

School of Design and Informatics

BSc Ethical Hacking, 2022/23

**An Evaluation of Modular Incident Response Plans for Efficient Cyber Incident Mitigation in Businesses**

**SnakeKeylogger Analysis**

**Author:** Martin Georgiev **Supervisor**: Natalie Coull

Head of Division of Cybersecurity Abertay

Contents

[1 Introduction 1](#_Toc134539262)

[**1.1** **Background** 1](#_Toc134539263)

[**1.2** **Aims** 1](#_Toc134539264)

[2 Procedure 2](#_Toc134539265)

[**2.1** **Overview** 2](#_Toc134539266)

[**2.2** **Static Analysis** 2](#_Toc134539267)

[**2.2.1** **File Hashes Discovery** 2](#_Toc134539268)

[**2.2.2** **String Extraction** 3](#_Toc134539269)

[**2.2.3** **In-depth Inspection** 4](#_Toc134539270)

[**2.2.4** **Dropped Files** 6](#_Toc134539271)

[**2.3** **Dynamic Analysis** 10](#_Toc134539272)

[**2.3.1** **Detonation Symptoms** 10](#_Toc134539273)

[**2.3.2** **Network Symptoms** 13](#_Toc134539274)

[**2.3.3** **Memory Analysis** 13](#_Toc134539275)

[**2.4** **Yara Rule** 15](#_Toc134539276)

[3 Results 16](#_Toc134539277)

[**3.1** **Analysis Results** 16](#_Toc134539278)

[4 Discussion 17](#_Toc134539279)

[**4.1** **General Discussions** 17](#_Toc134539280)

[**4.2** **Countermeasures** 17](#_Toc134539281)

[**4.2.1** **Pre-infection Countermeasures** 17](#_Toc134539282)

[**4.2.1.1** **Frequent Security Updates** 17](#_Toc134539283)

[**4.2.1.2** **Distinguishing Spam** 17](#_Toc134539284)

[**4.2.1.3** **Blacklisting Unknown Applications and Anti-Virus Software** 17](#_Toc134539285)

[**4.2.1.4** **File Scanners and IDS** 17](#_Toc134539286)

[**4.2.2** **Post-infection Countermeasures** 18](#_Toc134539287)

[**4.2.2.1** **Multi-Factor Authentication** 18](#_Toc134539288)

[**4.2.2.2** **Password Change** 18](#_Toc134539289)

[**4.2.2.3** **Eradicate the Malware** 18](#_Toc134539290)

[5 Discussion 19](#_Toc134539291)

[**5.1** **Conclusion** 19](#_Toc134539292)

[**5.2** **Future Work** 19](#_Toc134539293)

[6 References 20](#_Toc134539294)

[7 Appendices 21](#_Toc134539295)

[**Appendix A – Dropper Strings** 21](#_Toc134539296)

[**Appendix B – Second-Stage Payload Strings** 28](#_Toc134539297)

[**Appendix C – Memory Forensics** 38](#_Toc134539298)

[**Appendix C1 – Command Line** 38](#_Toc134539299)

[**Appendix C2 – Malfind** 39](#_Toc134539300)

[**Appendix C3 – Privileges** 39](#_Toc134539301)

[**Appendix C4 – Network Connections** 40](#_Toc134539302)

[**Appendix D – Yara Rule** 41](#_Toc134539303)

# **Introduction**

## **Background**

**SnakeKeylogger** is a credential stealer and keylogger first discovered in **November 2020**. The malware is written using .NET, using various modules. As mentioned earlier, its key capabilities include theft of saved credentials, taking screenshots, keylogging and collecting data from the user’s clipboard that are then sent to the attacker. They, however, could depend based on the sample.

The malware does not have any sophisticated propagation mechanisms like WannaCry. For this reason, it is mostly spread through phishing campaigns. In most identified cases, the phishing emails contained an attached PDF with macros. The document then enables macro execution and/or uses a vulnerable MS Office/PDF reader version to execute the malware upon opening the infected document. The embedded malicious code was usually a downloader that used PowerShell scripts to download the keylogger’s payload.

As the malware can steal various data types, it is considered a significant threat to corporate and private users. Based on the statistics provided by Checkpoint (Checkpoint, 2022) **SnakeKeylogger** was the second most prevalent malware infection after **AgentTesla** for 2022.

## **Aims**

The report aims to provide the reader with an analysis of **SnakeKeylogger** sample with the simplified Malware Analysis and Digital Forensics Framework and how it can show them information about its capabilities. The information can be used to take precautions until a professional team starts handling the case. It can also be given to the team handling the incident as this could greatly decrease the time to respond to the attack. The report will be split into three major sections to efficiently achieve the goal:

* Procedure – Analysis of the sample using the methodology and creation of a Yara rule to detect its signature
* Results – Overview and summary of the procedure and its subsections
* Discussion – General discussion and appropriate countermeasures

# **Procedure**

## **Overview**

As the developed methodology is aimed at small and medium-sized businesses that may not have a specialised response team due to budget constraints, the procedure will attempt to obtain as much intel as possible with simple techniques and tools. This will ensure that even users without significant technological knowledge can seamlessly follow it without significant difficulties. Considering the beforementioned requirements, an advanced static and dynamic analysis will not be performed on the sample as they require extensive knowledge of how computers operate, as well as high and low-level programming languages (JavaScript, C++, Assembly, etc.)

The hybrid analysis will be conducted on the sample in a safe testing environment. The static analysis will be achieved with a multitude of techniques and tools – obtaining file hashes, inspecting any human-readable strings within the binary (**Strings** (Russinovich, 2021) and **Floss** (Ballenthin, 2016)), analysing the executable’s library imports, functions and file entropy (**PEStudio** (Fox, 2021), and checking whether any known file packers have been used (**Exeinfo PE** (ALS, 2023)).

The Dynamic and Digital Forensic analysis will be combined as both can be done simultaneously. This section will cover the post-execution behaviour of the malicious sample – inspection of system modification (deleted/created/altered files, registries; PowerShell cmdlets execution, detection evasion and persistence mechanisms) with **Procmon**, possible network propagation or attempted communication with a **C&C** (**TCPView** (Russinovich, 2022), **Inetsim** (Hungenberg and Eckert, 2007)and **Wireshark** (Wireshark, 1997 – Present Day)). Additionally, analysing the system’s memory could display hidden processes, stolen data stored in the clipboard, possibly recover encryption keys and many more (**Volatility** (Volatility Foundation, 2020) and **WinPMem** (Cohen, et al., 2019)).

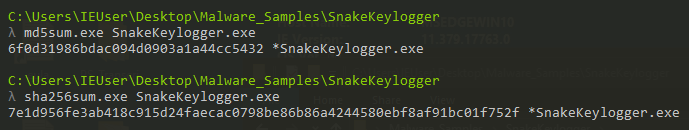
In the end, the obtained intel will be used to create a **Yara** (VirusTotal, 2013) rule for signature scanning that can then be incorporated with a scanner such as Strelka for passive file metadata scanning.

## **Static Analysis**

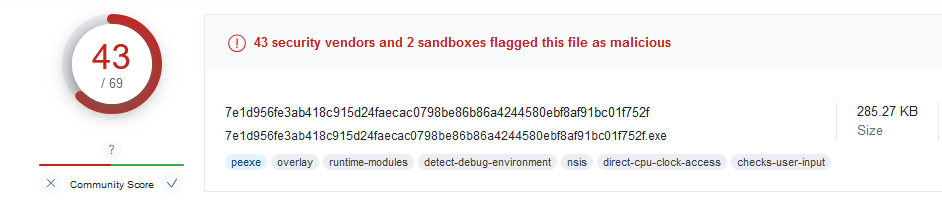
As mentioned in the previous section, the analysis will start with the static analysis of the sample. The basic static analysis provides limited information about the malware’s functionality. This, however, is often enough in introducing analysts with a base overview of the malicious software’s capabilities.

### **File Hashes Discovery**

Tools such as md5sum.exe and sha256sum.exe can be used from the command prompt to obtain the hashes of the malware. They will display the MD5 and SHA256 hashes of the sample respectively. The user can then copy them and check them in VirusTotal. This will check the databases of multiple antivirus vendors and display any matching results. (**Figure 2.2.1** and **Figure 2.2.2**)



***Figure 2.2.1*** *– Obtaining MD5 and SHA256 hashes.*

**

***Figure 2.2.2*** *– VirusTotal indicating the sample is malicious.*

### **String Extraction**

Human-readable strings can be extracted from files with a variety of tools. Some of the most common ones are **strings** and **floss**. In some cases, **floss** may be a better alternative to the former tool as it attempts to de-obfuscate and decode any strings which were intentionally made hard to read. The tool also has a multitude of flags for data filtering. In this case, **-n** was used as this puts a minimum character length of the extracted strings. The analyst set the length to **8** as they were interested in possible sentences, links and/or imported functions/libraries. The output was piped onto a text file (with **>**) to allow easier analysis and further filtering. (**Figure 2.2.3** and [**Appendix A**](#_Appendix_A_–))



***Figure 2.2.3*** *– Extracting strings with Floss.*

The analyst opened the file and searched for the “.exe” string to find executables embedded within the sample. They found a matching string in the manifest of the file – Nullsoft Install System(**Figure 2.2.4**) After some research, the investigator identified that this is a script-driven installer made for Microsoft Windows. As the tool is script-driven, malware developers can use it to create logic that can handle complex installation tasks. This also allows it to create web installers, communicate with the OS and software components, as well as install/update shared components for existing software.



***Figure 2.2.4*** *– Nullsoft Installl System name in the strings.*

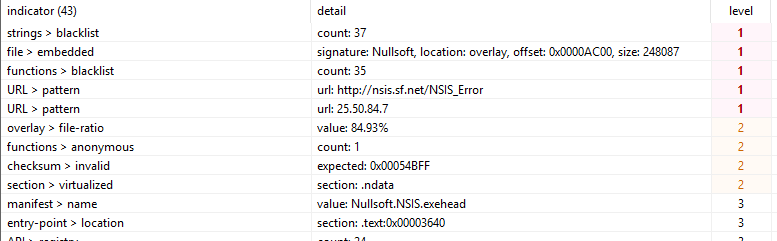
Further inspection showed a multitude of functions related to communication with application windows (SendMessageW, EnableWindow, SetWindowsTextW, etc.), clipboard manipulation (Close/OpenClipboard, SetClipBoardData, EmptyClipboard, etc.), file manipulation, environmental variable manipulation, disk space acquisition, etc. The sample could also alter the system’s registries and change shutdown privileges.

As the sample was under 300KB in size, the data that could be extracted from the strings was limited and the investigator had to proceed with the more in-depth inspection.

### **In-depth Inspection**

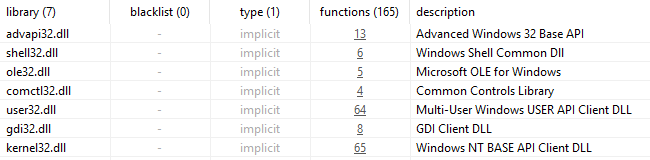
The analysis proceeded with **PEStudio**. **PEStudio** is a tool speeding up the initial malware analysis process and making it easier. The tool conducts a complete static analysis of the file and provides researchers with indicators, imports, libraries, and file entropy (randomness of data hinting at hidden/suspicious data).

Loading the sample into the tool revealed the entropy level (7.9 out of 8), indicating that the data within it is heavily obfuscated. Based on the compiler stamp, this specific binary was created on the 25th of September, 2021. The tool then showed five critical and four high indicators - blacklisted strings/functions, URL patterns, embedded files, virtualised sections and invalid checksums.(**Figure 2.2.5**)



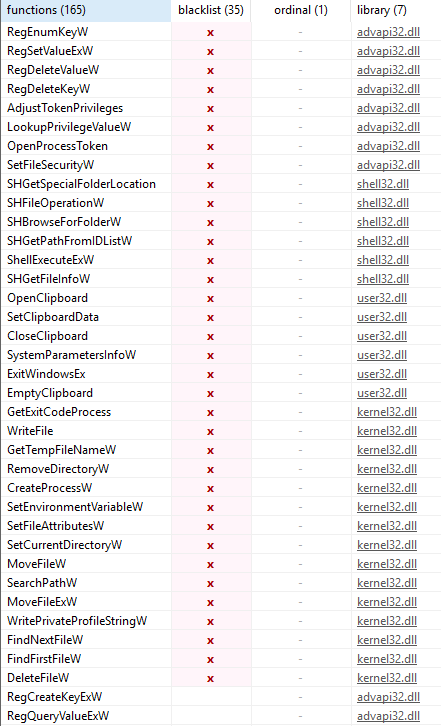
***Figure 2.2.5*** *– Malicious indicators.*

In terms of the libraries, the tool identified seven libraries, none of which were blacklisted. This, however, did not exclude that they could be used for malicious purposes (**Figure 2.2.6**)



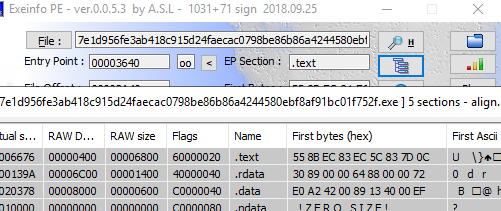
***Figure 2.2.6*** *– List of libraries.*

The tool discovered a total of thirty-five blacklisted functions. As mentioned in the previous section, they allowed the malware to alter the registries/privileges/clipboard, execute shell commands, and obtain information (file paths, names, etc.). There were also some non-blacklisted functions that were suspicious – custom function used to delete all pointers, object manipulation and image creation(**Figure 2.2.7**)



***Figure 2.2.7*** *– Partial list of functions utilised by the malware.*

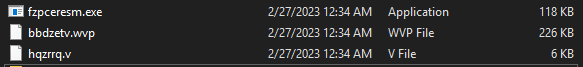
The analyst then checked whether the malware was packed using one of the commonly found packers such as UPX1 to evade detection. This was achieved with **ExeinfoPE**. Loading the file into the tool revealed that the EP section was in .text, indicating that no specific packer was used. “.Text” is the section of an executable that contains its code. (**Figure 2.2.8**)



***Figure 2.2.8*** *– SnakeKeylogger not using any specific code packer.*

### **Dropped Files**

Executing the file dropped a local second stage payload – an executable file (**fzpceresm.exe**), a WVP file (**bbdzetv.wvp**), and a V file (**hqzrrq.v**). The WVP file is usually associated as a project binary created by “Web Video Player”, while the V file extension usually contains components of a portable pixel map. The author could not identify their use as they were only written by the first executable and briefly accessed by the second stage executable. There were not associated with any other processes. (**Figure 2.2.9**)

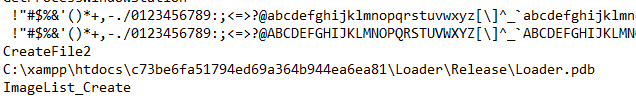


***Figure 2.2.9*** *– Dropped files.*

Based on the process tree, the initial binary was simply a dropper, while the dropped executable contained the malicious payloads. This was shown by the lifetime of the processes, the initial one stopping itself soon after execution, followed by the dropped file. This will be covered further in [**Section 2.3 Dynamic Analysis**](#_Dynamic_Analysis).

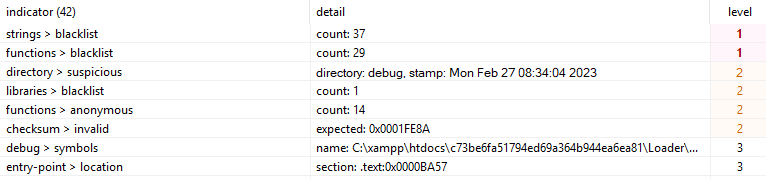
The analyst used **Floss** to extract the strings from the identified executable. As the other two files were not executables, **Floss** could not work for them. The researcher used **strings** but no human-readable strings were identified in the process.

Analysing the **Floss** output showed similar capabilities in terms of the displayed functions. ([**Appendix B**](#_Appendix_B_–Second-Stage)) It appeared that it could also create files, with a full Windows directory location being shown within the strings (**Figure 2.2.10**) XAMPP is an open-source web-server solution developed by Apache Friends. As this location did not exist on the system and there were no processes related, the author could not identify its exact functionality. It was possible that the adversary would use the server to connect to their C&C server and send the stolen data.



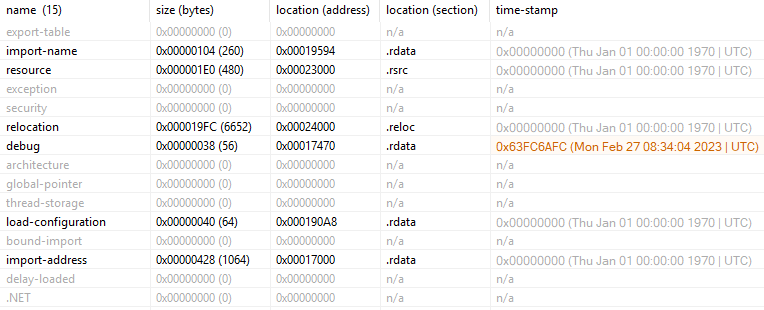
***Figure 2.2.10*** *– XAMPP related loader location found within the strings.*

Loading the file into PEStudio revealed two critical and four high level indicators – blacklisted functions/strings/libraries, suspicious debug directories (the 27th of February, 2023 – a day before the analysis of the sample), anonymous functions and invalid checksums. (**Figure 2.2.11**)



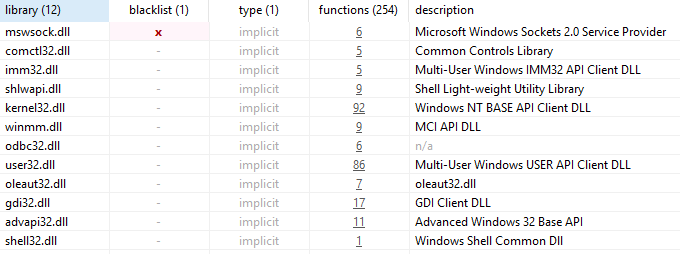
***Figure 2.2.11*** *– List of malicious indicators.*

Examining the sections proved that it really was last debugged on the 27th of February. (**Figure 2.2.12**)



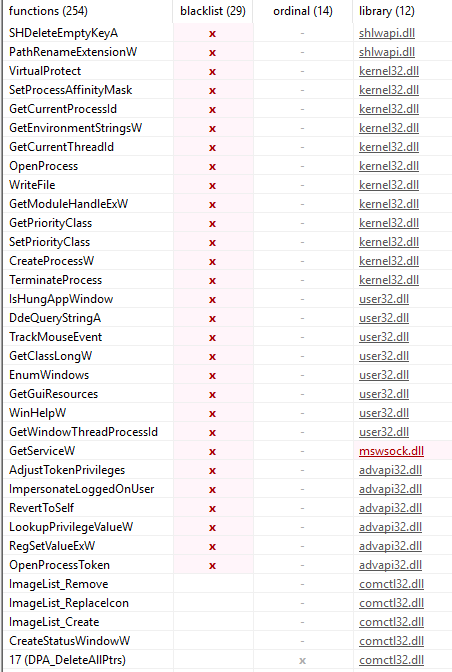
***Figure 2.2.12*** *– Sections within the executable.*

In terms of the libraries, there was only one blacklisted – **mswsock.dll** (Microsoft directory for web sockets, allowing communication between machines). The rest of the libraries provided it with shell capabilities, open database connectivity, multimedia creation and execution, user functions, etc. (**Figure 2.2.13**)



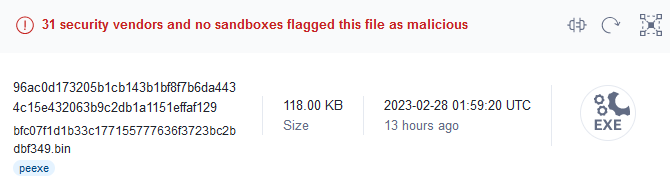
***Figure 2.2.13*** *– List of libraries.*

In terms of the functions, the binary contained a total of twenty-nine blacklisted functions. The allowed it to manage processes and memory, track mouse events, get graphical user interface resources, adjust privileges, get service names, set environmental values, etc. Based on this data and capability of other discovered samples, it was possible that the file contained the graphical libraries/functions to take screenshots of the user’s opened windows. (**Figure 2.2.14**)



***Figure 2.2.14*** *– List of functions.*

As the binary was recently debugged, the analyst was unsure if VirusTotal could identify it. Looking it up, however, showed that thirty-one out of sixty-nine vendors recognised it as a malicious program (Win32:Injector – an application which can open a second thread within its own code during its execution). (**Figure 2.2.15**)



***Figure 2.2.15*** *– Sha256 hash of the second stage payload recognised as malicious.*

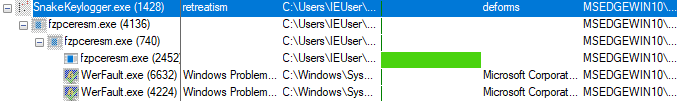
## **Dynamic Analysis**

Dynamic analysis of malware is achieved by detonating the sample in a safe environment (or surveying an already compromised environment) to see how it behaves on a local and network level. This may be dangerous if the safe environment is not properly set up as it may allow the malicious software to propagate to the physical machine, the user’s network, and possibly even other connected networks.

### **Detonation Symptoms**

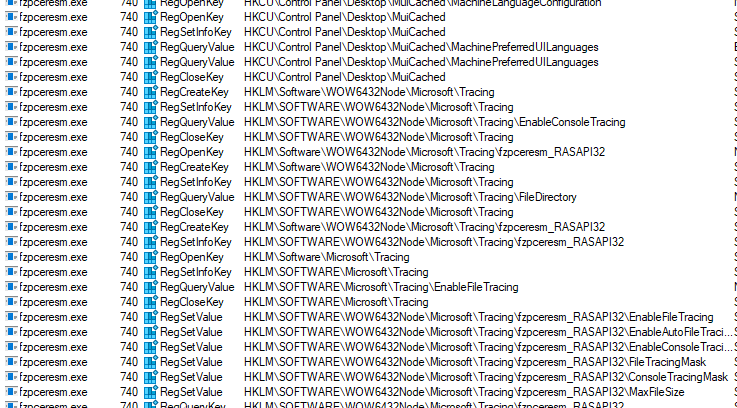
The dynamic analysis started with the initial detonation of the sample and inspection of the symptoms of the infected virtual machine. They attempted to detonate it both with and without administrator rights to see if there would be any differences. The sample executed itself in both cases. The virtual machine, however, showed no visible symptoms of infection.

Examining the behaviour in **Procmon** revealed that the initial sample was simply used as a detonator. The functionalities were then forwarded to **fzpceresm.exe** based on the lifetime of the processes. **SnakeKeylogger.exe** accessed some registries but there was no suspicious activity from it as no registries were altered – it was the generic process of installing its embedded libraries. All the malicious activity was committed by the dropped executable. (**Figure 2.3.1**)



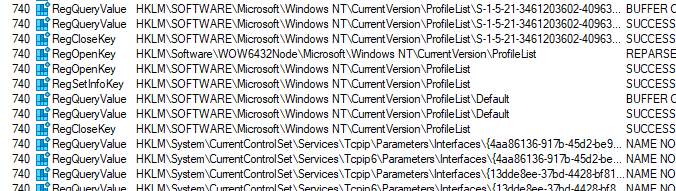
***Figure 2.3.1*** *– Lifetime of the malware’s processes.*

**WerFault.exe** was accompanied with the second-stage executable, possible to report any errors in its execution. They, however, appeared only in the start of the process’ lifetime. Compared to the detonator, this executable altered multiple registries connected to file tracing and the language lists. (**Figure 2.3.2**)



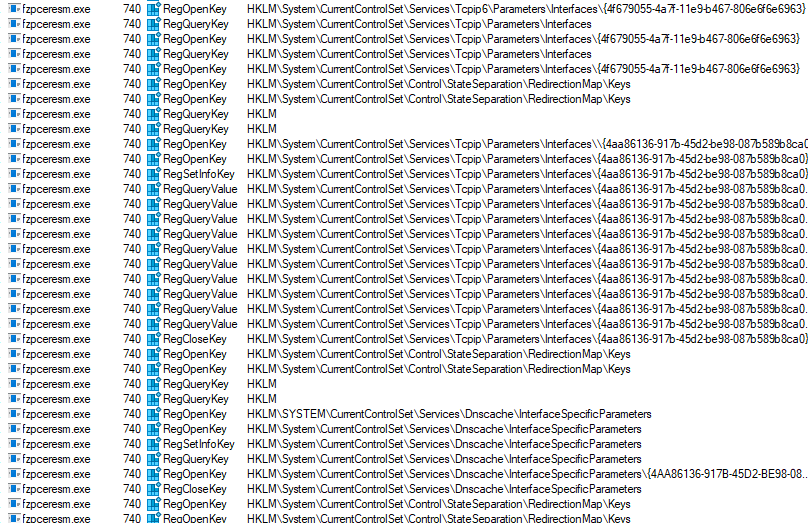
***Figure 2.3.2*** *– Writing to the file tracing and language registries.*

The malware then queried the profile list of the system, possible to identify the number of users and/or their names. (**Figure 2.3.3**)



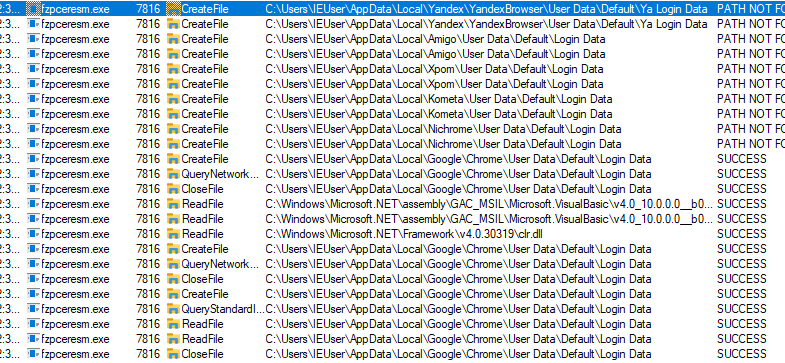
***Figure 2.3.3*** *– Profile list value query.*

It could also be seen that the malware accessed various registries connected to the system’s connectivity functions (DNSCache, TCP/IP, etc., possibly as a preparation to launch the XAMPP web server. (**Figure 2.3.4**)



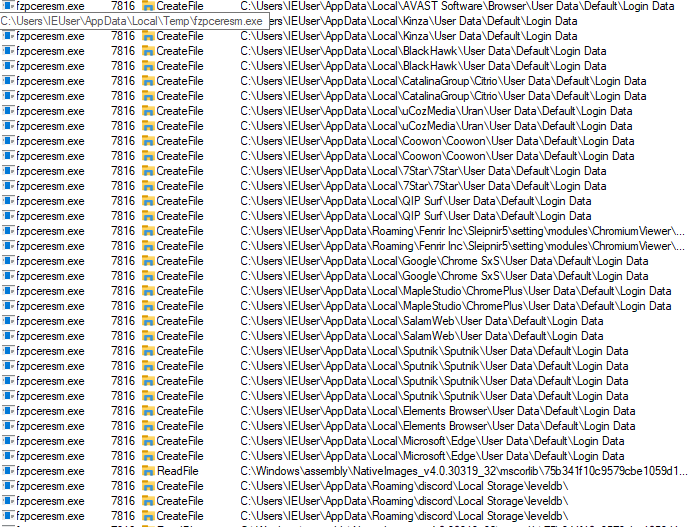
***Figure 2.3.4*** *– Accessing networking-related registries.*

The malware also stole data from various browsers including Yandex, Chrome, etc. (**Figure 2.3.5**)



***Figure 2.3.5*** *– Stealing saved browser credentials.*

Among other browser types, the malware also attempted to steal data related to the popular instant messaging social platform Discord (Joesandbox, 2023). (**Figure 2.3.6**)



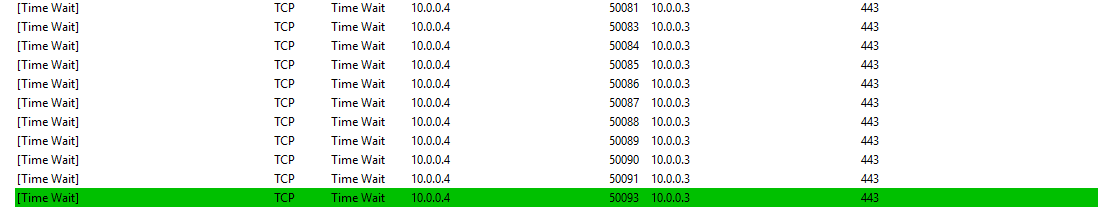
***Figure 2.3.6*** *– Attempting to access more browser and Discord data.*

The sample did not have any sort of persistence mechanism. It was possible that the connection it attempts ([**Section 2.3.2 Network Symptoms**](#_Network_Symptoms)) could be its C&C server. As it could not successfully connect, it may have not received appropriate instructions and potential third-stage payload files from the Telegram link.

### **Network Symptoms**

To examine the network symptoms, the analyst used tools such as **TCPView** and **Wireshark** while the machine was connected to a second one (**Remnux**) that had **Inetsim** running in the background. **Inetsim** is a tool which simulates an internet connection and a DHCP server, allowing it to capture all traffic in a host-only environment. **TCPView** allows the analyst to examine all connections going from and to the machine, while **Wireshark** is used to capture and examine the traffic towards **Inetsim**.

Opening **TCPView** revealed many attempts for connection from port **50010** on the local machine to port **443** (HTTPS)on the external address. The application iterated the local port many times until it closed all attempted connections. Some research revealed that it had to connect to a Telegram server with the following IP – 149.154.167.220:443. (**Figure 2.3.7**) The full data was not displayed due to the lack of a proper internet connection.

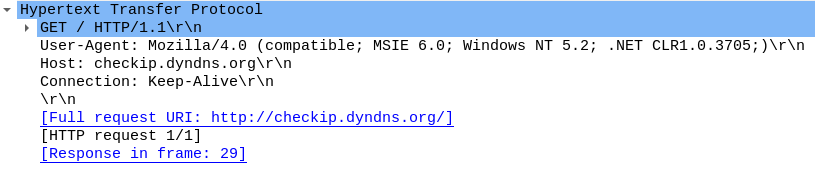


***Figure 2.3.7*** *– Attempting to connect to port 443 in an outgoing address.*

Afterwards, the analyst examined the network traffic in Wireshark, but the sample remained static. It only attempted to connect a specific domain that checks the IP of the infected machine. (**Figure 2.3.8** and **Figure 2.3.9**)



***Figure 2.3.8*** *– DNS traffic of the malware.*

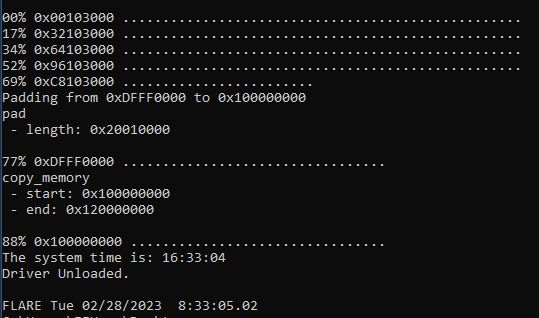


***Figure 2.3.9*** *– Http traffic of the sample.*

### **Memory Analysis**

Memory analysis can be a tedious task, but simple checks may reveal malicious behaviour in different processes. To analyse the memory of the infected virtual machine, the analyst used **Volatility 3.0** and **WinPMem** to dump the memory.

Dumping the memory could be easily achieved with the beforementioned tool. The analyst ran the following command in a **CMD** window to acquire the memory data in raw format: **winpmem\_mini\_x64\_rc2.exe memory.raw**. This command wrote the contents of the RAM in a file called **memory.raw**. (**Figure 2.3.10**)

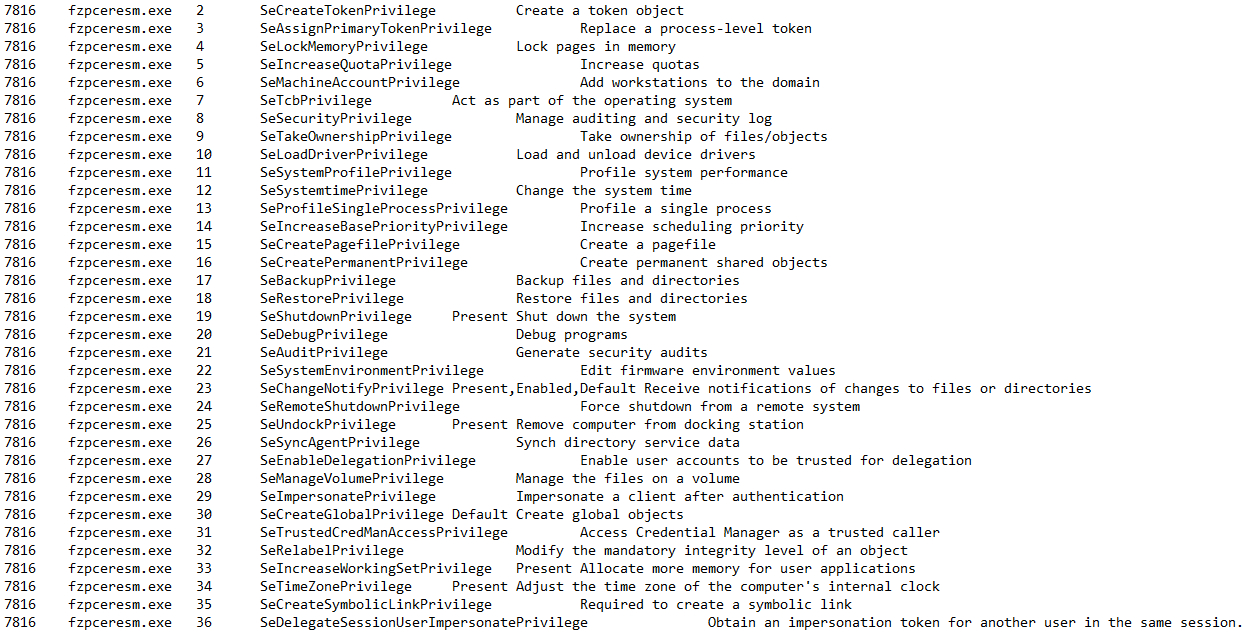


***Figure 2.3.10*** *– Dumping the memory using WinPMem.*

Afterwards, the researcher analysed different data from the raw memory dump. (**Figure 2.3.18**) This was achieved with various plugins, in this case specifically:

* windows.info – basic information about the windows machine
* windows.cmdline – executed cmd commands
* windows.malfind – looking for applications/services with injected malicious code
* windows.privileges – display the privileges for all dumped processes
* windows.pstree – lists the visible processes
* windows.netscan – lists the network connections.

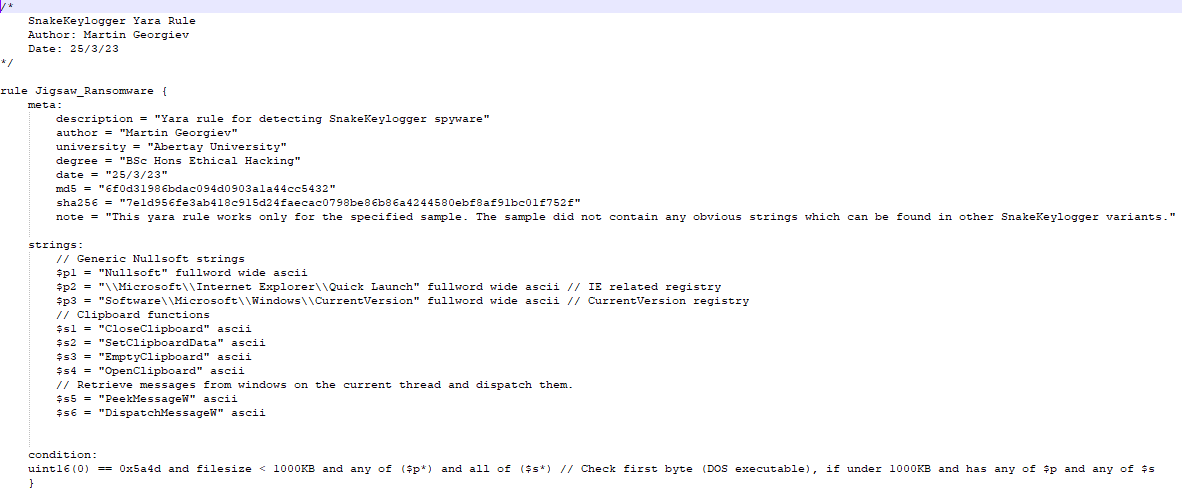
The analysis was achieved using the following command: **py vol.py -f C:\Users\IEUser\Desktop\memory.raw** **windows.***plugin\_name*. The results did not show any new details regarding the malware and its execution – no hidden connections, command line executions, and hidden processes. The privileges showed identical authority to the other analysed malware (Jigsaw and FileTour) (**Figure 2.3.11**). Malfind could not identify any injected code in legitimate processes. All extracted data can be found in [**Appendix C**](#_Appendix_C_–).



***Figure 2.3.11*** *– Privileges of the dropped executable.*

## **Yara Rule**

The Yara rule was created with notable data from the analysed strings. This is the easiest method to create simple Yara rules which can detect the malware – file names, specific commands and other strings related to this specific malware. (**Figure 2.4.1**) The rule can be found in [**Appendix D**](#_Appendix_D_–).



***Figure 2.4.1*** *– SnakeKeylogger Yara Rule.*

# **Results**

## **Analysis Results**

By conducting both static and dynamic analysis, the researcher identified a lot of valuable information regarding the behaviour of the keylogger and its capabilities. The static analysis revealed what functions have been used in the sample, the installer, and that the sample was created recently. It also showed that it may have the capability to take screenshots/media and send them to a foreign address through port 443.

The dynamic analysis showed how the logger works – does it contact any C&C servers, does it persist in the system, and does it steal any user information. The analysis proved the beforementioned points and revealed the detonation process of the sample, as well as what data it takes – browser and Discord credentials. The unsuccessful connection to the possible C&C server may have been the cause for the lack of persistence mechanisms.

# **Discussion**

## **General Discussions**

Analysing the sample revealed that it successfully checked the victim’s system for valuable data and attempted to steal it. The connection with the Telegram IP may have been to simply store the acquired data or to also obtain additional capabilities in a third-stage payload. The lack of a propagation mechanism was expected as it is known that SnakeKeylogger mainly propagates through phishing emails.

The sample did not appear to capture the user’s keystrokes, nor did it have any visible artefacts indicating at screenshot capturing, but this may not be the case with other samples. SnakeKeylogger is known to contain sophisticated methods of sensitive data acquisition and the capabilities may vary across different samples.

## **Countermeasures**

### **Pre-infection Countermeasures**

The most effective way to protect a system from infection would be before it becomes infected. As some of the modules contain cryptography functions, a fully working sample could encrypt the victim’s files. This, in most cases, would not allow them to retrieve their files.

#### **Frequent Security Updates**

One of the reasons why malware is successful is the lack of security patches or users refusing to apply the newest updates to their operating systems and/or anti-virus applications Keeping your system and anti-virus software up to date would ensure that publicly known vulnerabilities could not be exploited, and the AV may have updated signature databases to detect the sample.

#### **Distinguishing Spam**

As the malware is primarily distributed through social engineering, users must be able to distinguish spam emails from real ones. This also applies to legitimate and fake websites and/or files. Users should not open any links or execute files unless they know the sender and the nature of the link or file. Additionally, users should look for bad grammar, fearmongering, rushed actions or similar addresses to legitimate ones.

#### **Blacklisting Unknown Applications and Anti-Virus Software**

System administrators could put restrictions on users by blacklisting unknown software. This way they would not be able to execute suspicious applications and provide the system/network with damage control to prevent any harm. It could be achieved with Anti-Virus software and integrated browser protection.

An updated Anti-Virus software could be used to perform system scans or simply scan newly generated files. Some may even prevent the malicious software from executing itself if they recognise specific code patterns or behaviours.

#### **File Scanners and IDS**

Intrusion Detection Systems will alert security analysts if they detect any suspicious behaviour – phishing emails, specific signatures, etc. Some of them can also be combined with file scanners such as Strelka for greater detection accuracy. This way the internal SOC team could notice the threat before it causes any harm to the system.

### **Post-infection Countermeasures**

#### **Multi-Factor Authentication**

As the malware targets the user’s credentials, protecting the victim’s accounts is the only viable option for a post-infection countermeasure. With multi-factor authentication (MFA), the adversaries would not be able to log into the victim’s accounts unless the victim approves it through a third-party application. This will prevent them from accessing the accounts and any sensitive data they may hold.

#### **Password Change**

Combined with MFA, the infected users should identify their affected accounts (stored credentials in the accessed browser data) and should change the password of each affected account. This will ensure that the attackers would not have access to up-to-date credentials.

#### **Eradicate the Malware**

Despite not being able to identify any persistence mechanisms, the malware may contain hidden functionalities which can be discovered through advanced static analysis. It is vital to always remove all traces of infection to ensure that they will not execute themselves again and cause more damage.

# **Discussion**

## **Conclusion**

SnakeKeylogger is a powerful malware that can steal vital information about the infected users. Depending on the infected user and the data they have in their computers, damages could be catastrophic for a corporation. As the malware does not have any sophisticated methods of propagation, infections can be easily bypassed through basic knowledge about phishing emails and social engineering.

## **Future Work**

With more time, the analyst would attempt to reverse-engineer the sample to learn more data stealing functionalities of the malware. The data will show whether this specific sample has more hidden functions and if it is waiting for a third-stage payload to fully operate.

# **References**

ytsif (2014). TheZoo. [online] Available at: https://github.com/ytisf/theZoo [Accessed 15 Oct. 2022].

VirusTotal (2004 – Present Day). VirusTotal. [online] Available at: https://www.virustotal.com/gui/home/upload [Accessed 15 Nov. 2022].

Ballenthin, W. (2016). Floss. [online] Available at: https://github.com/mandiant/flare-floss. [Accessed 20 Nov. 2022].

Russinovich, M. (2021). Strings v2.54. [online] Available at: https://learn.microsoft.com/en-us/sysinternals/downloads/strings [Accessed 20 Nov. 2022].

ASL (2023). ExeInfoPE. [online] Available at: http://www.exeinfo.byethost18.com/?i=1 [Accessed 24 Nov. 2022].

Hungenberg, T and Eckert, M. (2007). INetSim. [online] Available at: https://www.inetsim.org/. [Accessed 27 Nov. 2022].

Russinovich, M. (2022). TCPView v4.17. [online] Available at: https://docs.microsoft.com/en-us/sysinternals/downloads/tcpview. [Accessed 27 Nov. 2022].

Wireshark. (1997 - Present Day). Wireshark Documentation. [online] Available at: https://www.wireshark.org/docs/. [Accessed 27 Nov. 2022].

Russinovich, M. (2022). Process Monitor v3.89. [online] Available at: https://docs.microsoft.com/en-us/sysinternals/downloads/procmon. [Accessed 28 Nov. 2022].

Fox, N. (2021). PeStudio Overview: Setup, Tutorial and Tips. [online] Available at: https://www.varonis.com/blog/pestudio. [Accessed 28 Nov. 2022].

Volatility Foundation (2020). Volatility 3.0. [online] Available at: https://www.volatilityfoundation.org/3 [Accessed 15 Dec. 2022].

Cohen, M., et al. (2019). WinPmem. [online] Available at: https://github.com/Velocidex/WinPmem [Accessed 15 Dec. 2022].

JoeSandbox (2023). SnakeKeylogger Analysis. [online] Available at: https://www.joesandbox.com/analysis/816683/0/html [Accessed 2023 20 Mar. 2023].

AbuseCH (2020). MalwareBazaar. [online] Available at: https://bazaar.abuse.ch/ [Accessed 24 Mar. 2023].

VirusTotal (2013). Yara - The pattern matching Swiss knife for malware researchers (and everyone else). [online] Available at: https://virustotal.github.io/yara/ [Accessed 29 Mar. 2023].

# **Appendices**

## **Appendix A – Dropper Strings**

*Note: The output was truncated to remove a lot of illegible data.*

FLOSS static ASCII strings

!This program cannot be run in DOS mode.

NulluN E

D$,+D$$P

SETUPAPI

CRYPTBASE

RichEd32

RichEd20

RegEnumValueW

RegEnumKeyW

RegQueryValueExW

RegSetValueExW

RegCloseKey

RegDeleteValueW

RegDeleteKeyW

AdjustTokenPrivileges

LookupPrivilegeValueW

OpenProcessToken

SetFileSecurityW

RegOpenKeyExW

RegCreateKeyExW

ADVAPI32.dll

SHFileOperationW

SHGetFileInfoW

SHBrowseForFolderW

SHGetPathFromIDListW

ShellExecuteExW

SHGetSpecialFolderLocation

SHELL32.dll

CoTaskMemFree

IIDFromString

CoCreateInstance

OleUninitialize

OleInitialize

ole32.dll

ImageList\_Destroy

ImageList\_AddMasked

ImageList\_Create

COMCTL32.dll

EndPaint

DrawTextW

FillRect

GetClientRect

BeginPaint

DefWindowProcW

SendMessageW

InvalidateRect

EnableWindow

ReleaseDC

LoadImageW

SetWindowLongW

GetDlgItem

IsWindow

FindWindowExW

SendMessageTimeoutW

wsprintfW

ShowWindow

SetForegroundWindow

PostQuitMessage

SetWindowTextW

SetTimer

CreateDialogParamW

DestroyWindow

ExitWindowsEx

CharNextW

DialogBoxParamW

GetClassInfoW

CreateWindowExW

SystemParametersInfoW

RegisterClassW

EndDialog

ScreenToClient

GetWindowRect

EnableMenuItem

GetSystemMenu

SetClassLongW

IsWindowEnabled

GetWindowLongW

SetWindowPos

GetSysColor

SetCursor

LoadCursorW

CheckDlgButton

GetMessagePos

CallWindowProcW

IsWindowVisible

CloseClipboard

SetClipboardData

EmptyClipboard

OpenClipboard

TrackPopupMenu

AppendMenuW

CreatePopupMenu

GetSystemMetrics

SetDlgItemTextW

GetDlgItemTextW

MessageBoxIndirectW

CharPrevW

CharNextA

wsprintfA

DispatchMessageW

PeekMessageW

USER32.dll

SelectObject

SetTextColor

SetBkMode

CreateFontIndirectW

CreateBrushIndirect

DeleteObject

GetDeviceCaps

SetBkColor

GDI32.dll

DeleteFileW

FindFirstFileW

FindNextFileW

FindClose

SetFilePointer

ReadFile

MultiByteToWideChar

lstrlenA

GetPrivateProfileStringW

WritePrivateProfileStringW

FreeLibrary

LoadLibraryExW

GetModuleHandleW

GlobalAlloc

GlobalFree

ExpandEnvironmentStringsW

lstrcmpW

lstrcmpiW

CloseHandle

SetFileTime

CompareFileTime

SearchPathW

GetShortPathNameW

GetFullPathNameW

MoveFileW

SetCurrentDirectoryW

GetFileAttributesW

SetFileAttributesW

GetTickCount

CreateFileW

GetFileSize

GetModuleFileNameW

GetCurrentProcess

ExitProcess

CopyFileW

SetEnvironmentVariableW

GetWindowsDirectoryW

GetTempPathW

GetCommandLineW

GetVersionExW

SetErrorMode

lstrlenW

lstrcpynW

WideCharToMultiByte

GetDiskFreeSpaceW

GlobalUnlock

GlobalLock

CreateThread

GetLastError

CreateDirectoryW

CreateProcessW

RemoveDirectoryW

lstrcmpiA

GetTempFileNameW

WriteFile

lstrcpyA

MoveFileExW

lstrcatW

GetSystemDirectoryW

GetProcAddress

GetModuleHandleA

GetExitCodeProcess

WaitForSingleObject

KERNEL32.dll

VerQueryValueW

GetFileVersionInfoW

GetFileVersionInfoSizeW

SHGetFolderPathW

SHFOLDER

SHAutoComplete

SHGetKnownFolderPath

InitiateShutdownW

RegDeleteKeyExW

ADVAPI32

GetUserDefaultUILanguage

GetDiskFreeSpaceExW

SetDefaultDllDirectories

KERNEL32

[Rename]

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0"><assemblyIdentity version="1.0.0.0" processorArchitecture="\*" name="Nullsoft.NSIS.exehead" type="win32"/><description>Nullsoft Install System v3.08</description><trustInfo xmlns="urn:schemas-microsoft-com:asm.v3"><security><requestedPrivileges><requestedExecutionLevel level="asInvoker" uiAccess="false"/></requestedPrivileges></security></trustInfo><compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1"><application><supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}"/><supportedOS Id="{1f676c76-80e1-4239-95bb-83d0f6d0da78}"/><supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/><supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/></application></compatibility></assembly>

NullsoftInstB>

r#dFZ(`G

rsNRAT4{

qhk&dZjm

H 2U[0Hw

}Uu[;%g\

41g:;iF:

"Xq|(\*qmV

ZF2GzXiyd

)^1jA]m\

Vx.|<SMZ

1H@r/8,(

TD$6jc\/

\_/ q5'EZWw`

Mqq&n69E7

l7Lt{-a3

BPB!#KU5

|}K:Nqkv

&l0h3[6bvpG

[EJUWVt%

f<):sME~

P~:@-{QLjd

r8HrH2^#i

OB\H2?L`

\G{(hi?T

>'qy`CJ(

%YZu/f%4h

{}9v7ud\N

\_MBX\*VK(J

&{!n<N/

zaOsN.^

(TJ[@/T<n

FB442|~l

4rQX9a!+

H{s=wD#\_m

s;%F7bhS].

QMHMLzPX

+uSe}4Ke

u(^>;\_I)pQ

cF+Nd5HSH

Pd#Lmo c

Wi{eY5qk

gpk3\wCa`q

I0cpAbhn

t}WlaOud

JTZE&oVU

2}$W!`d7

p>`ykUxi

JhON)"|BpR

6[(O+9yG

=bW{?A[&

Y2o/a[&o&\*

V('FVqT,

W4OTQdFs4hU

gTZ{6}Ahw

X.D635`E0

uHg~wyk;Ty

w]j? ABii

vz;sEt5Q@p

W>\*4ySH4

!\_xf{;l3

FLOSS static Unicode strings

RichEdit

RichEdit20W

.DEFAULT\Control Panel\International

Control Panel\Desktop\ResourceLocale

Software\Microsoft\Windows\CurrentVersion

\Microsoft\Internet Explorer\Quick Launch

verifying installer: %d%%

unpacking data: %d%%

... %d%%

Installer integrity check has failed. Common causes include

incomplete download and damaged media. Contact the

installer's author to obtain a new copy.

More information at:

http://nsis.sf.net/NSIS\_Error

Error writing temporary file. Make sure your temp folder is valid.

Error launching installer

SeShutdownPrivilege

NSIS Error

%u.%u%s%s

\*?|<>/":

%s%S.dll

MS Shell Dlg

MS Shell Dlg

msctls\_progress32

SysListView32

MS Shell Dlg

VS\_VERSION\_INFO

StringFileInfo

040904b0

CompanyName

FileDescription

retreatism

FileVersion

25.50.84.7

LegalCopyright

Copyright peripteral

LegalTrademarks

Canterburian

ProductName

25.50.84.7

VarFileInfo

Translation

FLOSS decoded 0 strings

FLOSS extracted 0 stackstrings

Finished execution after 59.093000 seconds

## **Appendix B – Second-Stage Payload Strings**

*Note: The output was truncated to remove a lot of illegible data.*

FLOSS static ASCII strings

!This program cannot be run in DOS mode.

t6Ht$Hu5Ph

<at-<rt"<wt

< t8< t4

UQPXY]Y[

~';\_t|%3

tHHt\*Ht#

248058040134

UTF-16LE

CorExitProcess

Wednesday

Thursday

Saturday

February

September

November

December

MM/dd/yy

dddd, MMMM dd, yyyy

HH:mm:ss

!"#$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklmnopqrstuvwxyz{|}~

GetCurrentPackageId

MessageBoxW

GetActiveWindow

GetLastActivePopup

GetUserObjectInformationW

GetProcessWindowStation

!"#$%&'()\*+,-./0123456789:;<=>?@abcdefghijklmnopqrstuvwxyz[\]^\_`abcdefghijklmnopqrstuvwxyz{|}~

!"#$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`ABCDEFGHIJKLMNOPQRSTUVWXYZ{|}~

CreateFile2

C:\xampp\htdocs\c73be6fa51794ed69a364b944ea6ea81\Loader\Release\Loader.pdb

ImageList\_Create

ImageList\_ReplaceIcon

ImageList\_Remove

CreateStatusWindowW

COMCTL32.dll

ImmGetRegisterWordStyleA

ImmSetStatusWindowPos

ImmGetDescriptionW

ImmUnlockIMC

ImmCreateIMCC

IMM32.dll

PathRenameExtensionW

SHRegGetBoolUSValueW

SHEnumKeyExA

StrRStrIA

SHDeleteEmptyKeyA

StrStrIW

UrlApplySchemeW

StrRChrIW

StrChrIW

SHLWAPI.dll

CloseHandle

GetLastError

HeapAlloc

HeapFree

GetProcessHeap

InitializeCriticalSection

EnterCriticalSection

LeaveCriticalSection

SetEvent

ResetEvent

WaitForSingleObject

CreateEventW

GetCurrentProcess

GetCurrentProcessId

TerminateProcess

CreateThread

CreateProcessW

SetPriorityClass

GetPriorityClass

OpenProcess

GetSystemInfo

GetVersionExW

VirtualProtect

IsWow64Process

GetModuleHandleW

GetProcessAffinityMask

SetProcessAffinityMask

GetProcessIoCounters

lstrcmpW

lstrcpynW

lstrcpyW

lstrcatW

lstrlenW

KERNEL32.dll

mixerGetLineInfoW

joySetCapture

waveOutOpen

midiInGetErrorTextA

WOW32ResolveMultiMediaHandle

joyGetPosEx

timeGetTime

waveOutGetID

mmioStringToFOURCCW

WINMM.dll

ODBC32.dll

GetWindowWord

CharPrevW

SwapMouseButton

SetCursorPos

TrackMouseEvent

DdeQueryStringA

RemovePropW

LoadStringA

LoadStringW

wsprintfW

IsHungAppWindow

SendMessageW

SendMessageTimeoutW

PostMessageW

DefWindowProcW

CallWindowProcW

IsWindow

ShowWindow

OpenIcon

MoveWindow

SetWindowPos

GetWindowPlacement

IsWindowVisible

IsIconic

BringWindowToTop

IsZoomed

CreateDialogParamW

DialogBoxParamW

EndDialog

GetDlgItem

SetFocus

SetTimer

KillTimer

EnableWindow

GetSystemMetrics

LoadMenuW

GetMenuState

DrawMenuBar

CreatePopupMenu

DestroyMenu

CheckMenuItem

EnableMenuItem

GetSubMenu

GetMenuItemCount

InsertMenuW

AppendMenuW

RemoveMenu

DeleteMenu

TrackPopupMenu

TrackPopupMenuEx

SetMenuDefaultItem

DrawTextW

SetForegroundWindow

ReleaseDC

BeginPaint

EndPaint

InvalidateRect

SetWindowTextW

GetClientRect

GetWindowRect

MessageBoxW

GetCursorPos

MapWindowPoints

GetSysColor

FillRect

CopyRect

GetWindowLongW

SetWindowLongW

GetClassLongW

GetParent

EnumWindows

GetWindowThreadProcessId

GetWindow

CheckMenuRadioItem

LoadBitmapW

LoadIconW

DestroyIcon

LoadImageW

CreateIconIndirect

TileWindows

CascadeWindows

WinHelpW

GetGuiResources

USER32.dll

MigrateWinsockConfiguration

GetAddressByNameA

AcceptEx

WSARecvEx

s\_perror

GetServiceW

MSWSOCK.dll

OLEAUT32.dll

CreateCompatibleBitmap

CreateCompatibleDC

CreatePen

CreateSolidBrush

DeleteDC

DeleteObject

ExcludeClipRect

RestoreDC

SelectObject

SetBkColor

SetPixel

SetTextColor

MoveToEx

ExtTextOutW

GDI32.dll

OpenProcessToken

AdjustTokenPrivileges

ImpersonateLoggedOnUser

RevertToSelf

LookupPrivilegeValueW

GetUserNameW

RegCloseKey

RegCreateKeyExW

RegOpenKeyExW

RegQueryValueExW

RegSetValueExW

ADVAPI32.dll

Shell\_NotifyIconW

SHELL32.dll

ReadFile

GetCommandLineA

IsDebuggerPresent

EncodePointer

DecodePointer

IsProcessorFeaturePresent

InitializeCriticalSectionAndSpinCount

RtlUnwind

MultiByteToWideChar

GetConsoleMode

ReadConsoleW

SetFilePointer

SetFilePointerEx

GetStdHandle

GetFileType

DeleteCriticalSection

InitOnceExecuteOnce

GetStartupInfoW

InterlockedDecrement

ExitProcess

GetModuleHandleExW

GetProcAddress

AreFileApisANSI

WriteFile

GetModuleFileNameW

InterlockedIncrement

IsValidCodePage

GetOEMCP

GetCPInfo

WideCharToMultiByte

SetLastError

GetCurrentThreadId

GetModuleFileNameA

QueryPerformanceCounter

GetSystemTimeAsFileTime

GetTickCount64

GetEnvironmentStringsW

FreeEnvironmentStringsW

UnhandledExceptionFilter

SetUnhandledExceptionFilter

FlsAlloc

FlsGetValue

FlsSetValue

SetStdHandle

FlushFileBuffers

GetConsoleCP

LoadLibraryExW

OutputDebugStringW

LoadLibraryW

GetStringTypeW

LCMapStringEx

HeapReAlloc

CreateFileW

WriteConsoleW

HeapSize

SetEndOfFile

abcdefghijklmnopqrstuvwxyz

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

ABCDEFGHIJKLMNOPQRSTUVWXYZ

<?xml version='1.0' encoding='UTF-8' standalone='yes'?>

<assembly xmlns='urn:schemas-microsoft-com:asm.v1' manifestVersion='1.0'>

<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">

<security>

<requestedPrivileges>

<requestedExecutionLevel level='asInvoker' uiAccess='false' />

</requestedPrivileges>

</security>

</trustInfo>

</assembly>

0"0(0\_0q0

2/2;2K2V2]2

5-5:5@5l5r5

6%6<6T6e6

9\*909D9d9l9}9

;!;';0;?;I;C<L<W<^<j<p<

=>=D=N=T=^=m=

>B>W>b>h>n>

?8?J?T?[?y?

1!1'1B1H1Q1i1s1}1

2'2/252>2G2[2a2g2m2

3 474^4g4n4

6'6.6T6Z6

808I8b8{8

9\*9C9\9u9

;#;@;];z;

1(1=1W1`1~1

272B2U2l2

6%767K7d7y7

959H9]9v9

:C;J;\_;f;p;{;

<&<-<F<L<X<\<`<d<h<

= =E>Q>h>p>x>

6(6,60646

7 7\*747>7H7R7\7f7p7z7

8$8.888B8L8V8a8

9&909:9D9N9X9b9l9v9

: :0:6:<:J:R:n:t:

;&;2;7;@;L;Q;X;l;

<"<(<.<3<8<Y<\_<k<r<x<

=.=@=U=j=

>/>6>g>w>}>

?(?9?A?y?

0$0J0e0k0t0{0

1$1+1=1C1O1W1u1{1

252;2D2K2

3&3-343?3K3S3e3k3t3{3

5'5E5K5T5[5m5s5

7%7+747;7U7Z7e7q7y7

9"9,9:9H9T9

:1:7:D:J:O:r:

:%;+;4;=;C;V;p;u;

<-<4<B<H<M<T<e<o<

=.=4=9=@=W=d=x=

>0>:>K>U>i>p>

?"?)?4?G?M?Y?l?r?x?

0(0-030g0p0x0

2 2-2:2G2T2`2n2{2

3)353;3L3T3\_3e3k3w3}3

4\*4/454J4R4Z4b4k4u4

5 5%5-545<5A5H5P5W5\_5d5

6'62686C6Q6l6w6}6

8!8'858?8\8i8

9B9M9U9`9p9

:?:L:W:g:

;:;E;b;h;q;z;

<"<4<:<@<J<Z<`<f<p<

<4=?=K=U=u=

>:>V>e>n>x>

>(?4?D?V?\_?r?

30G0P0c0

2 2`2r2{2

5"5\*505>5D5O5}5

6%6+616S6\6e6|6

7 707:7~7

71898D8K8W8]8|8

9'9-979=9N9T9

:#:/:7:=:H:Q:W:^:

;&;,;D;Z;w;

=A=H=P=V=^=t=z=

?"?'?2?8?S?Y?d?j?{?

0 0\*040>0H0R0\0f0p0z0

0 1'141Y1g1t1

2 2\*242>2H2R2\2f2p2z2

3$3.383B3L3V3`3j3t3

4%4.434I4R4W4m4v4{4

5!5\*5/5E5N5S5i5r5w5

6&6+6A6J6O6e6n6s6

7"7'7=7F7K7a7j7y7

7"8Y8p8x8

969C9I9o9

:'<0<F<^<

<E=M=S=`=l=w=}=

? ?$?(?,?0?4?8?<?@?D?H?L?a?n?}?

0'0W0n0v0

2(23292B2R2d2j2~2

3$3\*303:3J3P3V3`3u3{3

4)444T4q4x4

5)5/5Q5W5]5j5r5|5

6'61696?6E6

7!7\*7B7L7T7Z7i7o7

8"8,848:8I8O8b8h8q8z8

;\*<<<I<d<j<s<|<

=/=4=H=Q=e=n=

>0>9>M>V>j>s>

?!?5?>?R?[?o?x?

1$1-1H1R1X1p1{1

2 2(2.262L2R2t2z2

3"3(3.343:3@3F3L3R3X3^3d3j3p3v3|3

5#5<5C5K5P5T5X5

52686<6@6D6

7/7a7h7l7p7t7x7|7

:!:%:\*:0:4:::>:D:H:N:R:d:

>4>e>r>{>

;1;P;];g;

565H5M5S5g5u5

979?9H9Q9q9

;";3;R;h;r;x;

=!=6=G=S=Z=c=|=

6%6.646=6B6Q6X6

<,=2=[=v=

> ?-?;?T?]?|?

1\*161E1N1[1

2;2O2Z2m2{2

3#3\*373@3L3Y3b3

585G5N5|5

6#616M6i6o6

; <Q<X<k<

=$=\*=4=>=N=^=n=w=

>7?<?A?X?

0!0\*060;0c0

5"5(5.565<5B5J5S5Z5b5k5}5

7"747F7e7w7

=}>#?K?Y?

1#1<1C1K1P1T1X1

12282<2@2D2

3/3a3h3l3p3t3x3|3

:#:':+:/:3:7:;:?:M:

=#=7===B=,?7?G?y?

4484<4@4D4P4T4X4`7

8$8,848<8D8L8T8\8d8

4(4D4H4h4t4

; ;$;(;,;0;4;8;<;@;D;H;T;X;\;`;d;h;l;p;t;x;|;

<,<<<L<\<l<

<0>4>8>@>D>H>L>P>T>X>\>`>d>p>t>x>|>

FLOSS static Unicode strings

Software\Wine\TaskManager

Preferences

Software\Wine\TaskManager

Preferences

SeDebugPrivilege

Software\Microsoft\Windows NT\CurrentVersion\AeDebug

Debugger

Software\Microsoft\Windows NT\CurrentVersion\AeDebug

Debugger

DRWTSN32

%d:%02d:%02d

Emscoree.dll

- floating point support not loaded

- not enough space for arguments

- not enough space for environment

- abort() has been called

- not enough space for thread data

- unexpected multithread lock error

- unexpected heap error

- unable to open console device

- not enough space for \_onexit/atexit table

- pure virtual function call

- not enough space for stdio initialization

- not enough space for lowio initialization

- unable to initialize heap

- CRT not initialized

- Attempt to initialize the CRT more than once.

This indicates a bug in your application.

- not enough space for locale information

- Attempt to use MSIL code from this assembly during native code initialization

This indicates a bug in your application. It is most likely the result of calling an MSIL-compiled (/clr) function from a native constructor or from DllMain.

- inconsistent onexit begin-end variables

DOMAIN error

SING error

TLOSS error

runtime error

Runtime Error!

Program:

<program name unknown>

Microsoft Visual C++ Runtime Library

Wednesday

Thursday

Saturday

February

September

November

December

MM/dd/yy

dddd, MMMM dd, yyyy

HH:mm:ss

dkernel32.dll

USER32.DLL

((((( H

h(((( H

H

FLOSS decoded 1 strings

1023442870282056

FLOSS extracted 0 stackstrings

Finished execution after 19.384000 seconds

## **Appendix C – Memory Forensics**

### **Appendix C1 – Command Line**

5680 svchost.exe C:\Windows\system32\svchost.exe -k LocalService -p -s BthAvctpSvc

3528 dllhost.exe C:\Windows\system32\DllHost.exe /Processid:{3AD05575-8857-4850-9277-11B85BDB8E09}

1548 Procmon.exe "C:\ProgramData\chocolatey\lib\sysinternals\tools\Procmon.exe"

6428 Procmon64.exe "C:\Users\IEUser\AppData\Local\Temp\Procmon64.exe" /originalpath "C:\ProgramData\chocolatey\lib\sysinternals\tools\Procmon.exe"

7372 tcpview.exe "C:\ProgramData\chocolatey\lib\sysinternals\tools\tcpview.exe"

7816 fzpceresm.exe "C:\Users\IEUser\AppData\Local\Temp\fzpceresm.exe"

4352 svchost.exe C:\Windows\System32\svchost.exe -k LocalService -p -s LicenseManager

1144 svchost.exe C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted -p -s DsSvc

7044 chrome.exe Required memory at 0xa74a5e2020 is not valid (process exited?)

5824 TrustedInstall C:\Windows\servicing\TrustedInstaller.exe

7188 TiWorker.exe C:\Windows\winsxs\amd64\_microsoft-windows-servicingstack\_31bf3856ad364e35\_10.0.17763.1934\_none\_56ae2b0b9948b43a\TiWorker.exe -Embedding

3840 svchost.exe C:\Windows\system32\svchost.exe -k netsvcs -p -s DsmSvc

3120 svchost.exe C:\Windows\System32\svchost.exe -k wsappx -p -s ClipSVC

6684 svchost.exe C:\Windows\system32\svchost.exe -k netsvcs -p -s wuauserv

### **Appendix C2 – Malfind**

Volatility 3 Framework 2.4.1

PID Process Start VPN End VPN Tag Protection CommitCharge PrivateMemory File output Hexdump Disasm

5148 SearchUI.exe 0x1a0eb850000 0x1a0eb86ffff VadS PAGE\_EXECUTE\_READWRITE 6 1 Disabled

48 89 54 24 10 48 89 4c H.T$.H.L

24 08 4c 89 44 24 18 4c $.L.D$.L

89 4c 24 20 48 8b 41 28 .L$.H.A(

48 8b 48 08 48 8b 51 50 H.H.H.QP

48 83 e2 f8 48 8b ca 48 H...H..H

b8 60 00 85 eb a0 01 00 .`......

00 48 2b c8 48 81 f9 70 .H+.H..p

0f 00 00 76 09 48 c7 c1 ...v.H.. 48 89 54 24 10 48 89 4c 24 08 4c 89 44 24 18 4c 89 4c 24 20 48 8b 41 28 48 8b 48 08 48 8b 51 50 48 83 e2 f8 48 8b ca 48 b8 60 00 85 eb a0 01 00 00 48 2b c8 48 81 f9 70 0f 00 00 76 09 48 c7 c1

5148 SearchUI.exe 0x1a0ec060000 0x1a0ec0c3fff VadS PAGE\_EXECUTE\_READWRITE 5 1 Disabled

e9 fb ff 08 00 00 00 00 ........

00 cc cc cc cc cc cc cc ........

e9 eb 0f 09 00 00 00 00 ........

00 cc cc cc cc cc cc cc ........

e9 db 7f 09 00 00 00 00 ........

00 cc cc cc cc cc cc cc ........

e9 cb 1f 09 00 00 00 00 ........

00 cc cc cc cc cc cc cc ........ e9 fb ff 08 00 00 00 00 00 cc cc cc cc cc cc cc e9 eb 0f 09 00 00 00 00 00 cc cc cc cc cc cc cc e9 db 7f 09 00 00 00 00 00 cc cc cc cc cc cc cc e9 cb 1f 09 00 00 00 00 00 cc cc cc cc cc cc cc

### **Appendix C3 – Privileges**

7816 fzpceresm.exe 2 SeCreateTokenPrivilege Create a token object

7816 fzpceresm.exe 3 SeAssignPrimaryTokenPrivilege Replace a process-level token

7816 fzpceresm.exe 4 SeLockMemoryPrivilege Lock pages in memory

7816 fzpceresm.exe 5 SeIncreaseQuotaPrivilege Increase quotas

7816 fzpceresm.exe 6 SeMachineAccountPrivilege Add workstations to the domain

7816 fzpceresm.exe 7 SeTcbPrivilege Act as part of the operating system

7816 fzpceresm.exe 8 SeSecurityPrivilege Manage auditing and security log

7816 fzpceresm.exe 9 SeTakeOwnershipPrivilege Take ownership of files/objects

7816 fzpceresm.exe 10 SeLoadDriverPrivilege Load and unload device drivers

7816 fzpceresm.exe 11 SeSystemProfilePrivilege Profile system performance

7816 fzpceresm.exe 12 SeSystemtimePrivilege Change the system time

7816 fzpceresm.exe 13 SeProfileSingleProcessPrivilege Profile a single process

7816 fzpceresm.exe 14 SeIncreaseBasePriorityPrivilege Increase scheduling priority

7816 fzpceresm.exe 15 SeCreatePagefilePrivilege Create a pagefile

7816 fzpceresm.exe 16 SeCreatePermanentPrivilege Create permanent shared objects

7816 fzpceresm.exe 17 SeBackupPrivilege Backup files and directories

7816 fzpceresm.exe 18 SeRestorePrivilege Restore files and directories

7816 fzpceresm.exe 19 SeShutdownPrivilege Present Shut down the system

7816 fzpceresm.exe 20 SeDebugPrivilege Debug programs

7816 fzpceresm.exe 21 SeAuditPrivilege Generate security audits

7816 fzpceresm.exe 22 SeSystemEnvironmentPrivilege Edit firmware environment values

7816 fzpceresm.exe 23 SeChangeNotifyPrivilege Present,Enabled,Default Receive notifications of changes to files or directories

7816 fzpceresm.exe 24 SeRemoteShutdownPrivilege Force shutdown from a remote system

7816 fzpceresm.exe 25 SeUndockPrivilege Present Remove computer from docking station

7816 fzpceresm.exe 26 SeSyncAgentPrivilege Synch directory service data

7816 fzpceresm.exe 27 SeEnableDelegationPrivilege Enable user accounts to be trusted for delegation

7816 fzpceresm.exe 28 SeManageVolumePrivilege Manage the files on a volume

7816 fzpceresm.exe 29 SeImpersonatePrivilege Impersonate a client after authentication

7816 fzpceresm.exe 30 SeCreateGlobalPrivilege Default Create global objects

7816 fzpceresm.exe 31 SeTrustedCredManAccessPrivilege Access Credential Manager as a trusted caller

7816 fzpceresm.exe 32 SeRelabelPrivilege Modify the mandatory integrity level of an object

7816 fzpceresm.exe 33 SeIncreaseWorkingSetPrivilege Present Allocate more memory for user applications

7816 fzpceresm.exe 34 SeTimeZonePrivilege Present Adjust the time zone of the computer's internal clock

7816 fzpceresm.exe 35 SeCreateSymbolicLinkPrivilege Required to create a symbolic link

7816 fzpceresm.exe 36 SeDelegateSessionUserImpersonatePrivilege Obtain an impersonation token for another user in the same session.

### **Appendix C4 – Network Connections**

0xba87f8c8bd70 UDPv4 0.0.0.0 0 \* 0 7816 fzpceresm.exe 2023-02-28 16:12:31.000000

0xba87f8c8bec0 UDPv4 0.0.0.0 0 \* 0 7816 fzpceresm.exe 2023-02-28 16:12:31.000000

0xba87f8c8bec0 UDPv6 :: 0 \* 0 7816 fzpceresm.exe 2023-02-28 16:12:31.000000

0xba87ff5fb5a0 UDPv4 0.0.0.0 3702 \* 0 2424 svchost.exe 2023-02-27 14:06:30.000000

0xba87ff5fb840 UDPv4 0.0.0.0 0 \* 0 7816 fzpceresm.exe 2023-02-28 16:12:31.000000

0xba87ff5fc560 UDPv6 ::1 1900 \* 0 916 svchost.exe 2023-02-27 14:06:30.000000

0xba87ff5fc6b0 UDPv4 0.0.0.0 0 \* 0 7816 fzpceresm.exe 2023-02-28 16:12:31.000000

0xba87ff5fc6b0 UDPv6 :: 0 \* 0 7816 fzpceresm.exe 2023-02-28 16:12:31.000000

## **Appendix D – Yara Rule**

