

Windows Infostealer Hits npm and Ruby

 opensource.malware.com/blog/windows-infostealer-stubmaker-npm-ruby

cb482791-4ef1-4762-96ad-b0ca4bdd538e

August 19, 2026



MALWARE

AUG 19, 2026



Today, we came across independent research from OpenHack documenting [a 37-package typosquatting campaign on npm](#), targeting chalk, commander, lodash, typescript, react, and axios. It described a 22 MB Rust loader carrying an embedded Go infostealer that decrypts entirely in memory with no second-stage download. That immediately caught our attention, because earlier this week we documented [the RubyGems typosquatting campaign StubMaker](#), which has the same shape. That includes the specific file size.

StubMaker used a fake native-extension install hook to deliver a Windows loader, which decrypted an embedded Go infostealer entirely in memory. That campaign included 16 malicious package names across three RubyGems accounts, typosquatting `bundler`, `i18n`, `rake`, and `activesupport`, and ending in theft of browser credentials, cryptocurrency wallets, and Telegram data, exfiltrated via Gofile to a webhook at `dresslee.com`. One name, `brumdlr`, was published under two separate owners. The operator got it taken down once and then successfully republished it under a second account.

See all the threat reports [here](#).

We pulled the payloads for the npm packages and cross-checked against the Ruby samples. Both sets of packages share infrastructure that confirms this is the same threat actor.

- Both installers beacon to the same C2 endpoint: `193.70.34.101:20099/vote`
- Both pull a Windows loader from the same GitHub release:
`github.com/bebraz1/qPzM50V1AKG0rVLH/releases/download/null/main.exe`
- The loader and the embedded Go infostealer it decrypts are byte-for-byte identical between the two campaigns (matching SHA-256 hashes, not just similar behavior)

Identical hashes mean identical files. This was one threat actor running two typosquatting fronts against two package ecosystems, sharing a single payload and a single C2 backend. This post covers what's different about the npm side, and what the timeline between the two campaigns actually tells us about how this operation moves.

TL;DR

- **Threat Type:** Supply-chain downloader and Windows infostealer
- **Affected Ecosystems:** RubyGems and npm
- **Campaign Scope:** 16 package RubyGems packages and 40 npm packages on npm (we found 3 beyond OpenHack's set)
- **Impact:** Theft of credentials, authenticated browser sessions, payment-card data, cryptocurrency wallets and seed phrases, Telegram data, and host information
- **Key Finding:** Both versions deliver the same Rust loader and the same Go infostealer (confirmed by matching SHA-256 hashes, not just similar behavior) through two independent typosquatting campaigns using different install hooks

- **Exfiltration:** Stolen data is uploaded to Gofile; the resulting link is reported to an attacker webhook, confirmed on RubyGems as `dresslee.com` over unencrypted HTTP

The npm packages

Like the Ruby packages, the npm cohort are typosquats of very popular packages. Some are quite clumsy, but a few (`chalk-core` among them) read as more plausible sub-package names than the rest. In addition to the 37 packages documented by OpenHack, we typecript-cli, typecript-core, and typescrit-cli to this campaign.

The 40 packages went live in an eight-minute burst starting 02:48 UTC on August 16, and npm had pulled every one of them down by 04:12 UTC. Registry metadata recovered after takedown shows the packages split across at five npm accounts, each registered with a distinct, generic-looking Gmail address:

- `verifikasikkiunila555a` `verifikasi.kki.unila@gmail.com`
- `whatisthisapplive20238261` `whatisthisapplive2023@gmail.com`
- `fullbasketpropertywebsiteb53a` `fullbasketpropertywebsite@gmail.com`
- `nodemailert451e6` `nodemailert4@gmail.com`
- `antoniorodriguezmonte84ddf8` `antoniorodriguezmonte84@gmail.com`

That's a burner-account pattern consistent with what we saw across `gemlewwqquh1`, `mod8rz41mje`, and `rbq95bwt6q` on RubyGems.

Package	Typosquat	npm Account	First Published (UTC)	Last Modified (UTC)
testingsmthb1g	none (initial linked sample)	<code>whatisthisapplive20238261</code>	02:28	04:09
axious-core	<code>axios</code>	<code>nodemailert451e6</code>	02:56	04:07
axois-http	<code>axios</code>	<code>nodemailert451e6</code>	02:54	04:07

Package	Typosquat	npm Account	First Published (UTC)	Last Modified (UTC)
chalk-core	chalk	whatisthisapplive20238261	02:48	04:09
chalk-es	chalk	antoniorodriguezmonte84ddf8	02:48	04:12
chalk-lib	chalk	verifikasikkiunila555a	02:48	04:09
chalk-util	chalk	verifikasikkiunila555a	02:48	04:09
comand	commander	verifikasikkiunila555a	02:49	04:10
comander-cli	commander	antoniorodriguezmonte84ddf8	02:49	04:12
comander-lib	commander	fullbasketpropertywebsiteb53a	02:49	04:11
comanderjs	commander	verifikasikkiunila555a	02:49	04:10
commander-lib	commander	antoniorodriguezmonte84ddf8	02:49	04:12
commandor-cli	commander	whatisthisapplive20238261	02:49	04:09
commandor-core	commander	nodemailert451e6	02:49	04:07
commandor-lib	commander	nodemailert451e6	02:50	04:07
commandorjs	commander	antoniorodriguezmonte84ddf8	02:49	04:12
ladash-cli	lodash	nodemailert451e6	02:51	04:07

Package	Typosquat	npm Account	First Published (UTC)	Last Modified (UTC)
loadashjs	lodash	verifikasikkiunila555a	02:50	04:10
lodahs-cli	lodash	whatisthisapplive20238261	02:51	04:09
lodahsjs	lodash	fullbasketpropertywebsiteb53a	02:50	04:11
lodash-lib	lodash	whatisthisapplive20238261	02:50	04:09
lodhash-cli	lodash	antoniorodriguezmonte84ddf8	02:51	04:12
lodsh-cli	lodash	whatisthisapplive20238261	02:51	04:09
typescript-cli	typescript	verifikasikkiunila555a	02:52	04:10
typescript-core	typescript	verifikasikkiunila555a	02:53	04:10
typecript-cli	typescript	nodemailert451e6	02:51	04:07
typecript-core	typescript	fullbasketpropertywebsiteb53a	02:52	04:11
typescript-cli	typescript	fullbasketpropertywebsiteb53a	02:52	04:11
typescript-core	typescript	verifikasikkiunila555a	02:53	04:10
typescirpt-cli	typescript	whatisthisapplive20238261	02:52	04:09
typescirpt-core	typescript	fullbasketpropertywebsiteb53a	02:52	04:11

Package	Typosquat	npm Account	First Published (UTC)	Last Modified (UTC)
typescrip-cli	typescript	verifikasikkiunila555a	02:52	04:10
typescriptt-cli	typescript	fullbasketpropertywebsiteb53a	02:52	04:11
typescriptt-core	typescript	fullbasketpropertywebsiteb53a	02:53	04:11
typescrit-cli	typescript	whatisthisapplive20238261	02:52	04:09
typesript-cli	typescript	nodemailert451e6	02:52	04:07
typesript-core	typescript	nodemailert451e6	02:52	04:07
typscript-cli	typescript	whatisthisapplive20238261	02:52	04:09
typscript-core	typescript	fullbasketpropertywebsiteb53a	02:52	04:11
raectjs	react	fullbasketpropertywebsiteb53a	02:53	04:11

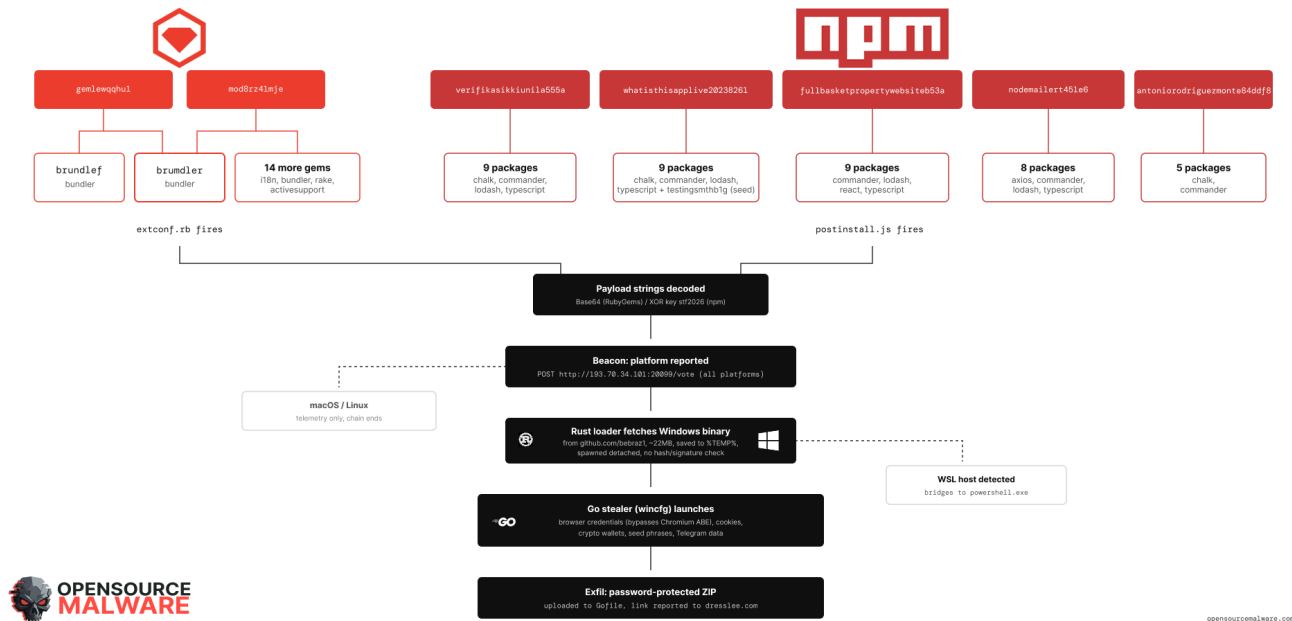
Analyzing two ecosystem attacks

The attack chain is very similar between the Ruby and npm packages, differing just at steps 1 and 2:

1. Victim installs a typosquatted package (gem or npm).
2. Install-time hook fires automatically without an import/require needed:
 - RubyGems: `extconf.rb`
 - npm: `postinstall.js`

3. Hook is obfuscated — payload strings, URLs, and IP octets are stored as byte arrays and XOR-decoded at runtime (key stf2026).
4. Beacon: POSTs platform label to `http://193.70.34.101:20099/vote` (install/host selector).
5. Windows-only payload — Linux/macOS installs receive telemetry only.
6. Downloads a Rust-based loader / `main.exe` (~22MB) from a GitHub Release hosted at `github.com/bebraz1`, saved to `%TEMP%`, spawned detached with no hash/signature check.
7. WSL hosts: script detects WSL (`/proc/version`, `WSL_DISTRO_NAME`) and bridges out to `powershell.exe` on the Windows side to escape the Linux sandbox.
8. Loader launches a Go-based stealer (`wincfg`) — harvests browser credentials (bypasses Chromium app-bound encryption), crypto wallets, seed phrases, Telegram data.
9. Exfil: password-protected ZIP via Gofile; C2/upload domain `dresslee.com`.

StubMaker Dual Ecosystem Attack Path



Here's a summary of what's unique about the npm packages:

The build-fakery is gone because npm doesn't need it

StubMaker's Ruby installer had to work inside `extconf.rb`, a file RubyGems expects to configure and build a native C extension. It fabricated a working-looking `Makefile` and `make_stub/make_stub.bat` scripts (why we called this campaign "StubMaker") to make a fake build look successful. npm has no equivalent expectation to fake. Instead it uses `postinstall.js`, a standard, documented lifecycle hook npm runs automatically, no pretense required. Moving to npm let the operator drop an entire subsystem of the Ruby version, because npm already hands you unrestricted code execution on install by design.

The obfuscation was re-implemented, not reused

The Ruby installer decoded its loader URL from Base64 whereas the npm installer uses repeated-key XOR with a hardcoded key, `stf2026`. Different technique, comparable simplicity. This reads as a deliberate rewrite for the new platform rather than a copy-paste of the Ruby approach, which argues for an operator (or toolkit) that treats the obfuscation layer as a per-ecosystem component to swap rather than a fixed piece of shared code.

npm ran wider and faster, from parallel accounts rather than one

RubyGems saw 16 malicious package names land across three accounts over roughly two days, including a clear catch-and-adapt cycle: the first two packages (`brumdlër`, `brundlëf`) were killed, and the operator came back under a new account with 15 more (one of them a successful republish of `brumdlër` itself under the new owner).

npm saw 40 packages land in an eight-minute window across five accounts running in parallel, with no visible adaptation cycle and no second wave yet identified. That's a structurally different approach, not just a bigger one: RubyGems' pattern was sequential and single-point-of-failure. One account gets caught, the operator adapts and returns. npm's pattern spread the same burst across multiple burner accounts simultaneously, so losing any one account wouldn't have taken down the whole batch. However, in spite of the different approach, the npm packages were rapidly discovered and removed as a cohort.

Indicators of Compromise (IOCs)

The network indicators and payload hashes below are shared with the original StubMaker RubyGems campaign. They're confirmed identical, not just similar.

Network Indicators

193.70.34.101
193.70.34.101:20099
http://193.70.34.101:20099/vote
github.com/bebraz1/qPzM50V1AKG0rVlH
https://github.com/bebraz1/qPzM50V1AKG0rVlH/releases/download/null/main.exe
https://api.ipify.org
https://upload.gofile.io/uploadfile
https://gofile.io/d/
dresslee.com
dresslee.com:20027
http://dresslee.com:20027/xf39jMJ9P1

Here's an example of how the threat actor "obfuscated" the IP address:

```
def telemetry_host
  %w[193 70 34 101].join('.')
end
```

File Indicators

postinstall.js (npm install hook, the equivalent of extconf.rb on RubyGems)
main.exe
abe_payload.dll (RubyGems-confirmed only)

SHA-256 Hashes

Downloaded Rust loader: main.exe
6f088ade49456db2422c3edfbb9998f4a3e9cce7c4c00a7279fb45d672a82b7d

Recovered Go infostealer
1afff50ca4064310d3492c652e1c3168216dcb42063e0b26c223038db46b8731

Recovered embedded ABE helper DLL (RubyGems-confirmed only)
67719fa6fcaa97936bf678565d6777db5c194e14efeba16864be8dac966e24bc

We don't have a hash for `postinstall.js` itself. OpenHack's write-up describes its behavior (XOR-decoding a config block with key `stf2026`) but doesn't publish a hash for the file, unlike our own `extconf.rb` and Ruby installer hashes in the original StubMaker post.

What this means for defenders

If your organization runs both a Ruby and a Node stack, this operation had two independent paths into you, sharing nothing but infrastructure: the same beacon IP, the same GitHub-hosted loader, the same Go infostealer. A team watching `Gemfile.lock` and a team watching `package-lock.json` could each investigate “a typosquat campaign” in isolation and never realize it’s the same actor hitting them from two directions. Blocking `193.70.34.101:20099` and the `dresslee.com` webhook at the network layer covers both fronts regardless of which ecosystem delivered the initial install hook. That’s a more durable control than package-name blocklisting alone, which has to be rebuilt every time the operator picks new typosquat targets.