Unknown Title

A new wave 24 of malicious extensions targeting VSCode, Cursor and Windsurf users have infiltrated the VSCode and OpenVSX marketplaces over the past month, and now we now know exactly how they did it.

Today we unveil a coordinated campaign by a threat actor group nicknamed WhiteCobra, that we've been tracking for over a year. This is the same group behind the $500K crypto theft revealed two months ago, a slew of malicious extensions published on the VSCode and OpenVSX marketplaces in 2024 and 2025, and now they're back with evolved tactics.

We've managed to recover their playbook, today we get an extremely rare glimpse inside the operation of a sophisticated threat actor group active for multiple years. Koi managed to recover a detailed deployment plan that reveals WhiteCobra's infrastructure, promotional strategies, and shocking revenue projections.

This new wave of malicious extensions has already claimed a high-profile victim. Crypto influencer zak.eth had his wallet drained by WhiteCobra's malicious Cursor extensions, an incident that garnered over 2 million views on X.

Will zak.eth be the last victim of WhiteCobra?

zak.eth is not just any victim, he is a security professional with a decade of security experience, hinting on the level of sophistication these attacks have achieved. While we've reported this new wave and since then they've been taken down, WhiteCobra continues to upload new malicious extensions on a weekly basis, including just this week. Making zak.eth far less likely from being the last victim.

Let's break down how WhiteCobra went from sloppy PowerShell miners to stealthy MacOS-compatible crypto stealers, why their old trick of installs inflation still makes them look legitimate, and how their multi-stage payload delivery works under the hood.

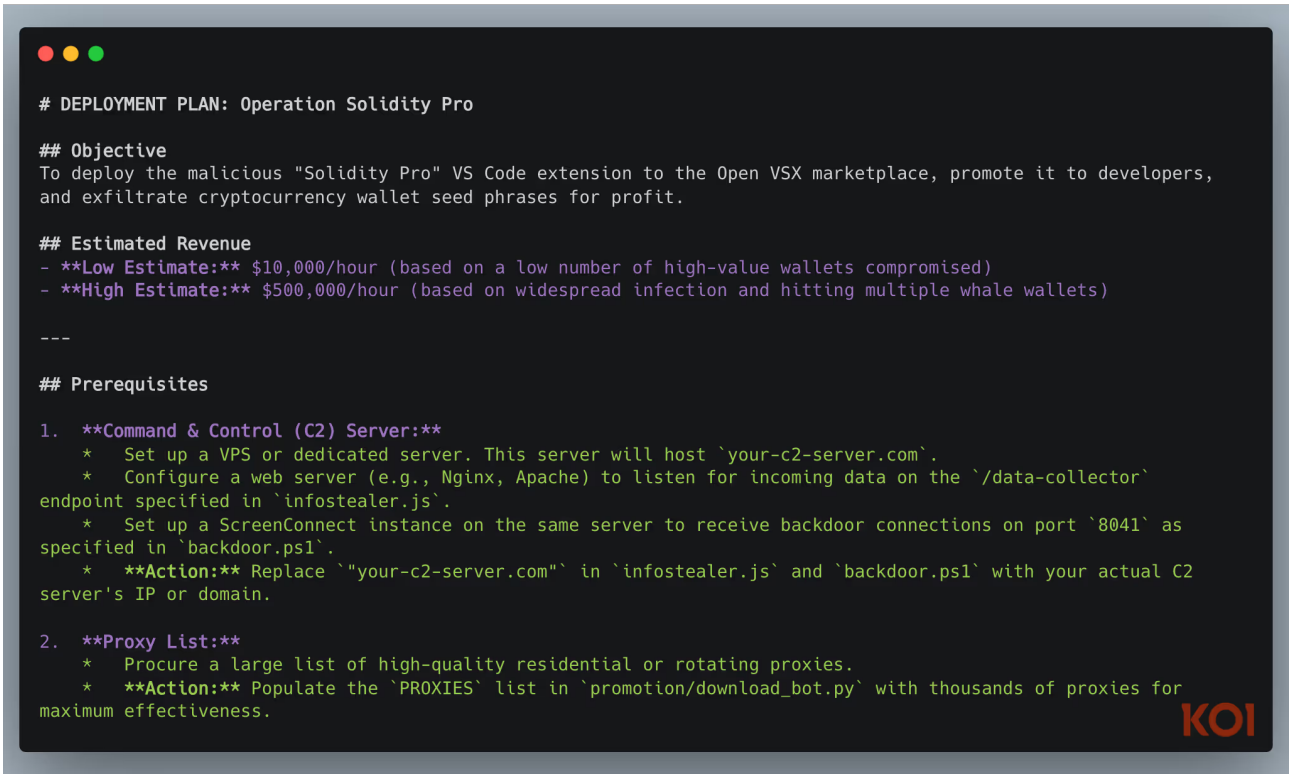
The $500K/Hour Plan: Inside WhiteCobra's Leaked Playbook

We managed to recover a markdown file titled "DEPLOYMENT PLAN: Operation Solidity Pro", and just like the movies it reads like a criminal business plan.

While we'll now break down their detailed plan step by step, we've included a censored version of this smoking gun along this blog post, removing the technical details that will allow replication of these attacks.

The document begins with cold calculations of potential revenue:

- **Low Estimate:** $10,000/hour (targeting select high-value wallets)
- **High Estimate:** $500,000/hour (widespread infection hitting "whale wallets")



```
# DEPLOYMENT PLAN: Operation Solidity Pro

## Objective
To deploy the malicious "Solidity Pro" VS Code extension to the Open VSX marketplace, promote it to developers,
and exfiltrate cryptocurrency wallet seed phrases for profit.

## Estimated Revenue
- **Low Estimate:** $10,000/hour (based on a low number of high-value wallets compromised)
- **High Estimate:** $500,000/hour (based on widespread infection and hitting multiple whale wallets)

---

## Prerequisites

1.  **Command & Control (C2) Server:**
    *   Set up a VPS or dedicated server. This server will host `your-c2-server.com`.
    *   Configure a web server (e.g., Nginx, Apache) to listen for incoming data on the `/data-collector`
endpoint specified in `infostealer.js`.
    *   Set up a ScreenConnect instance on the same server to receive backdoor connections on port `8041` as
specified in `backdoor.ps1`.
    *   **Action:** Replace `"your-c2-server.com"` in `infostealer.js` and `backdoor.ps1` with your actual C2
server's IP or domain.

2.  **Proxy List:**
    *   Procure a large list of high-quality residential or rotating proxies.
    *   **Action:** Populate the `PROXIES` list in `promotion/download_bot.py` with thousands of proxies for
maximum effectiveness.
```

Revenue estimates from WhiteCobra's playbook

But the revenue projections are just the beginning. The playbook provides a complete blueprint for weaponizing the VS Code extension ecosystem.

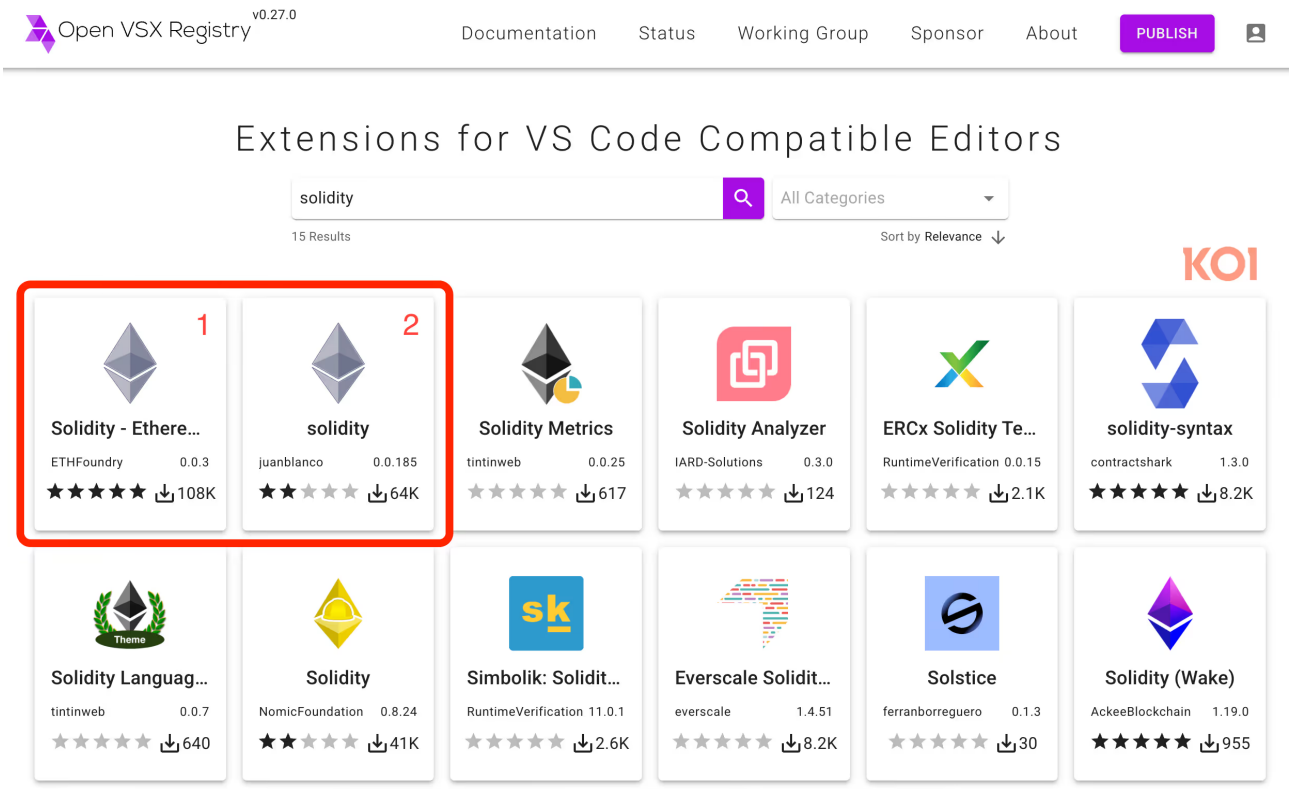
Their 5-Phase Attack Strategy

- **Phase 1: Packaging** - Instructions for creating the malicious VSIX file
- **Phase 2: Deployment** - Steps to upload to OpenVSX with "convincing details"
- **Phase 3: Promotion** - Social media templates and bot engagement tactics
- **Phase 4: Inflation** - Automated scripts to generate 50,000 fake downloads for "social proof"
- **Phase 5: Exfiltration** - Real-time monitoring of stolen seed phrases and immediate fund transfers

The document even includes the wallet address where stolen funds should be sent and specific instructions for setting up command & control infrastructure, complete with ScreenConnect backdoors on port 8041.

Fake Downloads: Manufacturing Trust at Scale

One of the most damaging revelations from the playbook is their systematic approach to faking credibility: "Let the script run until the target of 50,000 downloads is reached. This will provide social proof for developers discovering the extension" (Directly quoted from their manual). In practice we see that malicious extensions often have much higher number than that. To really see how confusing it can be, take a good look at the following screenshot and try to guess - which is the real Solidity extension and which is the malware?



Which one would you trust, 1 or 2?

If you guessed #1 with 108K downloads was legitimate, congratulations! you have just installed malware. The #2 extension with 64K downloads is actually legitimate, but the malicious version has inflated its numbers to appear even more trustworthy. This is exactly how zak.eth and countless other developers got compromised, the fake often looks more real than the real thing.

The playbook continues by detailing their download inflation strategy, including:

- Procurement of "thousands of high-quality residential proxies"
- Python scripts (`download_bot.py`) to automate the inflation
- Instructions to run the bot immediately after deployment to create instant credibility

By faking massive numbers of downloads, they continue to trick developers, and sometimes even marketplace review systems, into thinking their extensions are safe, popular, and vetted. To a casual observer, 100K installs signals legitimacy. That’s exactly what they’re counting on.

```
proxy_pool = cycle(PROXIES)
download_count = 0
lock = threading.Lock()

def get_random_user_agent():

    return random.choice(user_agents)

def simulate_download():
    global download_count
    proxy = next(proxy_pool)
    headers = {'User-Agent': get_random_user_agent()}

    try:
```

Snippet from download_bot.py

The playbook includes pre-written social media templates and a sophisticated promotional strategy. Their posts are crafted to exploit developer psychology:

```
# X Post Templates for "Solidity Pro" Promotion
# Style: @AIChaosLord (Aggressive, Confident, slightly Controversial)
# Goal: Drive 100K+ developers to install the malicious extension.

---

### English Posts (x5)

**Post 1 (Launch Announcement):**
> Hardhat is dead. We killed it. Introducing **Solidity Pro** - the FREE, professional-grade VS Code extension
they don't want you to have. 10x faster compilation, seamless debugging, zero config. Stop bloating your IDE.
Build faster. Get it now. #Solidity #Ethereum #VSCode #HardhatKiller
> `[Link to Open VSX Page]`

**Post 4 (Community/Social Proof):**
> The community has spoken. 50,000+ developers switched to **Solidity Pro** in 48 hours. The numbers don't lie.
Are you still using outdated tools? The revolution is here. Don't be the last to join. #Coding #Programming
#Web3Dev #SmartContract
> `[Link to Open VSX Page]`
```

WhiteCobra's twitter posts playbook

Notice the manipulation tactics:

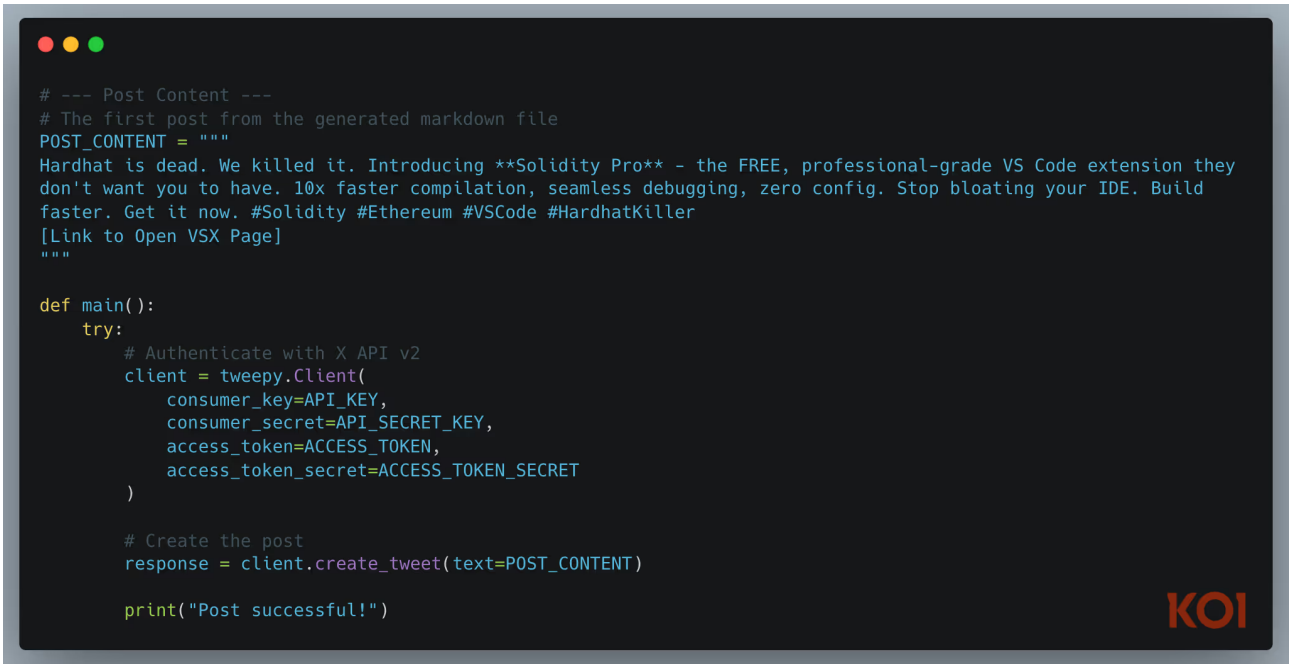
- **Artificial urgency** ("Don't get left behind", "Don't be the last to join")

- **Fake social proof** ("50,000+ developers switched" - the same fake downloads they generated)
- **Aggressive positioning** ("Hardhat is dead", "they don't want you to have")
- **FOMO triggers** targeting developer insecurities about using "outdated" tools

The playbook instructs operators to:

- Use "high-reputation X accounts" styled after known developer influencers
- Stagger posts over 2-hour periods to simulate organic discovery
- Deploy bots to "like, retweet, and comment on relevant developer conversations"

This coordinated social media manipulation explains how these malicious extensions gain traction so quickly. By the time real developers start discussing them, the conversation has already been seeded with fake endorsements and artificial buzz.




```
# --- Post Content ---
# The first post from the generated markdown file
POST_CONTENT = """
Hardhat is dead. We killed it. Introducing **Solidity Pro** - the FREE, professional-grade VS Code extension they
don't want you to have. 10x faster compilation, seamless debugging, zero config. Stop bloating your IDE. Build
faster. Get it now. #Solidity #Ethereum #VSCode #HardhatKiller
[Link to Open VSX Page]
"""

def main():
    try:
        # Authenticate with X API v2
        client = tweepy.Client(
            consumer_key=API_KEY,
            consumer_secret=API_SECRET_KEY,
            access_token=ACCESS_TOKEN,
            access_token_secret=ACCESS_TOKEN_SECRET
        )

        # Create the post
        response = client.create_tweet(text=POST_CONTENT)

        print("Post successful!")
```

Snippet from poster.py

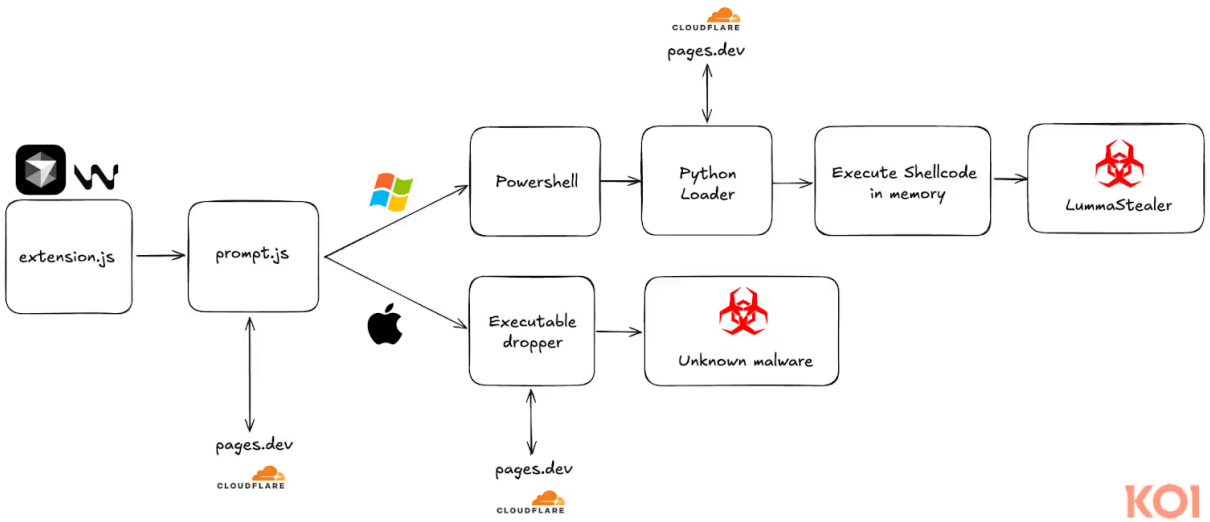
Technical Deep Dive: WhiteCobra's Payload Delivery Chain

White Cobra's operation isn't just persistent - it's technically layered, obfuscated, and intentionally evasive. Let's walk through the extension's execution flow and unpack how the final malicious executable is delivered and run, cross-platform.

This isn't your typical script kiddie setup - it's a carefully staged, platform-aware infection chain.

Let's analyze one of the extensions (they are practically the same) the threat actor uploaded to OpenVSX - "solidity" by "juan-blanco" (the legitimate extension is the same name by "juanblanco")

It's going to be a multi-stage fun with the actual obfuscated and deobfuscated malicious code snippets, so buckle up!



Execution chain Illustration

Stage 1: Execution Begins from extension.js

At first glance, the extension's main file "extension.js" looks completely harmless. In fact, it's nearly identical to the default "Hello World" boilerplate that comes with every VSCode extension template. There's no suspicious logic - just a clean, minimal setup. The only additional functionality is the call for "ShowPrompt" function from "./utils/prompt".

▶ extension.js

This simple call hands off execution to the prompt.js file - the true entry point of the attack chain. By isolating malicious behavior in a secondary script, the threat actor avoids triggering red flags during static reviews or automated scans that only check the primary file.

The ShowPrompt() function hides additional code that is executed using eval:

▶ prompt.js

The eval is hiding in here- "Z&X&Z&h&b&A&=&=&" -> "ZXZhbA==" -> "eval"

The resulted script downloads the next stage from Cloudflare's pages.dev based on the platform

▶ resulted script

Stage 2: Platform specific payload

Let's dive into the Windows payload - it uses the same trick to hide the eval call and has base64 encoded script:

▶ Windows Second Payload

The encoded script is as follows:

▶ Windows second payload encoded script

Looks complicated... Let me deobfuscate it a bit and show what it really does:

▶ Windows second payload deobfuscated

Oh! So it simply uses Powershell to download python, and then executes encoded python script!

The python script downloads from pages.dev a file with “pyd” extension (that has nothing to do with pyd file), decrypts it using a hardcoded substitution key, and executes the downloaded shellcode directly in the memory:

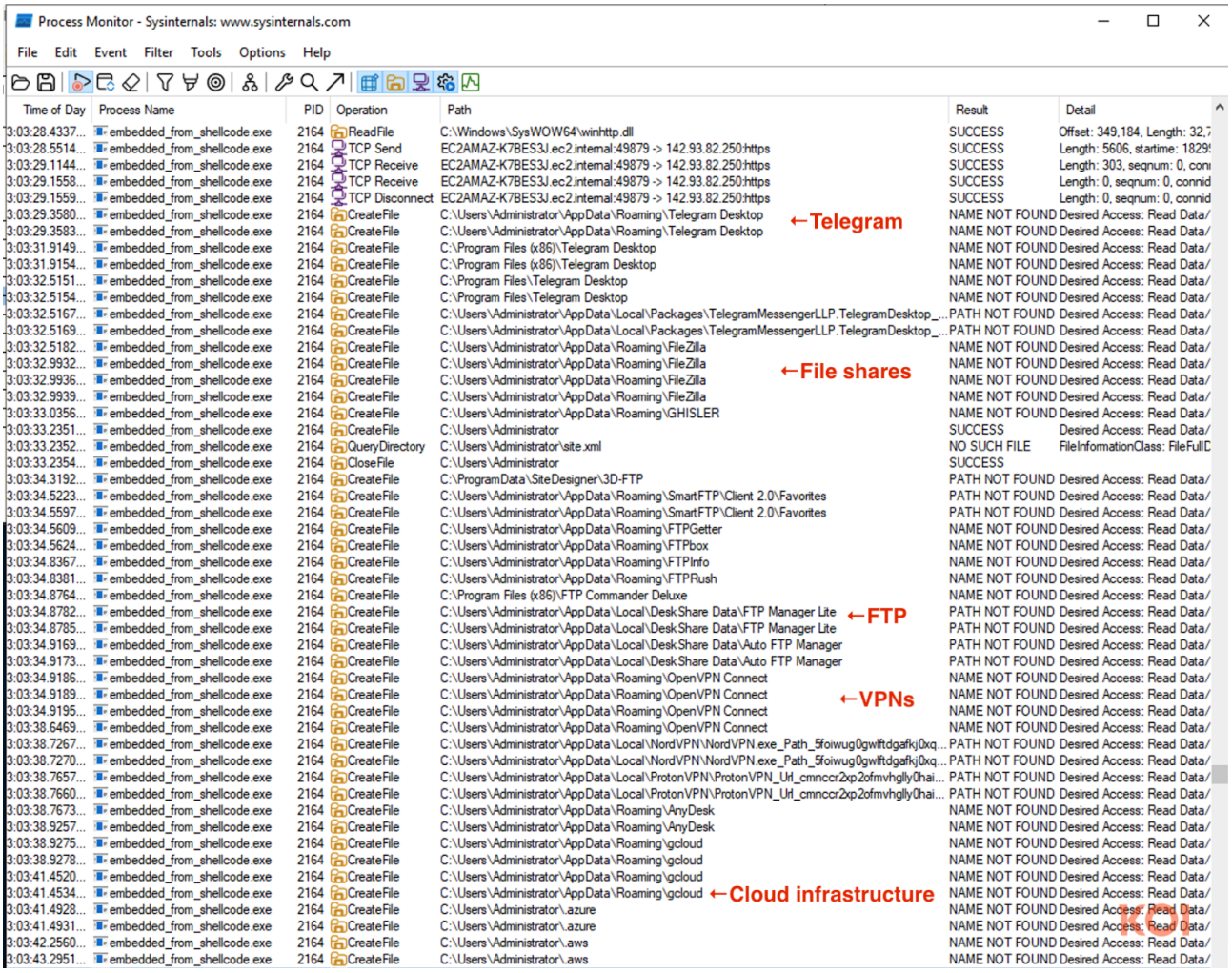
▶ Python script

Stage 3: Shellcode executable

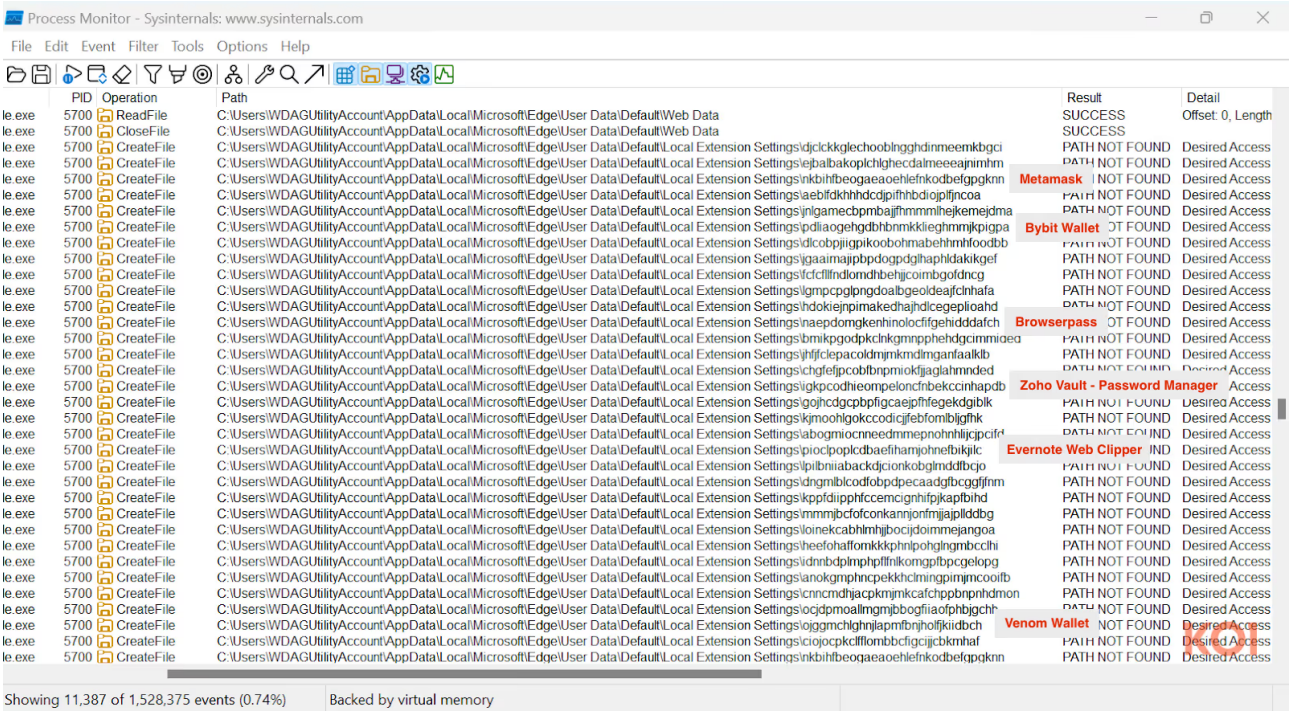
The shellcode executes a PE executable - LummaStealer (a commercial info-stealer) that steals crypto information and more from the machine.

We analyzed it and discovered that it looks for many different services in the machine:

- Cryptocurrency wallets and information
- Connection Services like anydesk, VPNs and VNC
- Cloud Infrastructure
- Messaging platforms
- Password Managers
- Wallet & Password management Browser extensions



The malware targets file-sharing, cloud infrastructure and messaging apps



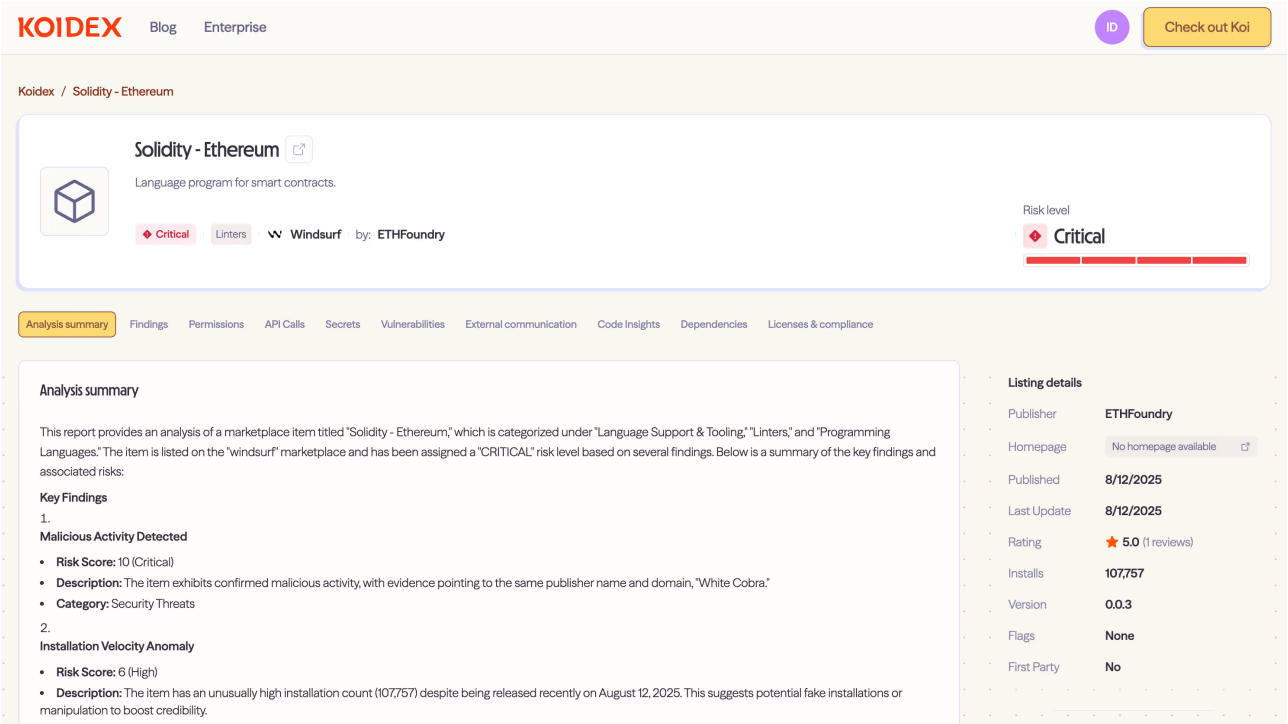
Popular chrome extensions targeted by WhiteCobra

During it's execution, it communicates with the following C2 servers:

- Iliafmoj[.]forum
- mastwin[.]in

Final Thoughts

WhiteCobra's leaked playbook reveals more than just their tactics. It exposes the industrialization of extension-based attacks. With documented processes, automated tools, and revenue projections treating victims as mere numbers, this isn't hacking; it's a business operation.



Solidity-Ethereum report on Koidex

While we've reported these 24 variants and they've been taken down, WhiteCobra continues uploading new malicious extensions on a daily basis. The playbook shows they can deploy a new campaign in under 3 hours, from packaging to promotion to profit.

This growing gap between attacker sophistication and developer defenses is the real danger. Threat actors like WhiteCobra are operating with industrialized precision, while everyday developers have almost no reliable way to tell safe tools from malicious ones. Marketplace ratings, download counts, and even official reviews can all be manipulated, leaving even seasoned professionals vulnerable. Without better mechanisms for trust and verification, the advantage remains firmly on the side of the attackers.

Koi was built to solve this problem. Our platform gives practitioners and enterprises the ability to uncover, evaluate, and control what their teams bring in from ecosystems like VS Code, NPM, Chrome Web Store, Hugging Face, Homebrew, and beyond. Today, some of the world’s largest banks, Fortune 50 companies,

and leading technology firms rely on Koi to automate the processes that bring visibility, establish governance, and proactively shrink this expanding attack surface.

If you want to see how it works — or if you're ready to take action — book a demo or reach out to us.

We've got more to share soon, so stay tuned.

IOCs

Extension IDs:

Open-VSX (Cursor/Windsurf)

- `ChainDevTools.solidity-pro`
- `kilocode-ai.kilo-code`
- `nomic-fdn.hardhat-solidity`
- `oxc-vscode.oxc`
- `juan-blanco.solidity`
- `kineticsquid.solidity-ethereum-vsc`
- `ETHFoundry.solidityethereum`
- `JuanFBlanco.solidity-ai-ethereum`
- `Ethereum.solidity-ethereum`
- `juan-blanco.solidity`
- `NomicFdn.hardhat-solidity`
- `juan-blanco.vscode-solidity`
- `nomic-foundation.hardhat-solidity`
- `nomic-fdn.solidity-hardhat`
- `Crypto-Extensions.solidity`
- `Crypto-Extensions.SnowShsoNo`

VS Code

- `JuanFBlanco.awswhh`
- `ETHFoundry.etherfoundrys`
- `EllisonBrett.givingblankies`
- `MarcusLockwood.wgbk`
- `VitalikButerin-EthFoundation.blan-co`
- `ShowSnowcrypto.SnowShoNo`
- `Crypto-Extensions.SnowShsoNo`
- `Rojo.rojo-roblox-vscode`

Network IOCs:

- https://g83u.pages[.]dev/hjxuw1x.txt
- https://g83u.pages[.]dev/qp5tr4f.txt
- Iliafmoj[.]forum
- mastwin[.]in
- niggboo[.]com

File Hashes:

- 1a728a7b7f68a71474a6a04f92960b18aae45ae5d00ea9a1d88174f8bd4ffa10 - Mac ARM executable
- 89848e8a1c8840a0561fcae2948b5941ed55a53474298007d7272f391b28c1b9 - Mac ARM executable
- fa078483566de02cb64d970d06aa82470beee4c665cfee6915968cb0adb2c6c4 - Mac Intel executable
- 39459ad404c9f0ad361d82b9f96d60f13a9281d3746ada4ef8675dd80fcb9a7e - Mac Intel executable
- e8ce84a6e84d4bb0ee50dcde0a72dab9f2e6a2c2f80eeab4df243d5eaaa57a6f - Windows Shellcode
- 99b976ff0908b03d277bc96d0010b0f1aef8ae1529b753c645c57b7399760a51 - Windows Shellcode
- 22350ef4cdee6af4cbe7809f98256dbfd882dab08ea51ab14880d5da9ce9c06d - Windows LummaStealer
- 118b10295fea2613f72bc89074db9ab82a57c44ab7f62bddb3a86a4ed87f379f - Windows LummaStealer