

GlassWorm RAT Delivered via Malicious Chrome Extension (Keylogger, Cookie Theft)

 aikido.dev/blog/glassworm-chrome-extension-rat

Ilyas Makari

March 18, 2026



A couple of days ago, we covered GlassWorm [compromising hundreds of GitHub repositories](#) and [a popular React phone number package on npm](#). We kept digging into the full payload and found a multi-stage framework that installs a persistent RAT and, deep in Stage 3, force-installs a

Chrome extension posing as Google Docs Offline. It logs keystrokes, dumps cookies and session tokens, captures screenshots, and takes commands from a C2 server hidden in a Solana blockchain memo.



Stage 1: The Initial Infection

Hijacked Projects

GlassWorm gains its initial foothold through malicious packages published across npm, PyPI, GitHub, and the OpenVSX marketplace. The threat actor operates on two tracks simultaneously: (1) crafting new malicious packages from scratch, and (2) [compromising the accounts of maintainers](#) to push malicious versions of legitimate projects.

Two Loader Flavors

GlassWorm is perhaps best known for its [invisible Unicode loader](#) we've covered in previous posts, but it is not the only delivery mechanism in use. A second, more direct variant uses a conventional obfuscated `preinstall` script, as seen in the recent [compromise of react-native-country-select](#) on npm. Both ultimately achieve the same outcome and share the same blockchain-based C2 beacon, but they take very different paths to get there.

Regardless of which loader reaches the victim machine, the Stage 1 execution logic is the same. After a 10-second startup delay, the loader performs two checks before proceeding.

Geofencing. The loader checks five locale signals (`os.userInfo().username`, `process.env.LANG`, `process.env.LANGUAGE`, `process.env.LC_ALL`, and the Intl resolved locale) against `/ru_RU|ru-RU|Russian|russian/i`, and checks the system timezone and UTC offset against a hardcoded list of Russian timezones spanning Europe/Moscow through Asia/Anadyr. If a Russian locale is detected, execution stops.

Rate limiting. The loader reads `~/init.json` (or `%USERPROFILE%\init.json` on Windows) and checks a stored timestamp. If the file was written less than two hours ago, execution stops. Otherwise, the timestamp is updated.

Solana Blockchain C2

Finally, the loader queries the Solana blockchain for its C2 address. Rather than hardcoding a URL that can be taken down, the threat actor stores it in the `memo` field of a Solana transaction. The loader calls `getSignaturesForAddress` against a hardcoded wallet, cycling through nine public RPC endpoints until one responds:

- `https://api.mainnet-beta.solana.com`
- `https://solana-mainnet.gateway.tatum.io`
- `https://go.getblock.us/86aac42ad4484f3c813079afc201451c`
- `https://solana-rpc.publicnode.com`
- `https://api.blockeden.xyz/solana/KeCh6p22EX5AeRHxMSmc`
- `https://solana.drpc.org`
- `https://solana.leorpc.com/?api_key=FREE`
- `https://solana.api.onfinality.io/public`
- `https://solana.api.pocket.network/`

Two wallet addresses have been observed across the two loader variants:

- `BjVeAjPrSKFiingBn4vZvghsGj9KCE8AJVtbc9S8o8SC` (Unicode loader)
- `6YGcuyFRJKZtcaYCCFba9fScNUvPkGXodXE1mJiSzqDJ` (Obfuscated preinstall loader)

The loader polls in a 10-second loop until it finds a transaction with a non-null memo field. Solana's memo feature was designed to add annotations to transactions, but here it functions as a covert dead-drop. The memos are permanent, publicly visible on-chain, and stored on infrastructure that cannot be taken down by any single party. The operator can update the Stage 2 URL at any time by sending a new Solana transaction with a new memo. No package needs to be modified, no infrastructure needs to be redeployed, and there is nothing for defenders to block at the network layer.

The memo observed on wallet `6YGcuyFRJKZtcaYCCFba9fScNUvPkGXodXE1mJiSzqDJ` is:

```
{"link": "aHR0cDovLzQ1LjMyLjE1MC4yNTEvM2U0VGc4ViUyRjhQ21PSktpcEFTQURnJTNEJTNE"}
```

The `link` value is a Base64-encoded URL. Decoded:

```
http://45.32.150.251/3e4Tg8V%2F8aCm0JKipASADg%3D%3D
```

The loader fetches this URL with an `os` header set to the current platform (`darwin`, `linux`, or `win32`), allowing the C2 at `45.32.150[.]251` to serve platform-specific payloads. The response body is the encrypted Stage 2 payload, which is subsequently decrypted and executed. At the time of writing, the C2 was still active and returning a payload for `win32`.

Stage 2: Secret Exfiltration

The Stage 2 payload is a full data-theft framework with credential harvesting, crypto wallet exfiltration, host profiling, and its own dropper logic for the final and persistent Stage 3. Everything collected is staged under `%TEMP%\hJxPxPHP\`, zipped, and sent via a POST request to `http://217.69.3[.]152/wall`.

Crypto Wallet Theft

The payload recursively searches `%APPDATA%` and `%LOCALAPPDATA%` for browser extension profiles and standalone wallet application data. It targets 71 browser extension wallet IDs, covering virtually every major wallet in use: MetaMask, Phantom, Coinbase, Exodus, Binance, Ronin, Keplr, etc. It also collects `.txt` files from the victim's Documents and Desktop folders, and copies images whose filenames match keywords associated with seed phrases or crypto holdings.

Developer Credential Theft

Stage 2 targets the credential stores that a software developer is likely to have on their machine. The payload reads `~/.npmrc` and `process.env.NPM_TOKEN`. Any token found is validated in real time against `https://registry.npmjs.org/-/whoami` before exfiltration. It also extracts tokens via the `git credential` command and from VS Code's internal storage.

Cloud Secrets Exfiltration

Stage 2 also copies credential files for AWS, GCP, Azure, Docker, Kubernetes, SSH keys, Heroku's `.netrc`, DigitalOcean's `doctl`, and Terraform.

Host Profiling

Finally, it generates a `system_info.txt` with a detailed hardware profile, environment variables, installed applications, running processes, disk layout, and OS details.

Preparing for Stage 3

After exfiltration, Stage 3 is prepared by downloading two components:

- **3a)** Phishing binary for physical crypto wallets (Ledger & Trezor).
- **3b)** An immortal Websocket-based RAT, saved as `%APPDATA%\QtCvyfVWKH\index.js`, which ships with several binaries:
 - `c_x64.node` / `f_ex86.node` — browser credential stealers
 - `data` — Chrome App-Bound Encryption bypass
 - `index_ia32.node` / `index_x64.node` — HVNC modules
 - `w.node` (Windows) / `m` (macOS) — Installs a malicious browser extension

The RAT payload isn't simply hardcoded with a URL. Instead, it fetches the public Google Calendar page at <https://calendar.app.google/2NkrckKj4T6Dn4uK6> and extracts the title of the invite. That value is Base64-decoded and appended as a URL slug to [http://45.32.150\[.\]251](http://45.32.150[.]251). Using Google Calendar as an indirection layer for payload delivery [is a pattern we've been tracking since March 2025](#), and it continues to appear consistently across this threat actor's tooling.

Stage 3a: Hardware Wallet Phishing

On machines where `%APPDATA%\Ledger Live` exists, the Stage 3 fetches a .NET WPF binary from [http://45.32.150\[.\]251/led-win32](http://45.32.150[.]251/led-win32), drops it at `%TEMP%\SKuyzYcDD.exe`, and adds `HKCU\Software\Microsoft\Windows\CurrentVersion\Run\UpdateLedger` for persistence. The file's internal name is `Assaac.exe`, attributed to a company calling itself "LLC LogicSub" (SHA-256: `06fab21dc276e3ab9b5d0a1532398979fd377b080c86d74f2c53a04603a43b1d`). The binary is not a RAT. Its sole function is to steal crypto wallets by impersonating Ledger Live and Trezor.

Screenshot of the [CAPE Sandbox](#) execution

On startup, it queries <https://ipapi.co/xml> and checks the returned country against nine CIS-region exclusions: Russia, Kazakhstan, Kyrgyzstan, Azerbaijan, Tajikistan, Uzbekistan, Belarus, Moldova, and Armenia. If the victim is in any of those countries, execution halts. This mirrors the geofencing logic from Stage 1.

The binary registers a WMI event subscription to detect USB device connections:

```
SELECT * FROM __InstanceCreationEvent WITHIN 2 WHERE TargetInstance ISA 'Win32_PnPEntity'
```

When a Ledger or Trezor device is plugged in, the matching phishing window opens. The Ledger UI displays a fake configuration error and presents 24 numbered recovery phrase input fields. The Trezor UI displays a fake "Firmware validation failed, initiating emergency reboot" message with the same 24-word input layout. Both windows include a "RESTORE WALLET" button.

A background loop running at one-second intervals calls `Process.GetProcessesByName` to kill any real Ledger Live processes and re-displays the phishing window if the victim closes it. The `Window_Closing` handler intercepts attempts to close with a dialog warning that closing could result in the loss of crypto assets.

When a valid phrase is submitted, the 24 words are transmitted via `System.Net.Http.HttpClient` to `45.150.34[.]158`.

Stage 3b: Websocket-based RAT

The RAT is saved as `%APPDATA%\QtCvyfVWKH\index.js` and comes with two persistence mechanisms to survive reboots. A scheduled task named `UpdateApp` runs at startup with the highest privileges, and an `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` key executes a PowerShell launcher at `%LOCALAPPDATA%\QtCvyfVWKH\Aghzgy.ps1`.

Immortality is built in. The script hooks `SIGINT`, `SIGTERM`, `SIGQUIT`, `SIGHUP`, `SIGUSR2`, `uncaughtException`, and `unhandledRejection`. On any of those signals, it schedules to re-download and restart the payload. If the process is killed, it just rebuilds itself later.

DHT-Based C2

The RAT does not hardcode its main C2 address. It performs a DHT lookup for the pinned public key `3c90fa0e84dd76c94b1468f38ed640945d72bc12`, bootstrapping via `dht.libtorrent.org`, `router.bittorrent.com`, and `router.utorrent.com`.

The operator stores live config in the DHT value at that key. If `dht.get` fails outright, the script retries after five minutes. If it returns no value, the script falls back to the Solana memo dead-drop: it polls `getSignaturesForAddress` for wallet `BjVeAjPrSKFiingBn4vZvghsGj9KCE8AJVtbc9S8o8SC`, decodes the IP from the memo's link field, and reruns the lookup. The Solana memo can be updated with a new IP at any time to reseed DHT reachability without touching the loader.

The recovered config of the attacker's infrastructure:

- `217.69.0[.]159:10000` - DHT bootstrap node
- `45.32.150[.]251` - WebSocket-based C2
- `217.69.3[.]152:80` - Exfiltration server

The C2 Commands

Once the DHT config is resolved, the script opens a `Socket.IO` connection to `45.32.150[.]251:4787` and processes five categories of C2 command:

- `start_hvnc` / `stop_hvnc`: Deploys the HVNC native module for hidden remote desktop access.

- **start_socks / stop_socks**: Fetches a WebRTC module from `http://45.32.150[.]251/module/wrtc`, installs it to `%APPDATA%\_node_x64\webrtc\wrtc-win32-x64\index.js`, and runs it as a SOCKS proxy, turning the victim machine into a proxy node, serving as infrastructure for the threat actor to carry out other attacks from the victim's IP.
- **reget_log**: Runs the full browser credential theft and exfiltration pipeline.
- **get_system_info**: Sends OS, CPU, memory, hostname, uptime, username, and home directory.
- **command**: Executes arbitrary attacker-supplied JavaScript via `eval()`, giving the operator full remote code execution.

Browser Credential Theft

The script targets Chrome, Edge, Brave, Opera, Opera GX, Vivaldi, and Firefox. It enumerates profile directories, checks cookies to identify live profiles, then calls into `c_x64.node` to extract credentials directly from the browser's SQLite databases (Login Data, Cookies, Web Data). Chrome v127+ encrypts its master key using App-Bound Encryption, which ordinarily prevents a process outside Chrome from reading it. The data binary attempts to bypass this security feature. Results are staged as JSON files under `%TEMP%\EUXFuxz0Ve\`, covering cookies, saved logins, autofill entries, browsing history, bookmarks, and payment cards. The directory is zipped, encrypted, and POSTed to `217.69.3[.]152:80/log`.

Malicious Chrome Extension RAT

The script also force-installs an extension masquerading as *Google Docs Offline* (version 1.95.1). It resolves its C2 from a separate Solana memo, polling `getSignaturesForAddress` for wallet `DSRUBTziADDHSik7WQvSMjvwCHFsbsthrbbjwMoJPUIw`. The memo from that wallet carries `c2server` and `checkIp` fields pointing to the extension's API server and geo-IP helper. After parsing the memo, the extension registers as an agent via `POST /api/register`, stores the returned `agent_id` in `chrome.storage.local`, and begins polling `GET /api/commands?agent_id=<id>` at a randomized 5-30 second interval.

Screenshot of the malicious extension next to the legitimate one

The operator can issue the following commands:

- **domsnapshot**: exfiltrates the full `document.documentElement.outerHTML` of the active tab
- **getcookies**: collects cookies, optionally filtered by domain
- **localstoragedump**: dumps all `localStorage` key/value pairs
- **take_screenshot**: captures a base64-encoded tab screenshot
- **capture_clipboard**: reads the clipboard via `navigator.clipboard.readText()`
- **history**: extracts up to 5,000 browser history entries (default window: 7 days)
- **bookmarks**: exports the full bookmark tree

- **enumeration**: fingerprints the browser, hardware, WebGL/GPU details, and probes installed extensions via `chrome-extension://<id>/manifest.json`
- **startkeylogger / stopkeylogger / getkeyloggerdata**: hooks keydown, keyup, keypress, input, change, focus, and blur events across all pages; captures typed values, form metadata, and target element context; flushes to `POST /api/exfil` every five seconds

All collected data is sent to `POST /api/exfil` as `{ agent_id, action, payload }`.

The extension also performs targeted session surveillance. It pulls monitored site rules from `/api/get-url-for-watch` and ships with Bybit (.bybit.com) pre-configured as a target, watching for the secure-token and deviceid cookies. On detection it fires an auth-detected webhook to `/api/webhook/auth-detected` containing the cookie material and page metadata. The C2 can also supply redirect rules that force active tabs to attacker-controlled URLs.

Detection and Protection

Invisible threats require active defenses. You cannot rely on visual code review or standard linting to catch what you cannot see. At Aikido, we've built detection of Glassworm and other threat actors directly into our malware scanning pipeline.

If you already use Aikido, these packages would be flagged in your feed as a 100/100 critical finding.

Not on Aikido yet? [Create a free account](#) and link your repositories. The free plan includes our malware detection coverage (no credit card required).

Finally, a tool that can stop supply-chain malware in real time as they appear can prevent a serious infection. This is the idea behind [Aikido Safe Chain](#), a free and open-source tool that wraps around npm, npx, yarn, pnpm, and pnpm and uses both AI and human malware researchers to detect and block the latest supply chain risks before they enter your environment.

Indicators of Compromise

Network — IP Addresses

- `45.32.150[.]251` — Stage 2 payload delivery, Stage 3 WebSocket RAT (:4787)
- `217.69.3[.]152` — Exfiltration server: Stage 2 (/wall), Stage 3 (/log)
- `217.69.0[.]159` — DHT bootstrap node (:10000)
- `45.150.34[.]158` — Ledger/Trezor seed phrase exfiltration

Network — C2 URLs

- `http://45.32.150[.]251/3e4Tg8V%2F8aCm0JKipASADg%3D%3D` — Stage 2 encrypted payload

- [http://45.32.150\[.\]251/led-win32](http://45.32.150[.]251/led-win32) — Ledger/Trezor phishing binary download
- [http://45.32.150\[.\]251/get_arhive_npm/nt70c2J3PG%2BfPBSFHJKoWQ%3D%3D](http://45.32.150[.]251/get_arhive_npm/nt70c2J3PG%2BfPBSFHJKoWQ%3D%3D) — Native module archive
- [http://45.32.150\[.\]251/get_encrypt_file_exe/E/E%2BT9tEjmURMwNnCCY2CA%3D%3D](http://45.32.150[.]251/get_encrypt_file_exe/E/E%2BT9tEjmURMwNnCCY2CA%3D%3D) — HVNC operating payload
- [http://45.32.150\[.\]251/module/wrtc](http://45.32.150[.]251/module/wrtc) — SOCKS proxy WebRTC module
- [http://45.32.150\[.\]251:4787](http://45.32.150[.]251:4787) — WebSocket RAT channel (Socket.IO)
- [http://217.69.3\[.\]152/wall](http://217.69.3[.]152/wall) — Stage 2 exfiltration endpoint
- [http://217.69.3\[.\]152:80/log](http://217.69.3[.]152:80/log) — Stage 3 browser credential exfiltration endpoint
- [https://calendar.app\[.\]google/2NkrckKj4T6Dn4uK6](https://calendar.app[.]google/2NkrckKj4T6Dn4uK6) — Stage 3 URL indirection via Google Calendar Invite

Solana Wallets

- [BjVeAjPrSKFiingBn4vZvghsGj9KCE8AJVtbc9S8o8SC](#) — Stage 1 Unicode loader C2 dead-drop; Stage 3 DHT fallback
- [6YGcuyFRJKZtcaYCCFba9fScNUvPkGXodXE1mJiSzqDJ](#) — Stage 1 obfuscated preinstall loader C2 dead-drop
- [DSRUBTziADDHSik7WQvSMjvwCHFsbsthrbbjWmoJPuiw](#) — Browser extension C2 dead-drop

File Hashes (SHA-256)

- [06fab21dc276e3ab9b5d0a1532398979fd377b080c86d74f2c53a04603a43b1d](#) — Assaac.exe / SKuzyYcDD.exe (Ledger/Trezor phishing binary)
- [f171c383e21243ac85b5ee69821d16f10e8d718089a5c090c41efaaa42e81fca](#) — c_x64.node (browser credential stealer, Windows x64)
- [9df62cefd87784c7ee1ca8b4e6fc49737a90492fa6c23901e3b7981b18c6c988](#) — f_ex86.node (browser credential stealer, Windows x86)
- [43253a888417dfab034f781527e08fb58e929096cb4ef69456c3e13550cb4e9e](#) — data (Chrome App-Bound Encryption bypass)
- [4a60afa085fe5a847aef164578537bc33b9b58954143381e0c65c6354e4501e3](#) — index_ia32.node (HVNC module, Windows x86)
- [de81eacd045a88598f16680ce01bf99837b1d8170c7fc38a18747ef10e930776](#) — index_x64.node (HVNC module, Windows x64)
- [fdb5be3da2467e642bd8710f971e6b266b30ac15f5f413982fd719d7e0bffd9](#) — w.node (Chrome extension force-installer, Windows x64)
- [ee3e4dd5c1e073b8805f4107ccc7bc7e6e3c209fe13ea04ff3f2173c8dbe74a6](#) — m (Chrome extension force-installer, macOS universal binary)

Files Names

- [~/init.json](#) / [%USERPROFILE%\init.json](#) — Rate-limiting timestamp; contains uuid, version, date

- %TEMP%\hJxPxHP\ — Stage 2 credential staging directory
- %TEMP%\EUXFUxz0Ve\ — Stage 3 browser credential staging directory
- %TEMP%\SKuyzYcDD.exe — Dropped Ledger/Trezor phishing binary
- %APPDATA%\QtCvyfVWKH\index.js — Stage 3 main RAT script
- %LOCALAPPDATA%\QtCvyfVWKH\AghzgY.ps1 — Stage 3 PowerShell persistence launcher
- %APPDATA%_node_x86\node\node.exe — Silently downloaded Node.js v22.9.0 x86 runtime
- %APPDATA%_node_x64\node\node.exe — Silently downloaded Node.js v22.9.0 x64 runtime
- %APPDATA%_node_x64\webRTC\rtc-win32-x64\index.js — SOCKS proxy module
- %LOCALAPPDATA%\Google\Chrome\jucku\ — Malicious Chrome extension directory (Windows)
- /Library/Application Support/Google/Chrome/myextension/ — Malicious Chrome extension directory (macOS)

Registry

- HKCU\Software\Microsoft\Windows\CurrentVersion\Run\UpdateApp — Stage 3 RAT persistence
- HKCU\Software\Microsoft\Windows\CurrentVersion\Run\UpdateLedger → %TEMP%\SKuyzYcDD.exe — Ledger phishing binary persistence

Scheduled Tasks

UpdateApp — Runs AghzgY.ps1 (Stage 3) at startup with highest privileges

WMI

SELECT * FROM __InstanceCreationEvent WITHIN 2 WHERE TargetInstance ISA 'Win32_PnPEntity' — USB hardware wallet detection trigger (Ledger phishing binary)

Browser Extension

Display name: Google Docs Offline (version 1.95.1)

- Extension directory name (Windows): jucku
- Extension directory name (macOS): myextension

Process Names

Assaac — Ledger/Trezor phishing binary internal process name

Subscribe for threat news.
