID, ARM_CLIENT_SECRET, ARM_OIDC_TOKEN_FILE_PATH, AZURE_VAULT_NAME
Google Cloud	GOOGLE_APPLICATION_CREDENTIALS, application default credential files
HashiCorp Vault	VAULT_TOKEN, VAULT_AUTH_TOKEN
GitHub	GITHUB_TOKEN, GITHUB_REPOSITORY, GITHUB_WORKFLOW_REF
npm	~/.npmrc
SSH	~/.ssh/

CI/CD targeting. The payload checks *GITHUB_REPOSITORY* and *GITHUB_WORKFLOW_REF* to detect GitHub Actions environments. If the workflow reference matches an expected pattern, it triggers a separate, targeted exfiltration path before exiting. This allows quiet operation inside automated pipelines without leaving an unusual process running.

GitHub API exfiltration. Stolen credentials are formatted into structured reports and exfiltrated via the GitHub API. The payload contains a dedicated GitHub API client (*githubFetch*, *githubJson*) with Authorization header support. The command-and-control (C2) endpoint URL is stored encrypted under an additional layer (f4abccab2 decryptor) on top of the obfuscator.io string table, preventing static extraction.

Worm propagation. The payload contains a *bQ()* ROT-N function and *pQ()* payload generator identical in structure to the obfuscation used to encode the *index.js* files. This strongly suggests the payload is capable of generating new malicious package versions — using any npm credentials it steals to publish poisoned packages to additional scopes.

All 31 packages contain both *_b* and *_p blobs* — they are not specific to chrome. Decrypting *_b* across multiple packages confirms it carries identical bun downloader code in every case; only the AES-GCM key, IV, and authentication tag differ, making the binary blob unique while the decrypted function is the same. The *_p* payloads are all unique-keyed and have not been decrypted for the other 30 packages.

A structural split is visible from file sizes alone: four packages (*chrome*, *frontend-components*, *frontend-components-advisor-components*, *sources-client*) produce *index.js* files of ~4.05 MB, while the remaining 27 produce files of ~4.29 MB — approximately 238 KB more ciphertext in *_p*.

Detection: Behavioral indicators

ReversingLabs Spectra Assure's behavior analysis detects the injection through a cluster of indicators that are uniformly absent in the referential versions and uniformly present in the malicious ones:

Indicator	Description
BH13852	Executes files during installation or upon launch
BH15358	Contains byte sequence decodable to printable string
BH13525	Evaluates code dynamically
BH15183	Converts binary data to string (obfuscation)

Two additional indicators appear only in the chrome package payload:

Indicator	Description
BH16168	Connects through HTTP
BH13831	Sets a JavaScript interval (recurring execution)

The presence of HTTP connectivity and a repeating timer in the chrome payload indicates a more capable stage — one that establishes periodic callbacks, consistent with a persistent info-stealer or C2 beacon executing from the developer's environment. The chrome package, used in the Red Hat Hybrid Cloud Console shell, runs in browser-facing build environments with access to Red Hat infrastructure credentials.

Clean versions already published

By the time of analysis, legitimate maintainers had already pushed clean follow-up versions for all 31 packages (N+1 patch versions). The malicious versions have since been removed from npm. Any project that pinned to these exact versions or ran npm install in the window between publish and removal is affected.

Recommendations

- **If you ran npm install on June 1, 2026 between ~10:54 and ~10:56 UTC** and any of these packages are in your dependency tree — direct or transitive — treat the build environment as compromised. Rotate all credentials, tokens, SSH keys, and secrets accessible from that environment.
- **Audit your lock files.** Check package-lock.json or yarn.lock for exact version pins matching the malicious list above.
- **Block the malicious hashes** in Artifactory, Nexus, or any other artifact proxy. See the indicators of compromise (IoC) table below.
- **Check transitive dependencies.** Several of these packages are widely used as indirect dependencies; the total exposure is likely larger than direct installs suggest.
- **Monitor for anomalous outbound connections** from build agents, particularly HTTP traffic initiated by Node.js processes during npm install.

Indicators of Compromise (IoCs)

All 31 malicious *index.js* hashes are unique (individually crafted payloads):

Package	Version	<i>index.js</i> SHA-256
chrome	2.3.1	21b6409a7b84446310daca5409ad6112ac60a1e4bef97736e53fff5f63bfdef4
compliance-client	4.0.3	5c6cb758a3447bc7e0de34406919a933f9351e90ef04ec43f3bbb401e7004e1b
config-manager-client	5.0.4	5dabf08e2655c012e478074a2cea2b0d34e286c27265a26f3846fc45e5584501
entitlements-client	4.0.11	2a446171b4b981d98b5af6c5606bd63b1570040334210b6ab0a10901b2606fe5
eslint-config-redhat-cloud-services	3.2.1	edd86c0efd776a6bd934fc7b0d4d6da2b256e147cfa83bb0c2814e81d849c427
frontend-components	7.7.2	3f8e522595f32277a0013c7ab0df3ecf336460b56e6b4be9130907f419db3b6d

frontend-components-advisor-components	3.8.2	d8d170af3de17bb9b217c52aaaffdf9395f35ef015a57ef676e406c121e5e223
frontend-components-config	6.11.3	545a1838c66e1771f58d84a17b3e1841e5eeab91a73f4ccc59c9492450a6d9c0
frontend-components-config-utilities	4.11.2	c2a60face766f69f82c972375f35f8ebaa45d6c464176974e631d9a78d6bea0a
frontend-components-notifications	6.9.2	080190bffcaaffacca1f0181fc9024aaaa21500ffdc9926fa5b689ba959965d
frontend-components-remediations	4.9.2	9b99482b75ee89f0d916f2743deeff381ea727e69c71491822477e67891841ad
frontend-components-testing	1.2.1	17c4312b50d69a6f61515edcf71cfaa8271fe2538b942128cfb639d021d042a7
frontend-components-translations	4.4.1	e5f73c888f1250a8895680801975cf177e8c690defd4a999e56f6c08ff64deb8
frontend-components-utilities	7.4.1	89f97557200bd26cc8941c9abaadac2d798a89562401016fbb2c757e3092dfdc
hcc-feo-mcp	0.3.1	c611e49ea46c91013448942c26049741b434cb5dac55fff7c376ca6a4f28580e
hcc-kessel-mcp	0.3.1	7cbace2a186cab2c652305b6e33c8eeb10d4a0ec3a0c8b795de012094fa0d845

hcc-pf-mcp	0.6.1	c178cafa2b3bcbefbbc283b5ab8fc6143e46650631f72451a44327f146a609c3
host-inventory-client	5.0.3	cffc487ee978f7bc06e3856b286940940658884847d38b619a137b8272a75980
insights-client	4.0.4	8d2a09b3727b50f3d035b58bd35b90b504d24dda73a8a24e926a010a58ba5f74
integrations-client	6.0.4	42e165602967c8e1a6fae0113a5179adbe33e18192244fe34b872db09c85e0e6
javascript-clients-shared	2.0.8	09b2301d1589416e0d5fb7a602427a9850dee6713ffa741c0efcfcb1eb4c8952
notifications-client	6.1.4	85b1ed56530bb64d925af4ca50faacd89efb1b63d615238a34adbea9f00e4754
patch-client	4.0.4	df1732f5bfec12e066be44dee02ec8a243e4868d38672c1b1d065359dd735a14
quickstarts-client	4.0.11	7b19ffc2f2bfff75989255e5e807d0f62513153de287eba9cc17003c1dcae8a8
rbac-client	9.0.3	94e8488fd033728eee6666550d5a94b0cc1f7b231d4d85d0affecb0615116722
remediations-client	4.0.4	396cac9e457ec54ff6d3f6311cb5cc1da8054d019ce3ffa1de5741506c7a4ea4
rule-components	4.7.2	1a30a9abe20bab121aaa75ed040565af14e6cdfb745609ee0e7b94a2d814fb9c
sources-client	3.0.10	f961d6897c0ec586cde633e100865b5b1d435cc7c301dbf0f41298ca5b42e17a
tsc-transform-imports	1.2.2	b390d9f708760b799ee5482e8050ce093219140627fcaec6df8812ac9abb9a9b
types	3.6.1	b86c5ae9e95bd841a595440faa3eb6317441e746f241ae8fd641ab59ed1d1966

vulnerabilities- 2.1.8 d1999fd543085918dd542322c6455abde3c57a93b8f7ce871b8809c8bb744af7
client

Watch this space

This attack illustrates the leverage a single scope-level credential compromise can provide: 31 packages, 9.8 million downloads, and a payload capable of self-propagation — all published in under two minutes. The individually crafted payloads across every package, the per-package AES-GCM key material, and the ROT-N variance are deliberate choices to defeat both hash-based detection and bulk analysis. That each package requires independent decryption to fully characterize is itself a structural defense the attacker built in.

ReversingLabs is continuing to monitor the [@redhat-cloud-services](#) scope and will update this post if additional affected versions or payload variants are identified. Full IoC coverage for all 31 packages is available in Spectra Assure. Teams using Spectra Assure to monitor open-source dependencies will have detected this attack via the BH13852 and BH13525 behavior indicators present across all affected versions.

Join RL's free [Spectra Assure Community](#), which helps secure open-source development.

This post was researched by RL AI agents and written by generative AI.

Tags: [Threat Research](#)

Topics

 **REVERSINGLABS**

WEBINAR

SECURING AGENTIC DEVELOPMENT

Featuring Insights from
Forrester's 2026 Agentic Development Security (ADS) Tools Research



GUEST SPEAKER
Janet Worthington
SENIOR ANALYST, FORRESTER

LIVE: JUN 18, 1-2PM ET

FEATURING
FORRESTER®

Forrester Names RL in Agentic Development Security Market

The new landscape report maps 35 vendors addressing an emerging category of risk: AI agents writing insecure code at machine speed.

[Read More about Forrester Names RL in Agentic Development Security Market](#)
[Forrester Names RL in Agentic Development Security Market](#)

Subscribe

Get the best of RL Blog delivered to your in-box weekly. Stay up to date on key trends, analysis and best practices across threat intelligence and software supply chain security.