Xehook Stealer: Cinoshi’s Crypto & 2FA Target Evolution

Published: 2024-03-12 · Archived: 2026-04-05 20:27:29 UTC

Xehook Stealer: Evolution of Cinoshi’s Project Targeting Over 100 Cryptocurrencies and 2FA Extensions

Xehook Stealer: Evolution of Cinoshi’s Project Targeting Over 100 Cryptocurrencies and 2FA Extensions

CRIL analyzes Xehook stealer and it's evolution from Cinoshi project.

Key Takeaways

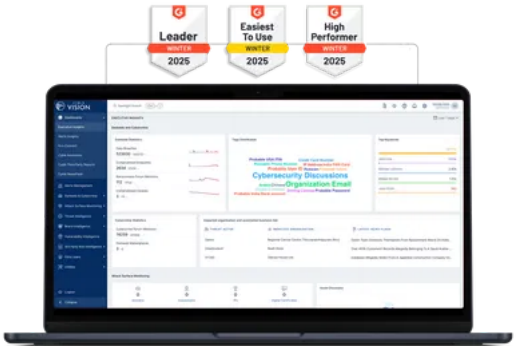
- Xehook Stealer, discovered by CRIL in January 2024, is a .NET-based malware targeting Windows operating systems.
- The Stealer boasts dynamic data collection capabilities from Chromium and Gecko-based browsers, supporting over 110 cryptocurrencies and 2FA extensions. It also includes an API for creating custom traffic bots and a feature for recovering dead Google cookies.
- CRIL investigation reveals a potential connection between Xehook Stealer, Agniane, and the Cinoshi project.
- The sequence of events suggests a progression from the free MaaS Cinoshi Project to the emergence of Agniane Stealer and, eventually, Xehook Stealer, indicating possible rebranding and development iterations.
- SmokeLoader binaries have been identified as a common vector for distributing Xehook Stealer, indicating active propagation efforts.
- Xehook Stealer shares significant code overlaps with Agniane Stealer, suggesting an evolutionary relationship between the two. Configuration data similarities and communication with the same C&C server reinforce this connection.
- Similarities in web panel design between Cinoshi, Agniane, and Xehook Stealer panels further support the notion of continuous development and iteration.

Overview

CRIL found a new stealer named Xehook in January 2024. Xehook Stealer targets the Windows operating system and is coded in the .Net programming language. The Threat Actor (TA) claims this stealer offers dynamic data collection from all Chromium and Gecko-based browsers, supporting over 110 cryptocurrencies and 2FA extensions.

The TA behind this stealer also mentioned that it includes customizable build settings, seamless integration with Telegram for real-time notifications, and the ability to send logs directly to Telegram. Additionally, Xehook Stealer provides an API for creating custom traffic bots and includes a feature for recovering dead Google cookies.

The TA claimed that this stealer gathers a wide range of data, including passwords, cookies, autofill information, and credit cards from browsers, alongside sessions from messaging platforms like Telegram and Discord. It supports over 15 desktop cryptocurrency wallets and includes a recursive file grabber for collecting specific file formats from user directories.



Xehook Stealer is sold on a subscription model, which is available on a monthly, quarterly, and semi-annual basis, with prices ranging from $50 for one month to $600 for an unlimited period. An additional $100 provides access to the API for an indefinite duration, ensuring comprehensive support and functionality for subscribers.

The figure below shows the Xehook stealer post on a cybercrime forum.

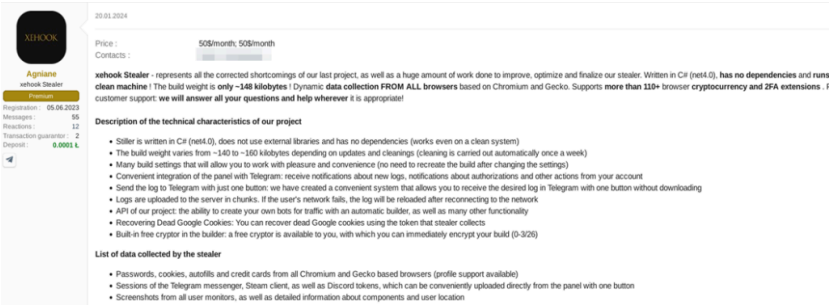


Figure 1 – Xehook Stealer Post on a Cybercrime Forum

Notably, when this post about Xehook Stealer was made on a cybercrime forum, the TA's username was "thx4drugs," which was subsequently changed to "Agniane," as indicated in the figure below.

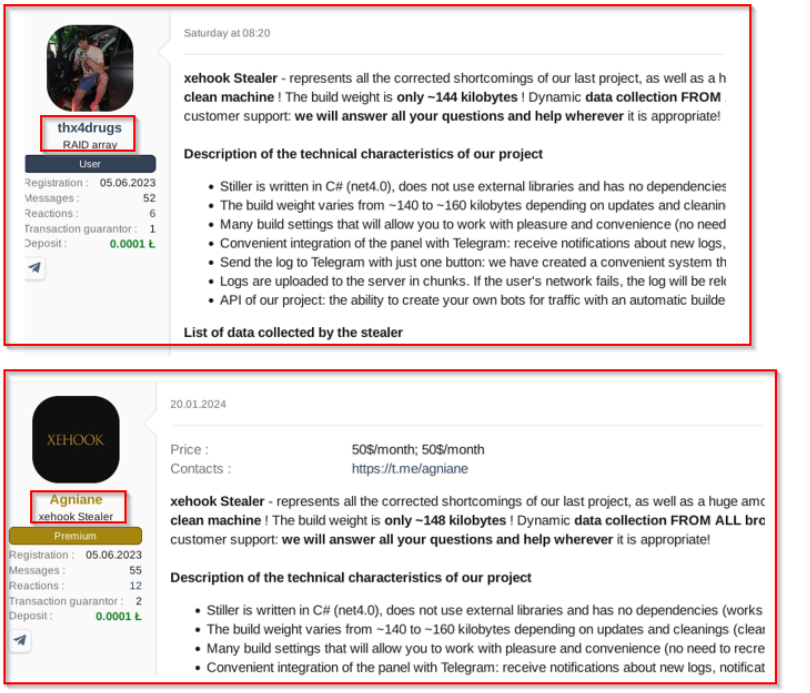


Figure 2 – TAs Renamed Account

Upon further investigation, we discovered the emergence of a stealer named Agniane in August 2023, as reported by Zscaler. Notably, the Telegram handle mentioned in the Xehook stealer post corresponds to the one utilized by the Telegram bot associated with Agniane stealer. Interestingly, Agniane stealer is believed to have connections with the Cinoshi project, which CRIL initially uncovered in March 2023. This project operated under a Malware-as-a-Service (MaaS) model, offering a stealer and web panel for free upon its launch. The following sequence of events unveils the connection between these projects:

- A TA launched the free MaaS Cinoshi Project in March 2023.
- Agniane Stealer emerged in August 2023.
- Agniane Stealer references Cinoshi in its note.
- The Telegram bot used by Agniane Stealer mentions the TA's handle, Agniane, in its bio.
- A TA named thx4drugs posts about Xehook stealer on a cybercrime forum.
- The Xehook stealer post also mentions the same Telegram handle, "Agnianne," as the Agniane stealer bot mentioned.
- The account of thx4drugs is later renamed to Agniane.

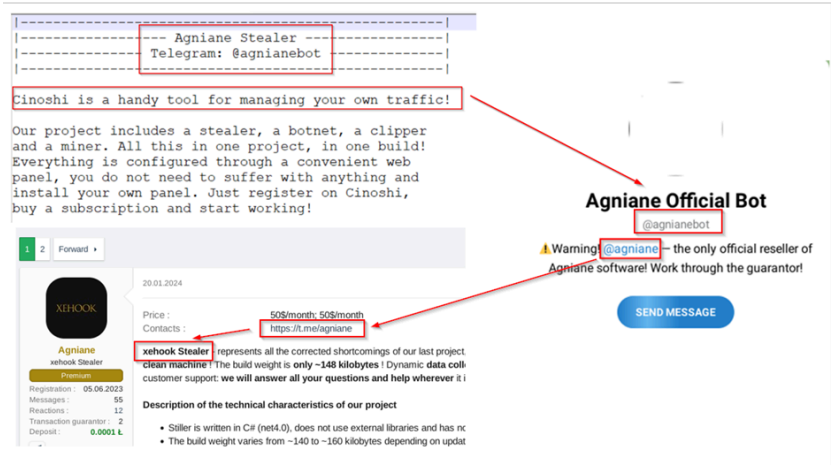


Figure 3 – Linking TA Profiles

There is a high chance that the TA launched the Cinoshi project as a free M-a-a-S model to gain a user base, and after enhancing the product, the TA started renaming it on each iteration and selling it. We also observed a lot of similarities between the Web panel utilized by the Cinoshi project, Agniane stealer, and Xehook stealer, such as the same Font scheme and Structure of the panel.

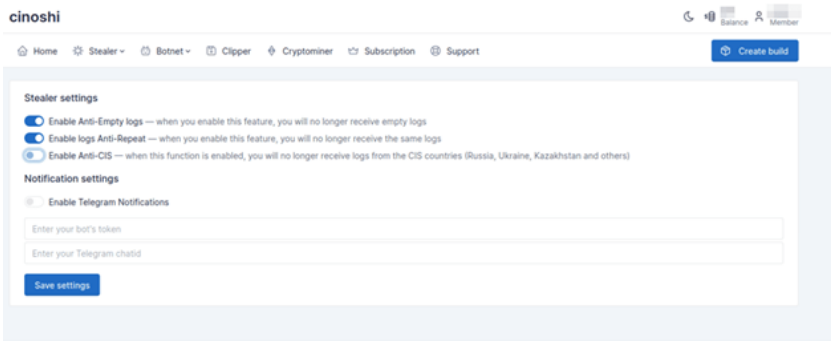


Figure 4 – Cinoshi Web Panel

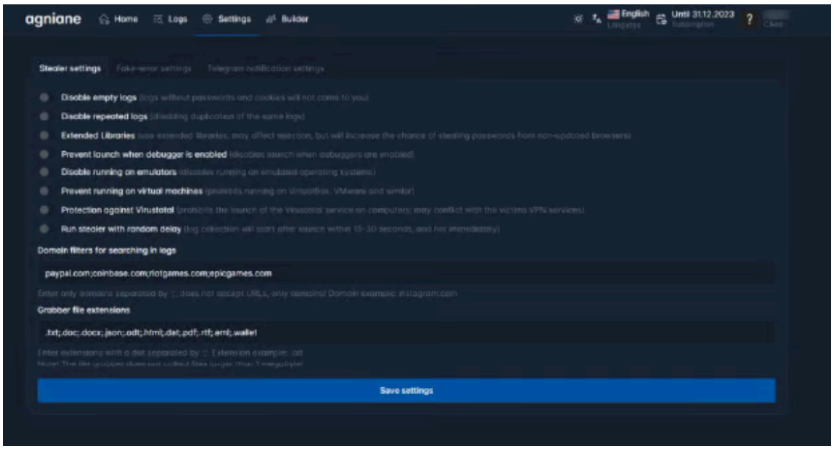


Figure 5 – Agniane Web Panel (Source: Zscaler)

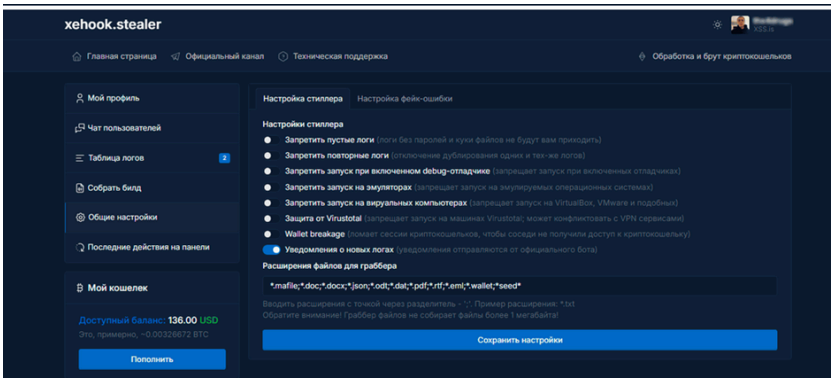


Figure 6 – Xehook Stealer Web Panel

Overlaps with Agniane Stealer

During our analysis, we discovered that the Xehook stealer appears to be an upgraded iteration of the Agniane stealer, sharing many similarities in their code base and functionalities. Upon further investigation, we came across a Cisco report detailing the Agniane stealer; we observed that the configuration data mentioned therein closely resembled that of the Xehook stealer. However, we identified three additional fields in the Xehook stealer's configuration, namely "selfmelf," "domaindetect," and "filext," indicating that the Xehook stealer boasts enhanced capabilities compared to its predecessor. Furthermore, both stealer binaries were found to be communicating with the same command-and-control (C&C) server (*hxxps://trecube[.]com/*), following a similar sequence of requests, suggesting a strong connection between the two variants.

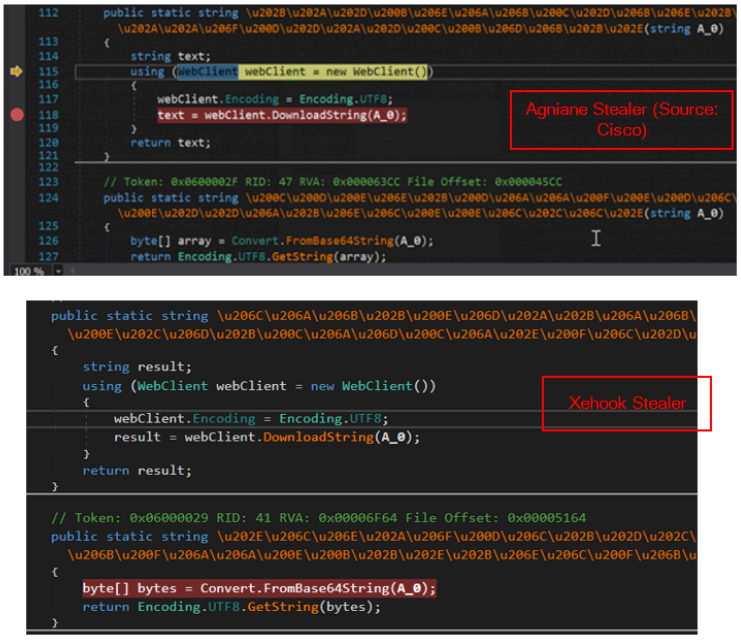


Figure 7 – Code Overlaps

Initial Infection

During our investigation, we came across a SmokeLoader binary (*Sha256*: *fa7f5300459c71d70f1f7b0d0c96aa245fad2a98d55d39a53455d2a7191d8cc9*) that was responsible for downloading the loader for the Xehook stealer from below URL

- *hxxps://45.15.156[.]174/index[.]php/s/24Sr2FjZQm8gXFA/download/ketamine[.]exe*.

Additionally, Spamhaus has reported an instance of a SmokeLoader binary distributing the Xehook stealer, indicating the active utilization of SmokeLoader in propagating the Xehook stealer payload.

Technical Analysis

The initial file is a 32bit .NET loader which is obfuscated using a crypter. This loader consists of a time-based restriction or expiration check, where after a certain date (*in this case, 14 days after February 24, 2024*), the method will throw an

exception and terminate the execution of the malware.

This time-based restriction serves as a control mechanism for the malware author. It allows them to limit the lifespan of the malware, potentially evading detection, or analysis after a certain period.

The figure below shows the time-based check.

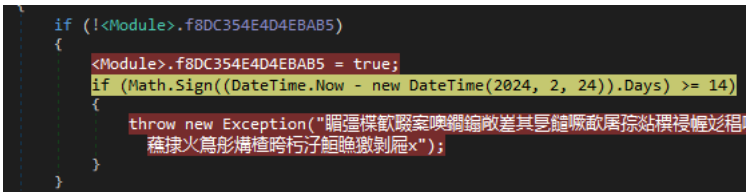


Figure 8 – Time-Based Check

After that, the Loader binary decodes the kernel32.dll name, which is stored in reverse order. It utilizes various functions of kernel32.dll, such as:

- *FreeConsole()*
- *GetProcAddress()*
- *LoadLibraryA()*
- *CreateThread()*
- *WaitForSingleObject()*
- *VirtualProtect()*

The loader will later leverage a few of these functions to inject the Stealer payload.

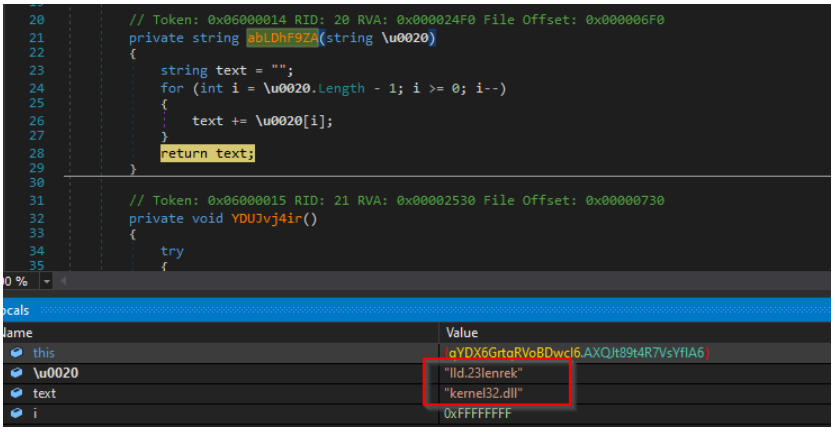


Figure 9 – Reversing DLL Name

The loader proceeds to decrypt the encrypted stealer payload contained within a byte array. This decryption occurs in two stages: the data undergoes mathematical operations and XORing.

The figure below shows the decryption process.

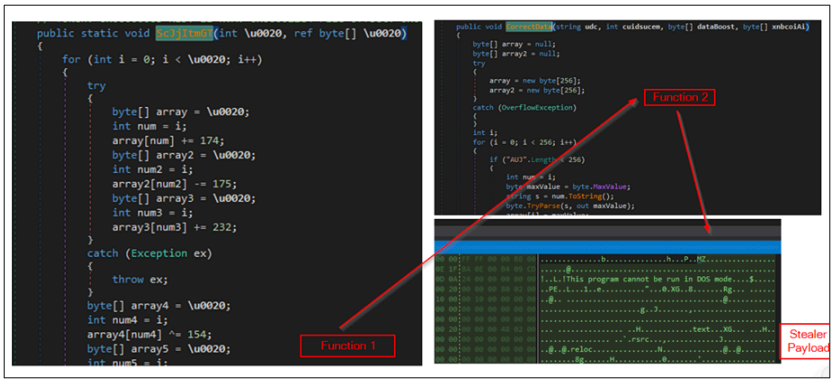


Figure 10 – Decrypts Stealer Payload

The loader initiates the execution of a legitimate Windows binary named *RegAsm.exe* located at *"C:\\Windows\\Microsoft.NET\\Framework\\Version_Number\\RegAsm.exe"*. It is an assembly registration tool primarily

used to register .NET assemblies with COM (Component object model).

The figure below shows the process tree.

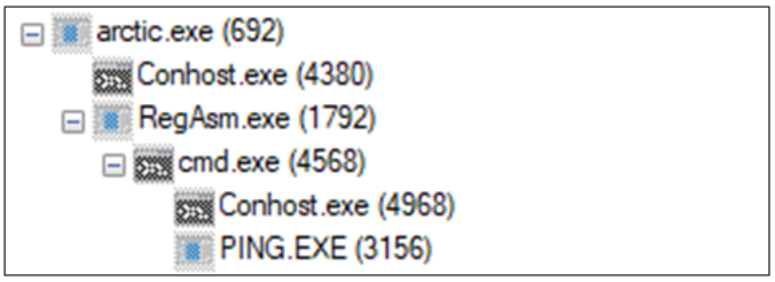


Figure 11 – Process Tree

Subsequently, it injects the stealer payload into the *RegAsm.exe* process. It utilizes functions such as *VirtualAlloc*, *VirtualProtect*, and WriteProcessMemory. This technique, known as Process Injection, is commonly utilized by malware to evade detection and defense mechanisms.

The diagram below illustrates the Process Injection method.

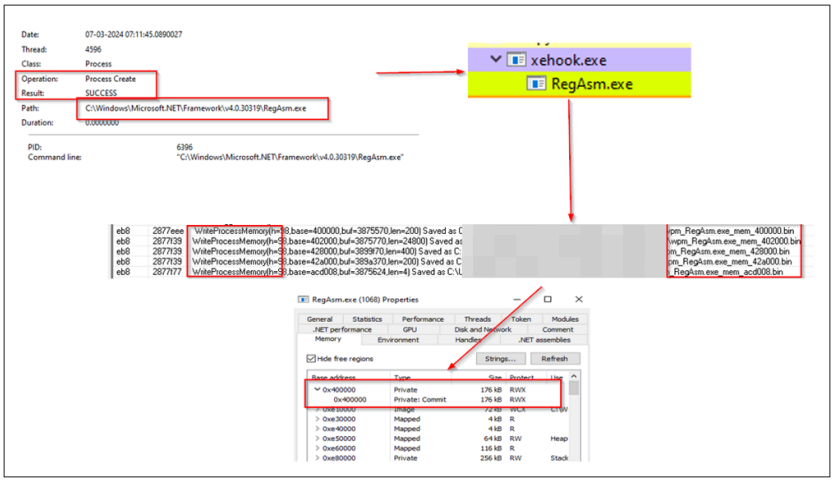


Figure 12 – Process Injection

The stealer payload is a 64-bit .Net binary. It is highly obfuscated and stores the encrypted strings in a byte array. It uses a single decryption function that applies some XOR and SHIFT operations to all the strings passed to it as a parameter. Initially, the stealer payload decrypts the C&C URL, as shown in the figure below.

- *hxxps://trecube[.]com/*
- *hxxps://nc1337[.]online/*

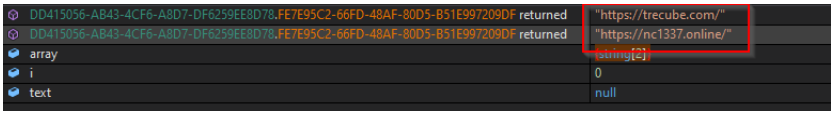


Figure 13 – Decrypts C&C URL

Then, the stealer proceeds to confirm the availability of the C&C servers by employing the *DownloadString()* method of the *WebClient* instance. This method retrieves the web content of the designated C&C URL as a string. Subsequently, it inspects the returned value for the existence of *"index.html."* If this string is found in the response, the stealer proceeds with the designated URL for C&C communications. The figure below shows the check for selecting the C&C URL.

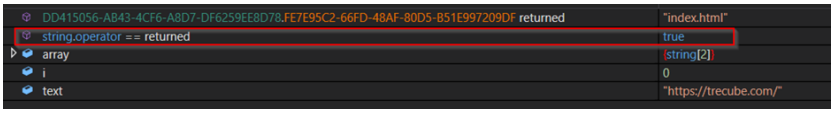


Figure 14 – Check of Selecting C&C

Now, the malware initiates a GET request to the below C&C URL:

- *hxxps://tricube[.]com/getjson[.]php?id=40*

In return, the C&C server sends configuration information for the stealer payload in JSON format. This configuration data likely contains instructions and settings for the malware to follow, specifying its behavior, targets, and other operational parameters.

The figure below shows the Configuration data sent by the C&C server.

Figure 15 – Configuration Data

The stealer payload then parses the configuration data by splitting it into an array and then initializes a dictionary to store key-value pairs for subsequent utilization.

The Xehook Stealer contains a code snippet that appears to be checking for the presence of specific system languages. It initializes an array of *CultureInfo* objects representing different languages. Then, it iterates through the installed system language, comparing each language with the ones specified in the array. If any of the installed languages match the ones specified in the array, the stealer payload terminates itself. This mechanism is used for language-based checks or configurations within a software application.

The stealer prevents its execution in the following countries.

System Language Code	Country
{ru-RU}	Russia
{kk-KZ}	Kazakhstan
{ro-MD}	Moldova
{uz-UZ}	Uzbekistan
{be-BY}	Belarus
{az-Latn-AZ}	Azerbaijan
{hy-AM}	Armenia
{ky-KG}	Kyrgyzstan
{tg-Cyrl-TJ}	Tajikistan

The figure below shows the decoded language codes.

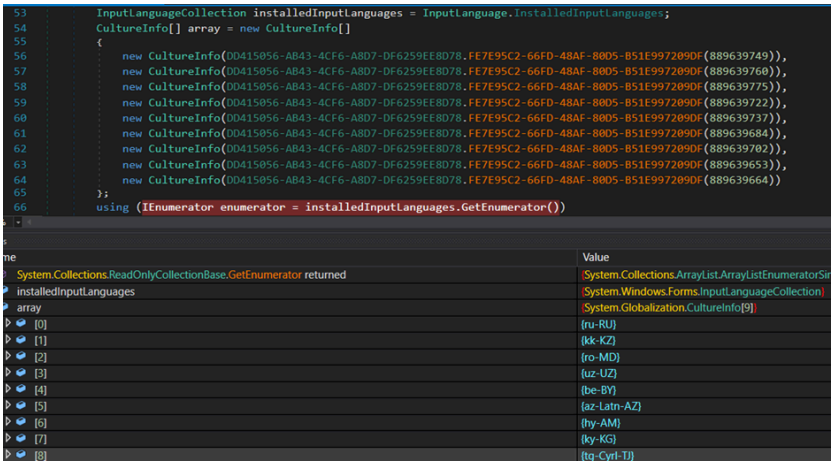


Figure 16 – Decoded Language Codes

After that, the stealer payload decrypts the names of processes associated with the malware analysis tools. Then, it employs the *GetProcesses()* method to retrieve the current list of running processes. It then compares these process names with the decrypted ones to determine if any match exists and terminates itself. This process allows the payload to identify potential instances of analysis or detection environments and avoid its execution.

The following are the process names for which the stealer does an Anti-Analysis check.

- *processhacker*
- *netstat*
- *netmon*
- *tcpview*
- *wireshark*
- *filemon*
- *regmon*
- *cain*

The Figure below shows the decryption process.

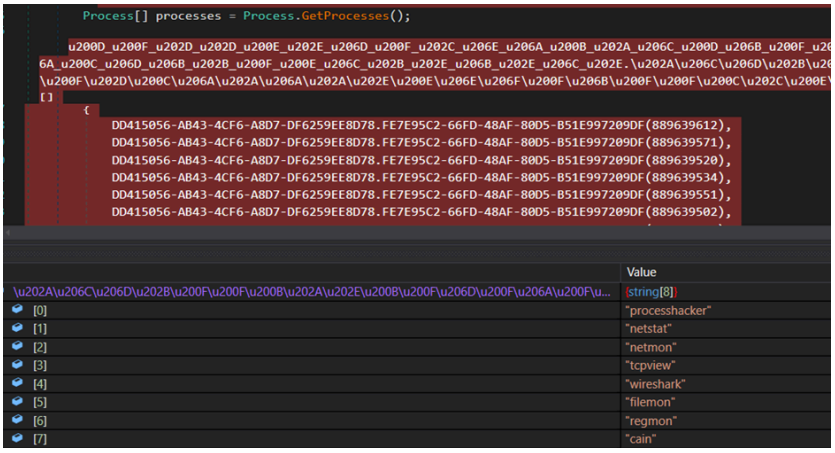


Figure 17 – Decrypting Process Names

Next, the malware utilizes *DateTime.Now.Ticks* method to perform a Tick count. It is a known Anti-Analysis technique utilized by the environment to detect the sandbox environment, as virtual machines often exhibit different timing behaviors compared to physical machines due to the underlying virtualization layer.

The figure below shows the tick count check.

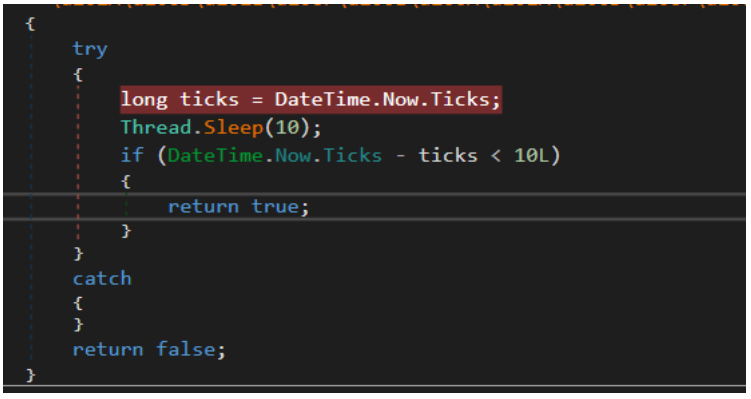


Figure 18 – Checking Tick Count

Now, the stealer binary uses the Windows Management Instrumentation (WMI) query "*Select * from Win32_ComputerSystem*" to gather information about the computer system. This query retrieves various system properties, including details about the hardware, operating system, and potentially installed software.

The stealer examines the data from the WMI query to determine if the computer is running in a virtual environment. It terminates itself if it finds strings like "VMware" or "VirtualBox," often used with virtual machines.

The figure below shows the WMI query.

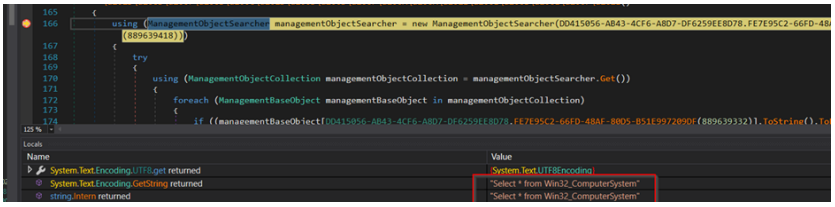


Figure 19 – Using WMI Query to Detect Virtualized Environment

Afterward, the stealer makes a GET request to the URL *hxxp://ip-api[.]com/json/?fields=11827*, which returns a JSON response containing information about the IP address. This response consists of the following fields:

- *country:* Indicates the country where the IP address originates.
- *countryCode:* Provides the country code corresponding to the country.
- *city:* Specifies the city associated with the IP address (in this case, it's empty).
- *zip:* Indicates the ZIP code of the location (empty in this response).
- *isp:* Represents the Internet Service Provider (empty in this response).
- *org:* Specifies the organization or company associated with the IP address (empty in this response).
- *as:* Provides information about the Autonomous System (AS) number or name (empty in this response).
- *query:* Provides the IP address from which the request originated.

Subsequently, the Xehook stealer employs a *MemoryStream* object to temporarily store sensitive data collected from the victim's system, which will be later converted to a stealer log.

The Xehook stealer verifies the configuration data to identify files to extract from the victim's system. The TA can define any file extension within the configuration, prompting the stealer to capture and transmit the specified files. In the stealer log data, these files will be stored under a folder named *"Files"*. The generated search event logs from the stealer payload are illustrated in the figure below.

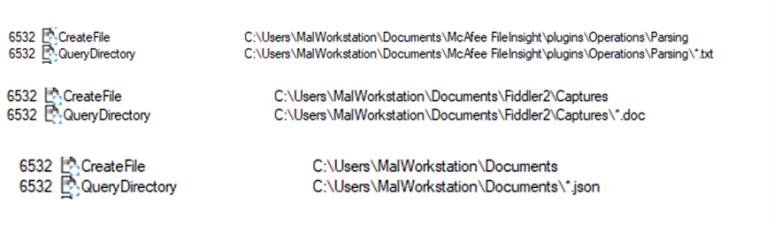


Figure 20 – File Grabber

The TA asserted in the cybercrime forum post that this stealer can effectively target all Chromium and Gecko based browsers. We examined a technique employed by the TA to accomplish this. During its directory traversal process, the stealer appends *"User Data\Local State"* to the directories it traverses. The presence of this path indicates the installation of a Chromium-based browser on the victim's system, allowing the stealer to proceed with the theft of browser-related data, including cookies, autofill, and login credentials. We did not observe this stealer binary targeting the Gecko browser. One possible reason could be that the stealer binaries are customizable through the web panel.

In contrast to other stealers, this particular one dynamically stores stolen data and generates logs directly for data exfiltration. As a result, folders such as "Cookies" and "Autofill" are created within the log file structure to store specific types of data such as cookies and autofill information.

This stealer configuration has a field named "domain detect". The TAs utilize this field to steal login credentials for only domains they mention in the config data.

The figure below shows the directory enumeration performed by the stealer to locate Chromium-based browsers.

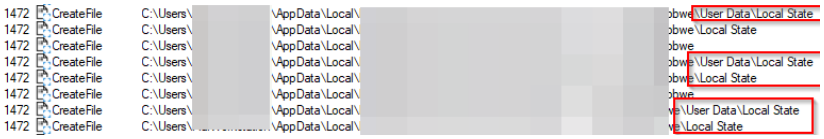


Figure 21 – Searching for Chromium-Based Browsers

We have observed over 110 chromium browser extensions, which this stealer targets. Each browser extension has a unique extension ID, so the stealer utilizes these IDs to search for extensions.

The Xehook stealer targets the following extensions.

Name	Extension ID	Name	Extension ID
Splikity	Jhfjfclepacoldmjmkmdlmganfaalklb	YubiKey	mammpjaaoinfelloncbbpomjcihbkmmc
Avira Password Manager	Caljgklbbfbcjjanaijlacgncafpegll	Google Authenticator	khcodhlfkpmhibicdjjblnkgimdepgnd
iWallet	Knccchdigobghenbbaddojjnnaogfppfj	Microsoft Authenticator	bfbdnbpibgndpjfhonkflpkijfapmomn
Wombat	Amkmjjmmflddogmhpjloimipbofnfjih	Authy	gjffdbjndmcafeoehgdldobgjmlepcal
MEW CX	Nlbmnnijcnlegkjjpcfjclmcfggfefdm	Duo Mobile	eidlicjlkaiefdbgmdepmmicpbggmhoj
NeoLine	Cphhlgmgameodnhkjdmkpanlelnlohao	OTP Auth	bobfejfdlhnabgglompioclndjejolch
Terra Station	Aiifbnbfobpmeekipheeijimdpnlpgpp	FreeOTP	elokfmmmjbadpgdjmgglocapdckdcpkn
Keplr	Dmkamcknogkgcdfhhbddcghachkejeap	Aegis Authenticator	ppdjlkfkedmidmclhakfncpfdmdgmjpm
Sollet	Fhmfendgdocmcbmfikdcogofphimnkno	LastPass Authenticator	cfoajccjibkjhbdjnpkbananbejpkkjb
ICONex	Flpiciilemghbmfalicajoolhkkenfel	Dashlane	flikjlpgnpcjdienoojmgliechmmheek
KHC	Hcflpincpppdclinealmandijcmnkbgn	Keeper	gofhklgdnbnpcdigdgkgfobhhghjmmkj
TezBox	Mnfifefkajgofkcjkemidiaecocnkjeh	RoboForm	hppmchachflomkejbhofobganapojjol
Byone	Nlgbhdfgdhgbiamfdfmbikcdghidoadd	KeePass	lbfeahdfdkibininjgejjgpdafeopflb
OneKey	Ilbbpajmiplgpehdikmejfemfklpkmke	KeePassXC	kgeohlebpjgcfiidfhhdlnnkhefajmca
Trust Wallet	Pknlccmneadmjbkollckpblgaaabameg	Bitwarden	inljaljiffkdgmlndjkdiepghpolcpki
MetaWallet	Pfknkoocfefiocadajpngdknmkjgakdg	NordPass	njgnlkhcjgmjfnfahdmfkalpjcneebpl
Guarda Wallet	Fcglfhcjfpkgdppjbglknafgfffkelnm	LastPass	gabedfkgnbglfbnplfpjddgfnbibkmbb
Exodus	Idkppnahnmmggbmfkjhiakkbkdpnmnon	Authenticator	bhghoamapcdpbohphigoooaddinpkbai
Jaxx Liberty	Mhonjhhcgphdphdjcdoeodfdliikapmj	EOS Authenticator	oeljdldpnmdbchonielidgobddfffflal
Atomic Wallet	Bhmlbgebokamljgnceonbncdofmmkedg	BrowserPass	naepdomgkenhinolocfifgehidddafch
Electrum	Hieplnfojfccegoloniefimmbfjdgcgp	MYKI	bmikpgodpkclnkgmnpphehdgcimmided
Mycelium	Pidhddgciaponoajdngciiemcflpnnbg	Bread	jifanbgejlbcmhbbdbnfbfnlmbomjedj
Coinomi	Blbpgcogcoohhngdjafgpoagcilicpjh	Airbitz	ieedgmmkpkbiblijbbldefkomatsuahh
GreenAddress	Gflpckpfdgcagnbdfafmibcmkadnlhpj	KeepKey	dojmlmceifkfgkgeejemfciibjehhdcl
Edge	Doljkehcfhidippihgakcihcmnknlphh	CommonKey	chgfefjpcobfbnpmiokfjjaglahmnded
BRD	Nbokbjkelpmlgflobbohapifnnenbjlh	Zoho Vault	igkpcodhieompeloncfnbekccinhapdb
Samourai Wallet	Apjdnokplgcjkejimjdfjnhmjlbpgkdi	Norton Password Manager	admmjipmmciaobhojoghlmleefbicajg
Copay	ieedgmmkpkbiblijbbldefkomatsuahh	Trezor Password Manager	imloifkgjagghnncjkhggdhalmcnfklk
Trezor	jpxupxjxheguvfyhfhahqvxvyqthiryh	MetaMask	nkbihfbeogaeaoehlefnkodbefgpgknn
Ledger Live	pfkcfdjnlfjcmkjnhcbfhfkkoflnhjln	TronLink	ibnejdfjmmkpcnlpebklmnkoeoihofec
Ledger Wallet	hbpfjlflhnmkddbjdchbbifhllgmmhnm	BinanceChain	fhbohimaelbohpjbbldcngcnapndodjp

Bitbox	ocmfilhakdbncmojmlbagpkjfbmeinbd	Coin98	aeachknmefphepccionboohckonoeemg
Digital Bitbox	dbhklojmlkgmpihhdooibnmidfpeaing		

Other browser extension IDs include.

Extension IDs	
lbfeahdfdkibininjgejjgpdafeopflb	fijngjgcjhjmmpcmkeiomlglpeiijkld
jbdaocneiiinmjbjlgalhcelgbejmnid	pdadjkfkgcafgbceimcpbkalnfnepbnk
afbcbjpbpfadlkmhmclhkeeodmamcflc	bfnaelmomeimhIpmgjnjophhpkkoljpa
hnfanknocfeofbddgcijnmhnfnkdnaad	fhilaheimglignddjgofkcbgekhenbh
blnieiiffboillknjnepogjhkgnoac	mgfffbidihjpoaomajlbgchddlicgpn
cgeeodpfagjceefieflmdfphplkenlfk	aodkkagnadcbobfpggnjeongemjbjca
ocefimbphcgjaahbclemolcmkeanoagc	kpopkelmapcoipemfendmdghnegimn
fihkakfobkmkjojpchpfgcmhfjnmnfpi	hmeobnffcmdkdcmlb1gagmfpfboieaf
nfinomegcaccbhchhgflladpfbajihdf	Ipfcbjknijpeeillifnkikgncikgfhdo
nanjmdkhkinifnkgdeggcnhdaammmj	dngmlblcodfobpdpecaadgfbeggfjfnm
nkddgncdjgifcddamgcmfnlhccnimig	ejbalbakoplchlghecdalmeeeajnimhm
fnnegphlobjdpkhecapkijjdkgcjhkib	mlbafbjadjidk1bhgopoamemfibcpdfi
nphplpgoakhhjchkkhmiggakijnkhfnd	jnlgamecbpmbajjfhmmmlhejkemejdma
penjlddjkjgpnkllboccdgccekpkcbin	ppbibelpcjmhbdihakflkdcoccbgbkpo
fldfpgipfncgndfolcbkdeeknbbbnhcc	mcohilncbfahbmgdjkbpemcciiolgcge
pnccjgokhbnggghddhahcnaopgeipafg	enabgbdfcbaehmbigakijjabdpdnimlg
egjidjbpglichdcondbcbdnbeeppgdph	fopmedgnkfpebgllppeddmmochcookhc
imlcamfeniaidioeflifonfjeeppbIda	khpkpbbcccdmmclmpigdgddabeilkdpd
ajkifnllfhikkjbjopkhmjoieikeihjb	lnnnmfcpbkafcpgdilckhmhbkkbpkmid
kkpllkodjeloidieedojogacfhpaihoh	aholpfdialjgjfhomihkjbmgjidlcdno
kgdijkcfiglijhaglibaidbipiejjfdp	kilnpioakcdndlodeeceffgjdpojajlo
efbglgofoippbgcjepnhiblaibcnclgk	ebfidpplhabeedpnhjnobghokpiioolj
onhogfjeacnfoofkfgppdlbmlmnplgbn	mdjmfdffdcmnoblignmgpommbefadffd
phkbamefinggmakgklpkljjmgibohnba	aijcbedoijmgnlmjeegjaglmepbmpkpi

This stealer also targets applications such as Steam, Telegram, Discord, and FileZilla. Additionally, the stealer captures a screenshot of the victim's system, which will be saved as *"Screenshot.jpg"* in the log file.

Once all the stolen data is gathered in the memory stream, it is converted into a byte array and then written to a log file utilizing the *File.WriteAllBytes()* method.

The resulting log file is stored within a folder created under the *AppData\Local* directory, with a name generated randomly using alphanumeric characters (A-Z, 0-9) and having a length of 32 characters. The figure below shows the method for storing the stolen data.

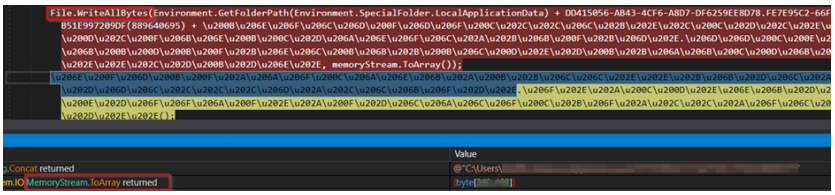


Figure 22 – Writing Stolen Data to Disk

The following figure illustrates the malware directory with log files generated by the stealer.

Figure 23 – Log File

The figure below shows the “About PC.txt” file containing the infected machine’s system details.

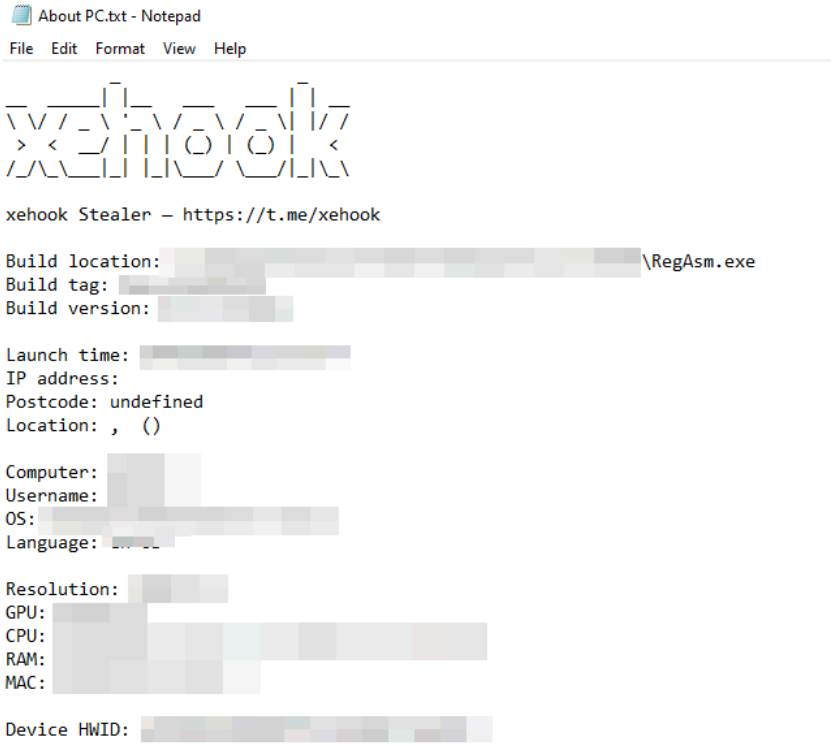


Figure 24 – About PC.txt

Then, the malware starts exfiltrating the data using a POST request. It exfiltrates the data to the following URL:

- *hxxps[:]//trecube[.]com/gate.php?id=40&build=new_cloudnever&passwords=0&cookies=0&username=&country=&ip=&BSSID=&wallets=0&token=xehook&ext=&filters=&p*

The figure below shows the network activity.

Figure 25 – Exfiltrating Data

After the upload operation, the code deletes the uploaded file from the victim’s system, potentially erasing any theft traces.

The figure below shows the code for uploading and deleting log files.

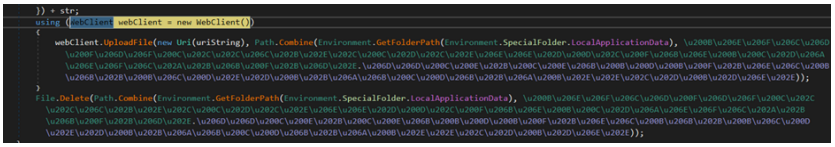


Figure 26 – Deleting the Log FIle

Finally, the stealer is designed to throw a fake error message, providing a layer of deception to its operation. This fake error message is configurable by the threat actor (TA), who can choose whether the error message should be displayed and its content through the configuration settings.

The figure below shows the fake error message box.

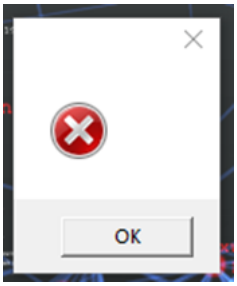


Figure 27 – Fake Error Message Box

Conclusion

Xehook Stealer is one of the few stealers with dynamic data collection capabilities and can target many browser extensions. The connection between Xehook Stealer, Agniane, and the Cinoshi project reveals a complex ecosystem of malware development and propagation. This linkage suggests a potential strategy of rebranding and iterative enhancement to evade detection and prolong malicious operations.

The codebase, communication infrastructure, and distribution vectors overlap among entities like Xehook Stealer, Agniane, and the Cinoshi project, underscoring the interconnected nature of cyber threats. This overlap indicates that cybercriminals often reuse or repurpose code, infrastructure, and tactics across different malware variants and campaigns. As a result, proactive detection and robust defense mechanisms become essential to combat such threats effectively.

Our Recommendations

- The initial entry point may originate via spam emails. Therefore, it's advisable to deploy strong email filtering systems to identify and prevent the dissemination of harmful attachments.
- Deploy strong antivirus and anti-malware solutions to detect and remove malicious executable files.
- Enhance the system security by creating strong, distinct passwords for each of the accounts and, whenever feasible, activate two-factor authentication.
- Set up network-level monitoring to detect unusual activities or data exfiltration by malware. Block suspicious activities to prevent potential breaches.
- Enable two-factor authentication whenever possible for an additional layer of security.
- Periodically change your passwords, especially for sensitive accounts like email, banking, and social media.
- Regularly back up data to guarantee the ability to recover it in case of an infection and keep users informed about the most current phishing and social engineering methods employed by cybercriminals.

MITRE ATT&CK® Techniques

Tactic	Technique	Procedure
Execution (TA0002)	User Execution (T1204)	The user needs to manually execute the file.
Defense Evasion (TA0005)	Obfuscated Files or Information (T1027)	Binary may include packed or crypted data.
Defense Evasion (TA0005)	Software Packing (T027.002)	Binary may include packed or crypted data.
Defense Evasion (TA0005)	Deobfuscate/Decode Files or Information (T1140)	Decode data using Base64 in .NET
Defense Evasion (TA0005)	Process Injection (T1055)	Loader injects stealer payload into RegAsm.exe.
Defense Evasion (TA0005)	Indicator Removal (T1070)	Delete the stealer logs.
Credential Access (TA0006)	OS Credential Dumping (T1003)	Tries to harvest and steal browser information (cookies, passwords, etc)

Discovery (TA0007)	System Information Discovery (T1082)	Queries the system information (host name, IP address, etc).
Discovery (TA0007)	File and Directory Discovery (T1083)	Stealer enumerate files for grabbing.
Collection (TA0009)	Data from Local System (T1005)	Tries to harvest and steal browser information (cookies, passwords, etc)
Collection (TA0009)	Archive Collected Data (T1560)	Stealer compress the stolen data with ZIP extension.
C&C (TA0011)	Application Layer Protocol (T1071)	Malware exe communicate to C&C server.

Indicators of Compromise (IOCs)

Indicators	**Indicator Type**	**Description**
a3882ac90190c7ccbea744dde58f0a107b67e3eea0024b12d18e72faf9a55b1c	SHA256	Loader Xehook stealer
daea71a3094e0c90554a77e95b0b354d1515f99e70fa5013f09302a5bb04dde0	SHA256	Xehook Stealer Binary
hxxps://trecube[.]com/ hxxps://nc1337[.]online/	URL	C&C
fa7f5300459c71d70f1f7b0d0c96aa245fad2a98d55d39a53455d2a7191d8cc9	SHA256	SmokeLoader
hxxps://45.15.156.174/index[.]php/s/24Sr2FjZQm8gXFA/download/ketamine[.]exe	URL	Malicious URL

Source: https://cyble.com/blog/xehook-stealer-evolution-of-cinoshis-project-targeting-over-100-cryptocurrencies-and-2fa-extensions/