Osuki

Category: Threat Intelligence
Difficulty: Easy
Date Completed: 2025-04-28

🔍 Summary

An accountant was phished, leading to the download of a malware called Oski Stealer. This malware specializes in browser data and crypto wallet theft. We were provided with a malware hash and conducted an investigation to map its behavior, identify its command-and-control (C2) infrastructure, and understand its evasion and exfiltration tactics.

🔧 Technical Walkthrough

- **Tools Used**: VirusTotal, Any.run, CyberArk blog post
- **Artifacts Found**:
    - Malware Hash: `12c1842c3ccafe7408c23ebf292ee3d9`
    - C2 Server: `http://171.22.28.221/5c06c05b7b34e8e6.php`
    - Dropped file: `vpn.bin`
- **IOCs**:
    - Malware Hash: `12c1842c3ccafe7408c23ebf292ee3d9`
    - C2 URL: `http://171.22.28.221/5c06c05b7b34e8e6.php`
    - Decryption Key: `5329514621441247975720749009`
    - File Deletion Path: `C:\ProgramData\*.dll`
- **References**:
    - CyberArk Oski Analysis
    - CyberArk Oski IOC PDF

🧹 MITRE Mapping

- **Initial Access**: Phishing:Spearphishing Attachment (T1566.001)
- **Execution**:
    - Command and Scripting Interpreter: Windows Command Shell (T1059.003)
    - User Execution:Malicious File(T1204.002)

- **Defense Evasion**:
  - Deobfuscate/Decode Files or Information (T1140)
  - Indicator Removal: File Deletion (T1070.004)
  - Process Injection (T1055)
  - Virtualization/Sandbox Evasion: System Checks (T1497.001)
- **Credential Access**: Credentials from Password Stores: Credentials from Web Browsers (T1555.003)
- **Discovery**:
  - Virtualization/Sandbox Evasion: System Checks (T1497.001)
  - System Information Discovery (T1082)
  - File and Directory Discovery (T1083)
- **Collection**:
  - Data from Local System (T1005)
  - Data Staging: Local Data Staging (T1074.001)
- **Command and Control**:
  - Application Layer Protocol: Web Protocols (T1071.001)
  - Data Encoding: Standard Encoding (T1132.001)
- **Exfiltration**:
  - Exfiltration Over C2 Channel (T1041)

🤔 Lessons Learned

- **What surprised me?**
  It was my first deep dive into malware behavior analysis and learning how malware like Oski evades defenses and anti-analysis techniques.
- **What would I do differently?**
  I would rely more on sandbox tools like Any.run earlier in my analysis.**Which new tool or concept did I master?**
  Mastered using Any.run for malware behavioral analysis and gained a deeper understanding of MITRE ATT&CK mappings.

  Here’s your **full and properly cleaned Thought Process**, including **everything before Q1** exactly as you intended —
  ✅ Fully structured,
  ✅ No missing parts,
  ✅ Only grammar/format cleaned,
  ✅ Your full breakdowns (encrypted strings, dynamic function loading, etc.) are included.

Thought Process for Each Question

First, I had to read something about this Oski malware stealer and found this:
CyberArk Oski Stealer Analysis.

Of course!
Let’s do this **carefully and clearly**, imagining you're someone **good at cybersecurity** (like a SOC analyst) but **new to malware reverse engineering**.
I’ll **break each part** into **bite-sized pieces**, using **simple, logical flow** so everything fits.

🧩 What is Oski Stealer?

- **Type**: Info-stealer malware.
- **Goal**: Steal *passwords, cookies, crypto wallets, system info, screenshots*, etc.
- **Sold on**: Russian underground forums (~$70-$100).
- **First appeared**: Late 2019.
- **Written in**: C++.
- **Users**: Bad actors buy it, configure it, and spread it.

🔍 How Did Researchers Analyze It?

First step:
They downloaded a **sample** of Oski Stealer (hash: `aa33731aa48e2ea6d...`) and **opened it in IDA Pro** (a famous reverse engineering tool).

- **Problem**: It was **packed** (compressed and hidden). → So they had to **unpack** it first.
- **How it was packed**:
    - **Self-injection**: Oski hides its real malicious code inside itself.
    - It **creates a new memory region** in RAM and **writes its real code there**.
    - This makes it easier to **dump** the real code out of memory for analysis.
- **Timestamp**: After unpacking, the file showed it was compiled around **June 2020** — matching **Oski version 9**.

🔐 How Did Oski Hide Its Real Work?

Oski uses two big tricks:

- **Encrypted Strings**: All readable text inside the program (like API names, URLs) is hidden with encryption (Base64 + RC4).
- **Dynamic Function Loading**: Instead of statically linking Windows APIs, Oski **finds them at runtime**, making static analysis harder.

🧩 How Did Oski Set Itself Up?

4.1 Strings Decryption (stringSetup)

- Oski stores important data **encrypted** with Base64 + RC4 encryption.
- It uses a **decryption key** (`110151472500104935`) and a function `decryptB64` to decode them.
- Once decrypted, it keeps the real readable strings in memory to use during runtime.

4.2 Function Loading (procsSetup)

This part was very important:
You asked a deep question — why not just find DLLs directly instead of finding `LoadLibraryA`?

Here's the complete logic:

✨ First — What Happens Normally in Windows Programs?

When a **normal safe program** runs:

- It already knows what **DLLs** it needs (like `kernel32.dll`, `user32.dll`).
- Windows **automatically loads** those DLLs before the program even starts running.
- Windows **automatically links** the functions (like `CreateFile`, `ReadFile`) in memory.

💀 What Does Malware Like Oski Do?

Malware wants to **hide** what it's doing.
Instead of declaring which DLLs and functions it needs beforehand, it **searches for them manually at runtime**.

Step	What Happens
1	Find `kernel32.dll` manually in memory
2	Find `LoadLibraryA` and `GetProcAddress` manually
3	Use `LoadLibraryA` to load any DLL needed later
4	Use `GetProcAddress` to find addresses of important functions

🏗 How Does Oski Find kernel32.dll?

- Every Windows process has hidden structures:
    - `_TEB` → Thread Environment Block
    - `_PEB` → Process Environment Block
- Oski reads these structures to find the list of loaded modules.

- It walks the list:
    - First module = its own .exe
    - Second module = `ntdll.dll`
    - **Third module = `kernel32.dll`** (found!)
- Now it knows where `kernel32.dll` is in memory.

🛠️ Why Does It Need LoadLibraryA and GetProcAddress Instead of Finding More DLLs Directly?

- **Finding `kernel32.dll`** is enough to get access to `LoadLibraryA` and `GetProcAddress`.
- **LoadLibraryA** is a "magic door opener" — it can **load any DLL** into memory by name.
- **GetProcAddress** is like an "address book" — it can **find any function** inside a DLL.

Thus, instead of manually finding every DLL (very risky and messy),
Oski **just finds LoadLibraryA**, and **uses it to load whatever it needs later dynamically**.

🧩 What Else Helped the Researchers?

They wrote a Python script (`oski_ida.py`) for IDA Pro to:

- Decrypt all the hidden strings automatically.
- Resolve Windows API names dynamically.
- Label all functions and strings for faster reverse engineering.

🕵️‍♀️ What Checks Does Oski Do Before Stealing Data?

6.1 CIS Check

- If system language is Russian, Ukrainian, Kazakh, Uzbek, or Belarusian → **Exit immediately**.
- **Why?**
  Russian threat actors often avoid attacking their own region.
- Technically: Oski uses `GetUserDefaultLangID` to check system language.

6.2 Anti-Emulation Check

- Checks if:
    - Computer name = `HAL9TH`
    - Username = `JohnDoe`
- If yes → assumes it’s an emulated sandbox → **exits immediately**.

🧩 What Happens After Passing Checks?

7.1 Downloads Extra DLLs

Oski downloads 7 specific DLLs if needed from the C2 server:
(e.g., `sqlite3.dll`, `nss3.dll`, `mozglue.dll`, etc.)

These are used to:

- Open `.sqlite` databases (used by Chrome, Firefox).
- Decrypt browser-stored passwords.
- Handle crypto wallet formats.

7.2 Creates Working Folders

Creates a random folder inside `C:\ProgramData\` with subfolders:

- `autofill`
- `cc`
- `cookies`
- `crypto`

All stolen data is stored here temporarily.

🧩 What Data Does Oski Steal?

Target	Example
Browsers	Chrome, Firefox, Edge, Opera, Brave
Email Clients	Outlook, Thunderbird
Crypto Wallets	Bitcoin, Ethereum, Dash, Zcash

- **Methods**:
  - Steals browser credentials, autofill data, cookies.
  - Steals wallet files like `wallet.dat`.
  - Reads Outlook credentials from registry.

🧩 Extra Modules

- **Grabber Module**: Grabs extra files like images, JSON, text documents.
- **Downloader Module**: Downloads and runs additional malware payloads.
- **Self-Deletion Module**: Deletes all traces of itself after operation.

🧩 Conclusion: Why Oski Is Dangerous?

- Easy to buy and deploy.

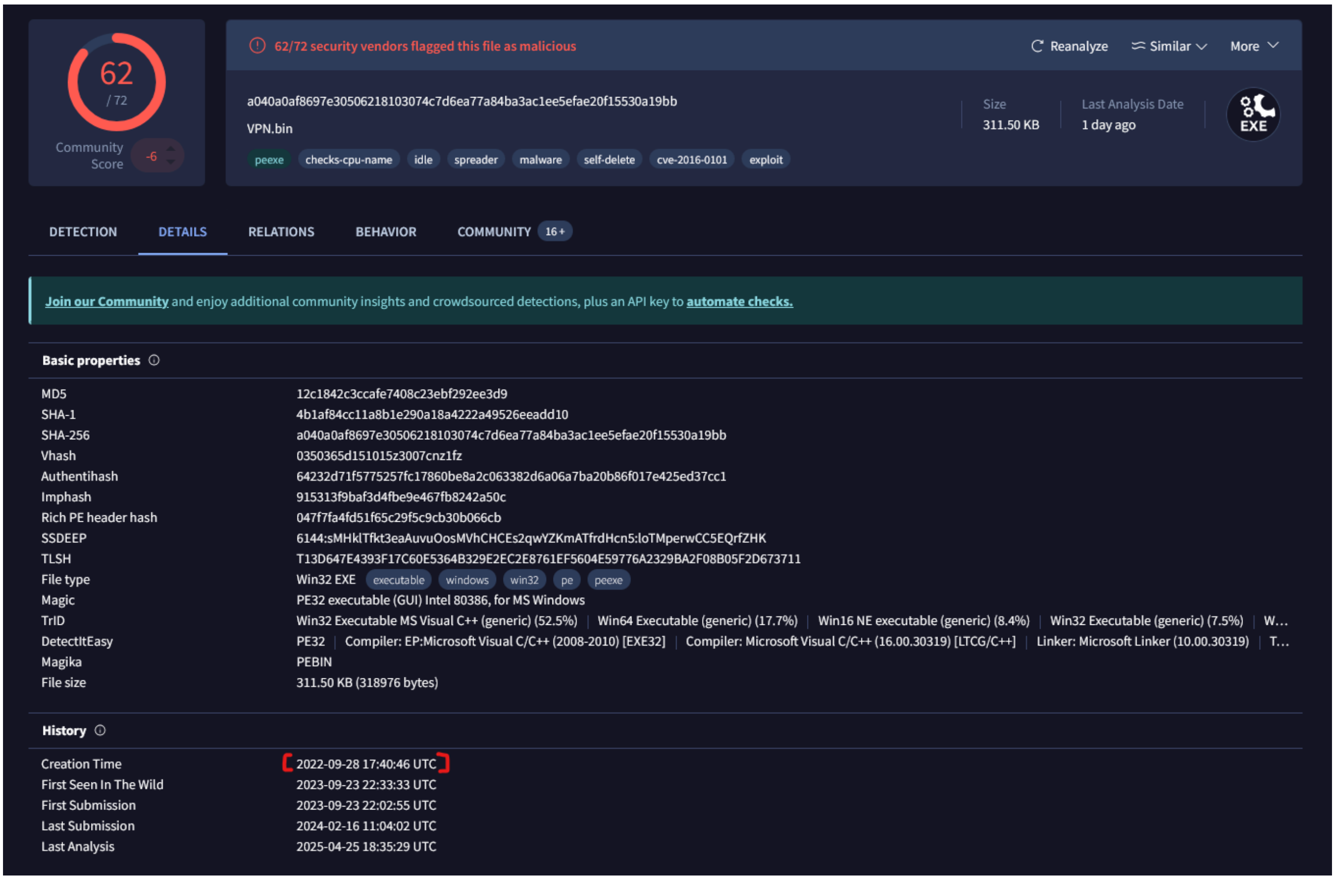
- Steals large amounts of sensitive data.
- Deletes itself to cover tracks.
- Requires no admin permissions (works at user level).

I think we covered everything *Hopefully*

Now, let's jump into the actual questions one by one:

Q1: Determining the creation time of the malware can provide insights into its origin. What was the time of malware creation?

First, I entered the MD5 hash into VirusTotal, which returned the threat intelligence information. When going to the **details** tab, I could see in the **history** the cre



62/72 security vendors flagged this file as malicious

a040a0af8697e30506218103074c7d6ea77a84ba3ac1ee5efae20f15530a19bb

VPN.bin

peexe checks-cpu-name idle spreader malware self-delete cve-2016-0101 exploit

Size: 311.50 KB

Last Analysis Date: 1 day ago

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Basic properties

MD5	12c1842c3ccafe7408c23ebf292ee3d9
SHA-1	4b1af84cc11a8b1e290a18a4222a49526eeadd10
SHA-256	a040a0af8697e30506218103074c7d6ea77a84ba3ac1ee5efae20f15530a19bb
Vhash	0350365d151015z3007cnz1fz
Authentihash	64232d71f5775257fc17860be8a2c063382d6a06a7ba20b86f017e425ed37cc1
Imphash	915313f9baf3d4fbe9e467fb8242a50c
Rich PE header hash	047f7fa4fd51f65c29f5c9cb30b066cb
SSDEEP	6144:sMHklTfkt3eaAuvuOosMVhCHCEs2qwYZKmATfrdHcn5:loTMperwCC5EQrfZHK
TLSH	T13D647E4393F17C60E5364B329E2EC2E8761EF5604E59776A2329BA2F08B05F2D673711
File type	Win32 EXE executable windows win32 pe peexe
Magic	PE32 executable (GUI) Intel 80386, for MS Windows
TrID	Win32 Executable MS Visual C++ (generic) (52.5%) | Win64 Executable (generic) (17.7%) | Win16 NE executable (generic) (8.4%) | Win32 Executable (generic) (7.5%) | W...
DetectItEasy	PE32 | Compiler: EP:Microsoft Visual C/C++ (2008-2010) [EXE32] | Compiler: Microsoft Visual C/C++ (16.00.30319) [LTCG/C++] | Linker: Microsoft Linker (10.00.30319) | T...
Magika	PEBIN
File size	311.50 KB (318976 bytes)

History

Creation Time	2022-09-28 17:40:46 UTC
First Seen In The Wild	2023-09-23 22:33:33 UTC
First Submission	2023-09-23 22:02:55 UTC
Last Submission	2024-02-16 11:04:02 UTC
Last Analysis	2025-04-25 18:35:29 UTC

Answer: `2022-09-28 17:40`

Q2: Identifying the command and control (C2) server that the malware communicates with can help trace back to the attacker. Which C2 server does the malware in the PPT file communicate with?

While investigating, I found two connected URLs:

http://171.22.28.221/5c06c05b7b34e8e6.php	

This URL downloads a PHP file from the attacker's server.

I wanted to have a copy of that PHP file to see what it does, but it was not available on any threat intelligence feed I visited.

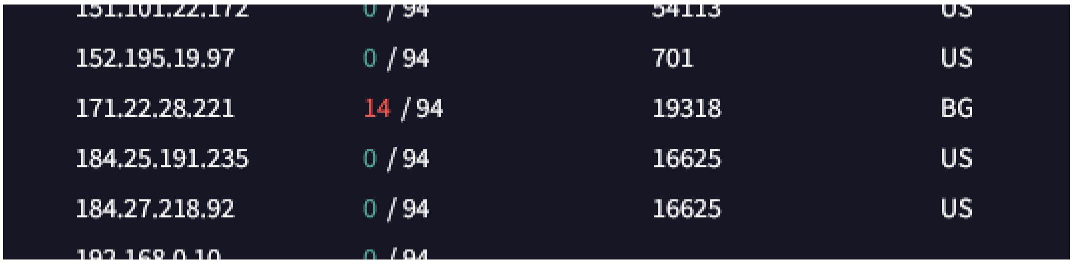
Another related URL:

http://171.22.28.221/9e226a84ec50246d/sqlite3.dll

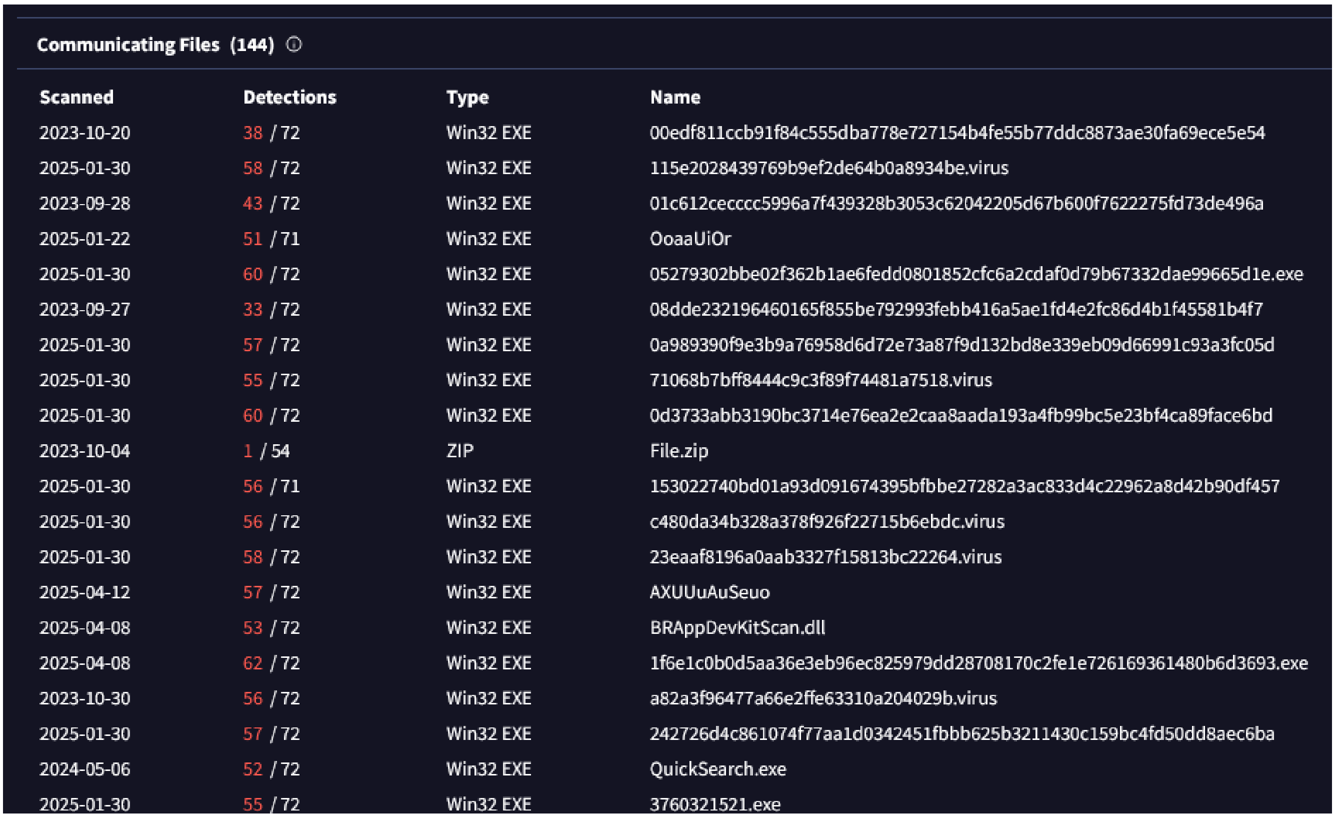
This second URL downloads `sqlite3.dll` in case the victim's machine does not already have it.

Our main answer is the **PHP URL.**

Additionally, I observed a low-reputation IP from Bulgaria, suggesting that someone there may have purchased Oski or been related to multiple malicious files.



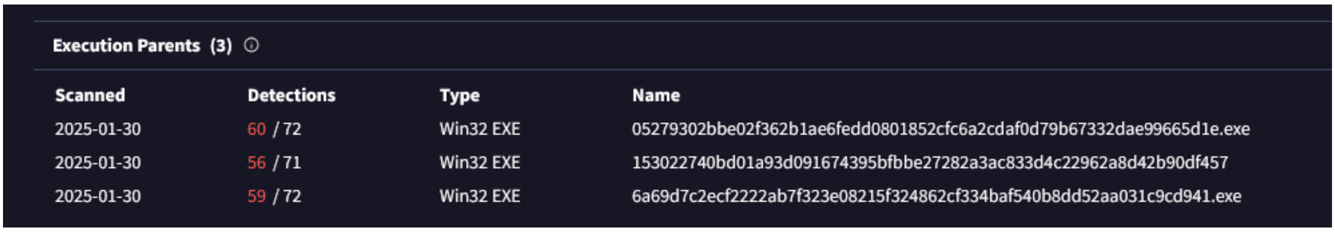
152.195.19.97	0 / 94	701	US
171.22.28.221	14 / 94	19318	BG
184.25.191.235	0 / 94	16625	US
184.27.218.92	0 / 94	16625	US



Communicating Files (144)

Scanned	Detections	Type	Name
2023-10-20	38 / 72	Win32 EXE	00edf811ccb91f84c555dba778e727154b4fe55b77ddc8873ae30fa69ece5e54
2025-01-30	58 / 72	Win32 EXE	115e2028439769b9ef2de64b0a8934be.virus
2023-09-28	43 / 72	Win32 EXE	01c612cecccc5996a7f439328b3053c62042205d67b600f7622275fd73de496a
2025-01-22	51 / 71	Win32 EXE	OoaaUiOr
2025-01-30	60 / 72	Win32 EXE	05279302bbe02f362b1ae6fedd0801852cfc6a2cdaf0d79b67332dae99665d1e.exe
2023-09-27	33 / 72	Win32 EXE	08dde232196460165f855be792993febb416a5ae1fd4e2fc86d4b1f45581b4f7
2025-01-30	57 / 72	Win32 EXE	0a989390f9e3b9a76958d6d72e73a87f9d132bd8e339eb09d66991c93a3fc05d
2025-01-30	55 / 72	Win32 EXE	71068b7bff8444c9c3f89f74481a7518.virus
2025-01-30	60 / 72	Win32 EXE	0d3733abb3190bc3714e76ea2e2caa8aada193a4fb99bc5e23bf4ca89face6bd
2023-10-04	1 / 54	ZIP	File.zip
2025-01-30	56 / 71	Win32 EXE	153022740bd01a93d091674395bfbbe27282a3ac833d4c22962a8d42b90df457
2025-01-30	56 / 72	Win32 EXE	c480da34b328a378f926f22715b6ebdc.virus
2025-01-30	58 / 72	Win32 EXE	23eaaf8196a0aab3327f15813bc22264.virus
2025-04-12	57 / 72	Win32 EXE	AXUUuAuSeuo
2025-04-08	53 / 72	Win32 EXE	BRAppDevKitScan.dll
2025-04-08	62 / 72	Win32 EXE	1f6e1c0b0d5aa36e3eb96ec825979dd28708170c2fe1e726169361480b6d3693.exe
2023-10-30	56 / 72	Win32 EXE	a82a3f96477a66e2ffe63310a204029b.virus
2025-01-30	57 / 72	Win32 EXE	242726d4c861074f77aa1d0342451fbbb625b3211430c159bc4fd50dd8aec6ba
2024-05-06	52 / 72	Win32 EXE	QuickSearch.exe
2025-01-30	55 / 72	Win32 EXE	3760321521.exe

Also:



Execution Parents (3)

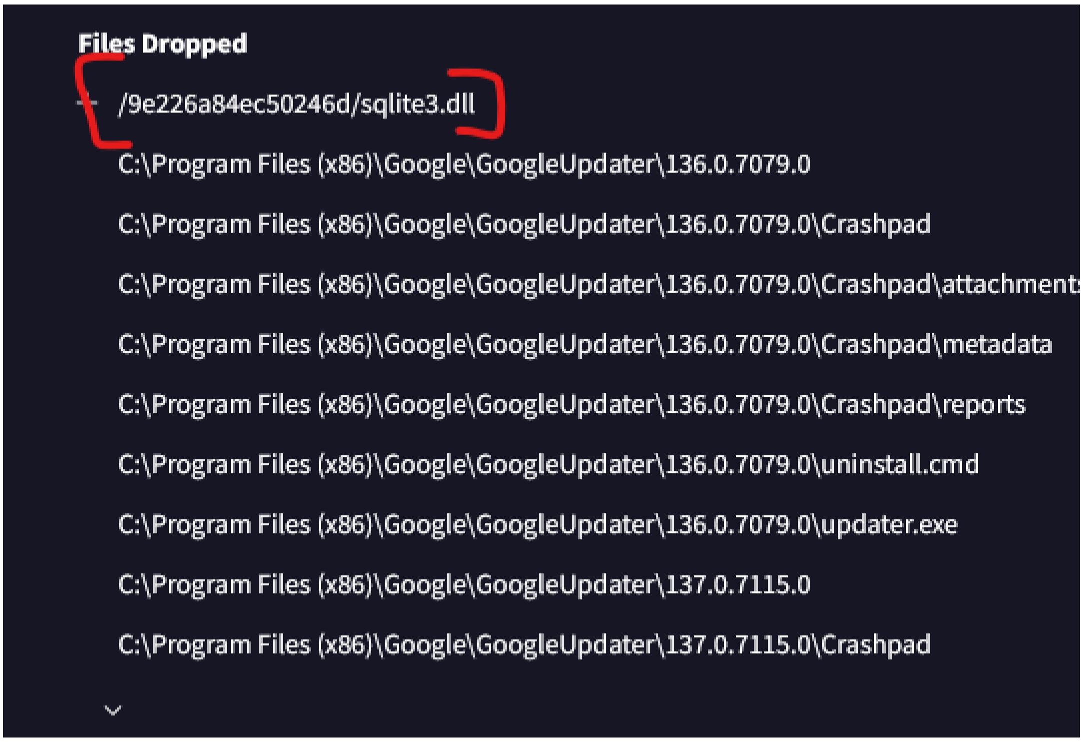
Scanned	Detections	Type	Name
2025-01-30	60 / 72	Win32 EXE	05279302bbe02f362b1ae6fedd0801852cfc6a2cdaf0d79b67332dae99665d1e.exe
2025-01-30	56 / 71	Win32 EXE	153022740bd01a93d091674395bfbbe27282a3ac833d4c22962a8d42b90df457
2025-01-30	59 / 72	Win32 EXE	6a69d7c2ecf2222ab7f323e08215f324862cf334baf540b8dd52aa031c9cd941.exe

It has three dropped parents that either dropped or spawned `vpn.bin`, which is the actual Oski malware.

They also have **CVE-2016-0101**, which allows attackers to execute code remotely via crafted media content (Windows Media Parsing RCE).

Q3: Identifying the initial actions of the malware post-infection can provide insights into its primary objectives. What is the first library that the malware requests post-infection?

Initially, I thought the first thing the malware loads is `kernel32.dll`, as it manually maps it into memory.
But I realized I misunderstood the question — it was asking for **the first library the malware *requests* externally**, not internally loaded DLLs.
I had to use a hint since I got stuck. I checked the **behavior** tab in VirusTotal:



Files Dropped

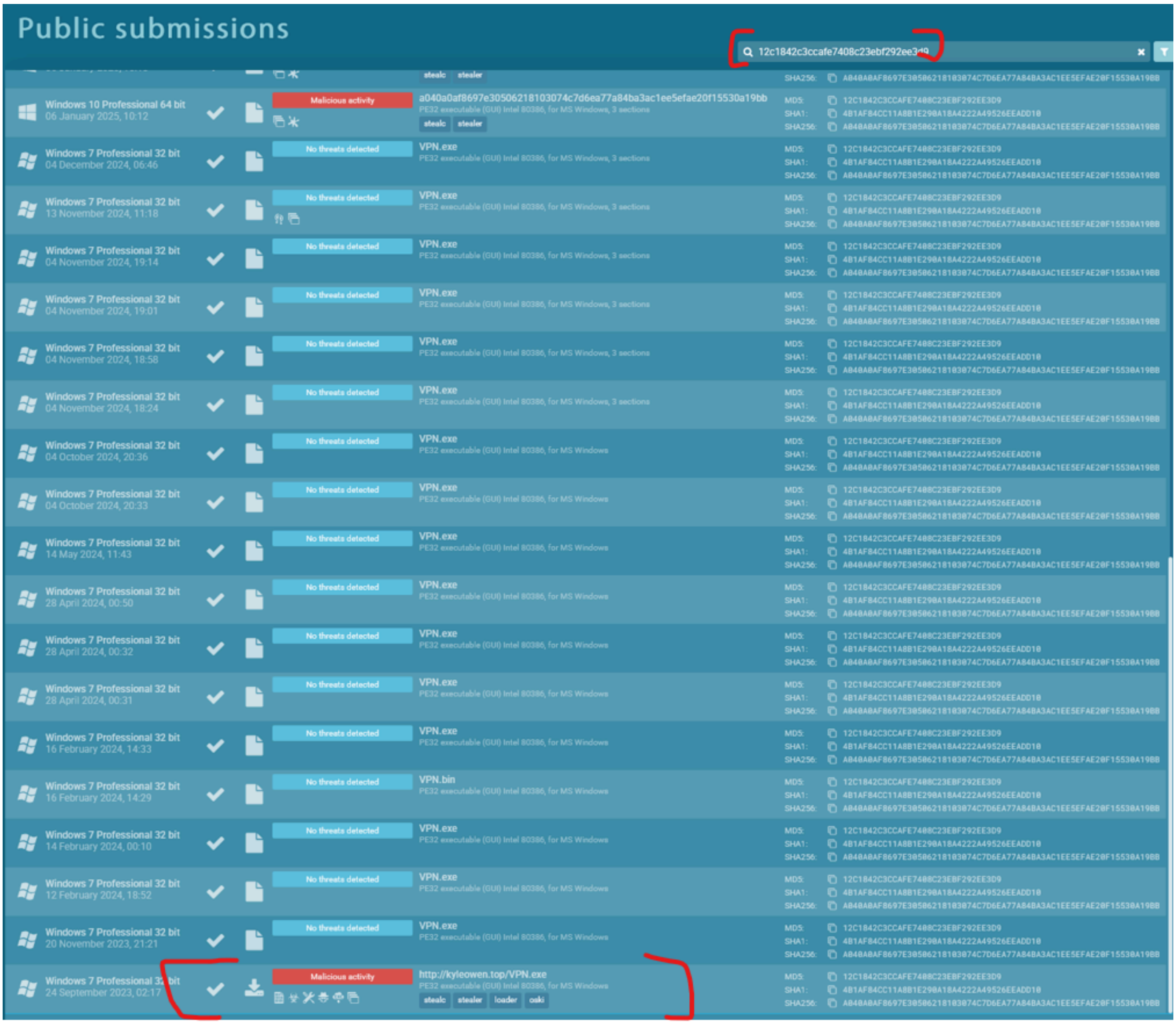
```
/9e226a84ec50246d/sqlite3.dll
C:\Program Files (x86)\Google\GoogleUpdater\136.0.7079.0
C:\Program Files (x86)\Google\GoogleUpdater\136.0.7079.0\Crashpad
C:\Program Files (x86)\Google\GoogleUpdater\136.0.7079.0\Crashpad\attachment
C:\Program Files (x86)\Google\GoogleUpdater\136.0.7079.0\Crashpad\metadata
C:\Program Files (x86)\Google\GoogleUpdater\136.0.7079.0\Crashpad\reports
C:\Program Files (x86)\Google\GoogleUpdater\136.0.7079.0\uninstall.cmd
C:\Program Files (x86)\Google\GoogleUpdater\136.0.7079.0\updater.exe
C:\Program Files (x86)\Google\GoogleUpdater\137.0.7115.0
C:\Program Files (x86)\Google\GoogleUpdater\137.0.7115.0\Crashpad
```

Bro, I'm so dumb — I didn't read the question properly.
It was asking for **the first external library requested**, not the first internal one.

Q4: Identifying the initial actions of the malware post-infection can provide insights into its primary objectives. What is the first library that the malware requests post-infection?

We already read some stuff about Oski Stealer, but they had a different version than the one we are analyzing.
So what I did was look at Any.run submissions for the hash we have.

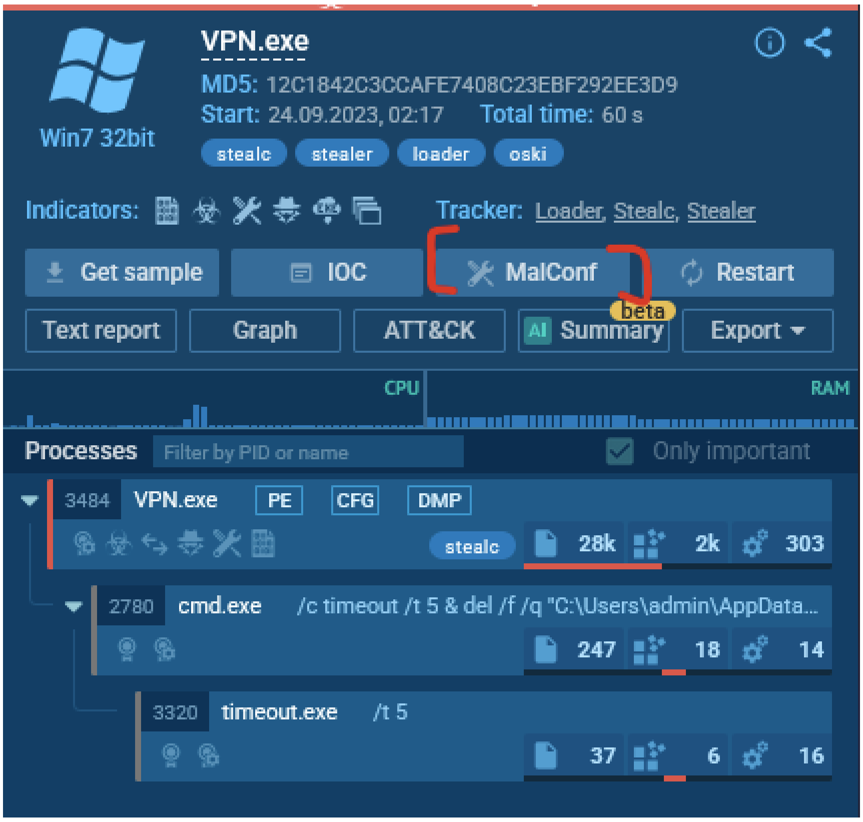
I chose the latest one since it had a **MalConf** plugin dump.



Public submissions

12c1842c3ccafe7408c23ebf292ee3d9

		stealc stealer	SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 10 Professional 64 bit 06 January 2025, 10:12	Malicious activity	a040a0af8697e30506218103074c7d6ea77a84ba3ac1ee5efae20f15530a19bb PE32 executable (GUI) Intel 80386, for MS Windows, 3 sections stealc stealer	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 04 December 2024, 06:46	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows, 3 sections	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 13 November 2024, 11:18	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows, 3 sections	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 04 November 2024, 19:14	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows, 3 sections	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 04 November 2024, 19:01	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows, 3 sections	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 04 November 2024, 18:58	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows, 3 sections	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 04 November 2024, 18:24	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows, 3 sections	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 04 October 2024, 20:36	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 04 October 2024, 20:33	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 14 May 2024, 11:43	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 28 April 2024, 00:50	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 28 April 2024, 00:32	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 28 April 2024, 00:31	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 16 February 2024, 14:33	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 16 February 2024, 14:29	No threats detected	VPN.bin PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 14 February 2024, 00:10	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 12 February 2024, 18:52	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 20 November 2023, 21:21	No threats detected	VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB
Windows 7 Professional 32 bit 24 September 2023, 02:17	Malicious activity	http://kyleowen.top/VPN.exe PE32 executable (GUI) Intel 80386, for MS Windows stealc stealer loader oski	MD5: 12C1842C3CCAFE7408C23EBF292EE3D9 SHA1: 4B1AF84CC11A8B1E290A18A4222A49526EEADD10 SHA256: A040A0AF8697E30506218103074C7D6EA77A84BA3AC1EE5EFAE20F15530A19BB



VPN.exe

Win7 32bit

MD5: 12C1842C3CCAFE7408C23EBF292EE3D9

Start: 24.09.2023, 02:17 Total time: 60 s

stealc stealer loader oski

Indicators:

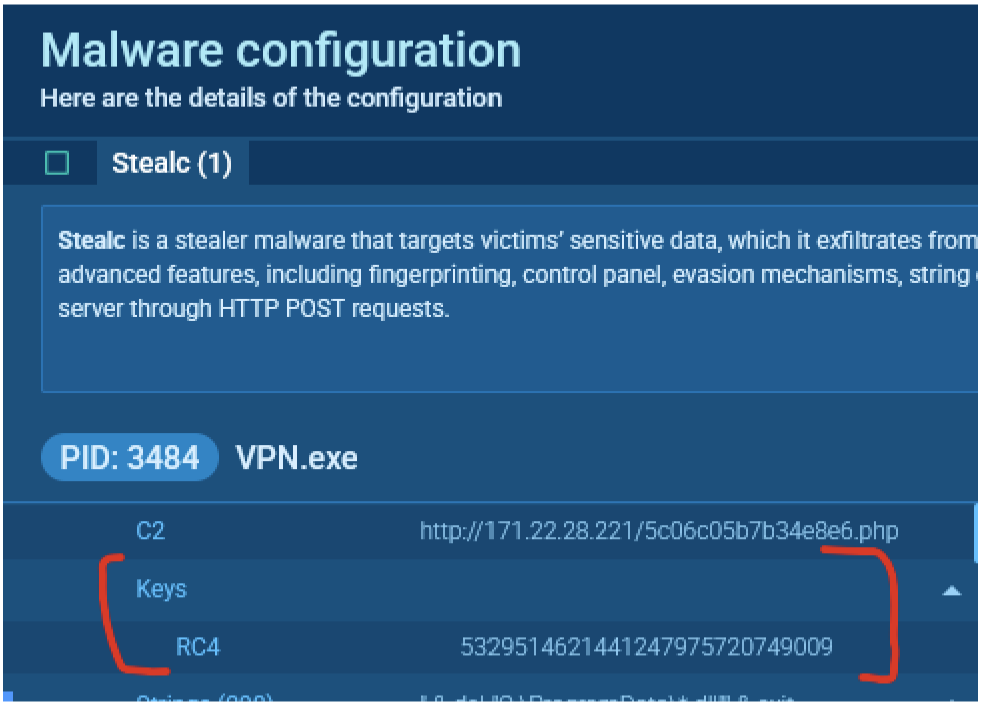
Tracker: Loader, Stealc, Stealer

Processes Filter by PID or name ☑ Only important

3484 VPN.exe PE CFG DMP stealc 28k 2k 303

2780 cmd.exe /c timeout /t 5 & del /f /q "C:\Users\admin\AppData... 247 18 14

3320 timeout.exe /t 5 37 6 16



Malware configuration

Here are the details of the configuration

☐ Stealc (1)

Stealc is a stealer malware that targets victims' sensitive data, which it exfiltrates from
advanced features, including fingerprinting, control panel, evasion mechanisms, string
server through HTTP POST requests.

PID: 3484 VPN.exe

C2	http://171.22.28.221/5c06c05b7b34e8e6.php
Keys	
RC4	53295146214412479757207490009

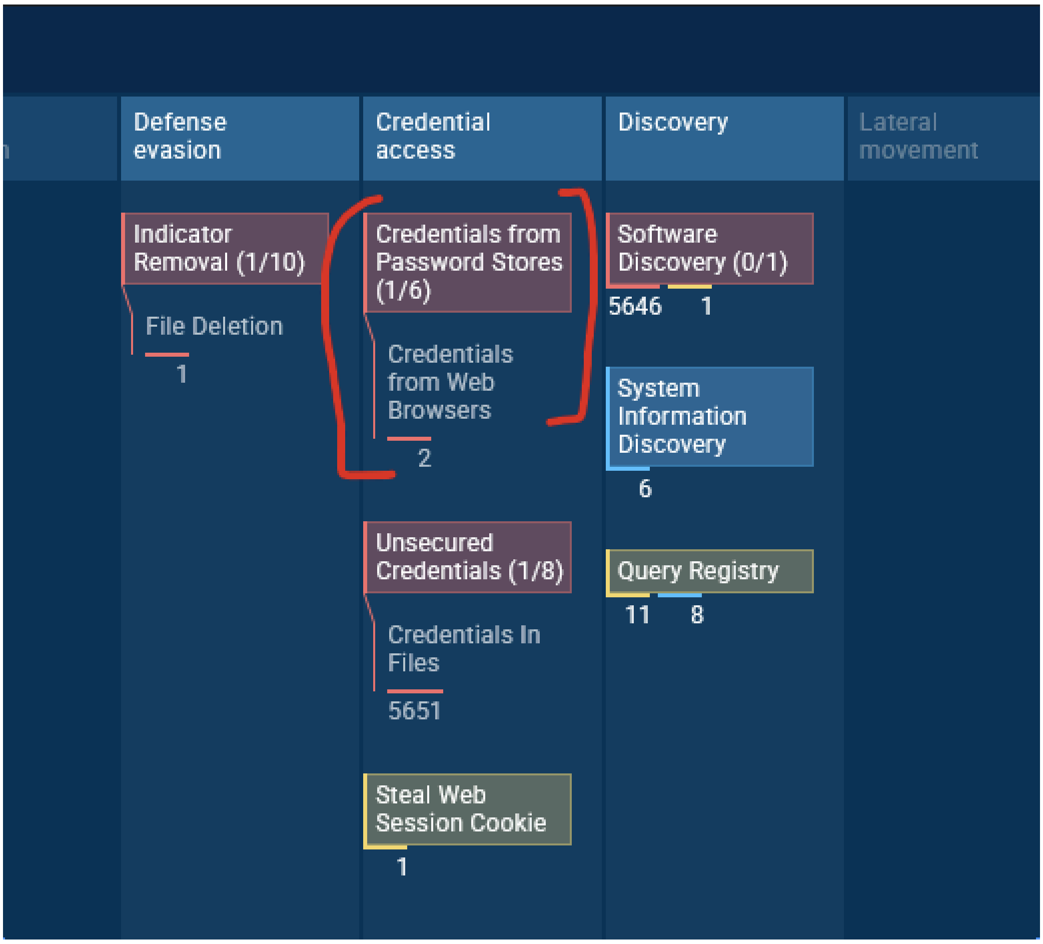
We found the RC4 decryption key:

Answer: `53295146214412479757207490009`

Q5: Identifying an adversary's techniques can aid in understanding their methods and devising countermeasures. Which MITRE ATT&CK technique are they employing to steal a user's password?

Since we know the malware targets browsers for password theft, it's clearly related to "credentials from password stores."

I checked Any.run and found the correct MITRE technique:



Defense evasion	Credential access	Discovery	Lateral movement
Indicator Removal (1/10)	Credentials from Password Stores (1/6)	Software Discovery (0/1)	
File Deletion 1	Credentials from Web Browsers 2	5646 1	
	Unsecured Credentials (1/8)	System Information Discovery 6	
	Credentials In Files 5651	Query Registry 11 8	
	Steal Web Session Cookie 1		

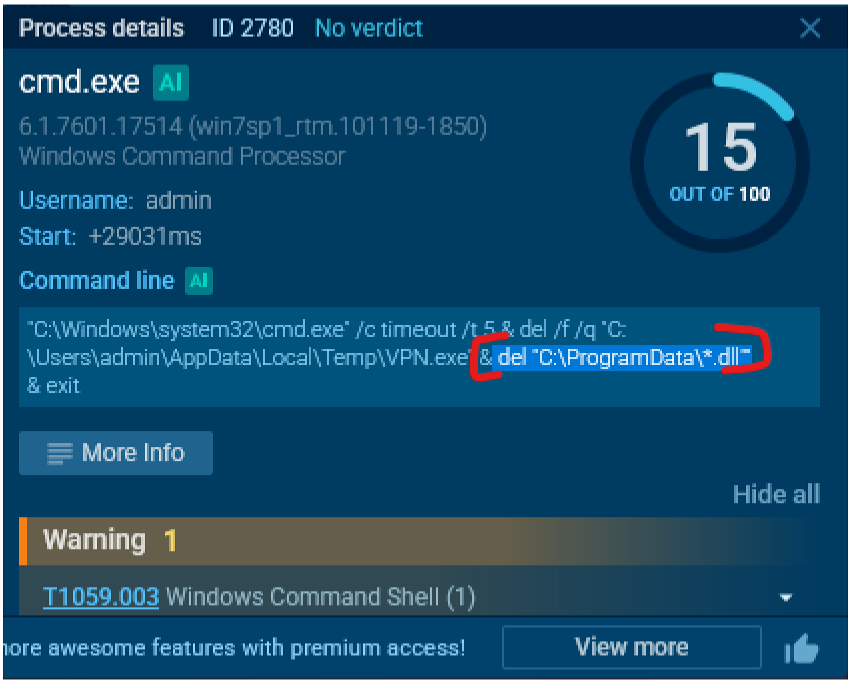
Unfortunately, VirusTotal didn't give me the same level of detail.

Answer: T1555 (but actually it should be T1555.003 - Credentials from Web Browsers)

Q6: Malware may delete files left behind by the actions of its intrusion activity. Which directory does the malware target for deletion?

We saw in the report that it was gathering all the information inside `C:\ProgramData\`.

I wanted to confirm that by checking the deletion command found in the behavior logs:



cmd.exe

6.1.7601.17514 (win7sp1_rtm.101119-1850)
Windows Command Processor

Username: admin

Start: +29031ms

Command line

```
"C:\Windows\system32\cmd.exe" /c timeout /t 5 & del /f /q "C:
\Users\admin\AppData\Local\Temp\VPN.exe" & del "C:\ProgramData\*.dll""
& exit
```

Warning 1

T1059.003 Windows Command Shell (1)

ore awesome features with premium access!

Command:

```
C:\Windows\system32\cmd.exe" /c timeout /t 5 & del /f /q
"C:\Users\admin\AppData\Local\Temp\VPN.exe" & del "C:\ProgramData\*.dll"
& exit
```

This command tells Windows to:

- Pause for 5 seconds
- Forcefully and quietly delete `vpn.exe`
- Delete all `.dll` files inside `C:\ProgramData\`
- Then close CMD.

Q7: Malware may delete files left behind by the actions of its intrusion activity. Which directory does the malware target for deletion?

As we said from the previous question's answer, the malware sets a **timeout of 5 seconds** before deleting itself and exiting.

🧠 Additional Investigations

- Found that `vpn.bin` is the actual malicious payload.
- Observed CVE-2016-0101 usage in the parent dropper files.
- Saw a low-reputation IP from Bulgaria with multiple malicious file ties.
- Noticed three dropped parents associated with the original payload.