# Malware Analysis of File using various tools and techniques

Supreet Kaur
Volgenau School of Engineering
George Mason University
Fairfax, VA, USA
supreet514@gmail.com

***Abstract*— The term malware means malicious software, this piece of software enters a computer system without the knowledge of the user with malicious intents, that can be stealing credentials, taking control over the system, denial of service. Malwares have their own defense system that makes it possible for malwares to hide from antivirus. In malware analysis, one of the biggest challenges is to collect useful data without risking the user's machine or system. In this paper, a case study of an executable is discussed, that involves the dynamic, static analyis and techniques used in reverse engineering to understand how it can infect the system and the level of threat it poses. Static analysis of a malware executable is valuable in getting insights of malware mechanisms; however, it does not provide the dynamic behavior of the malware.**

**The malware analysis of the sample was done in a closed lab environment using existing reverse engineering tools and techniques.**

***Keywords—static analysis, dynamic analysis, patching, YARA rules***

## I. INTRODUCTION

With the increase in the number of internet users, there has also been a substantial growth in the number of malwares. Computers systems are being compromised at a higher rate. Hacking has been far more advanced, sophisticated, and discreet. With the advancement of malware, malware analysis plays a crucial role as it helps an analyst to understand what a malware binary can do and how to detect it on a system and network, assess the damage, identify the exfiltrated file. Malware reverse engineering is the process of disassembling a software, which involves the process of reviewing the disassembled piece of code using a disassembler. With this technique binary instructions are converted to higher level constructs so that analysts can understand what that binary does and its impact, so that analysts can create a solution that mitigate the malware's malicious effects. This can also be helpful in determining identifiable patterns which can be used in preventing and curing future infections. The analysis of the given sample was done in an isolated lab environment. This paper is organized as follows Section II presents the Malware Detection Tools followed by Section III Lab Environment. Section IV discusses techniques used to analyze the sample and collected results from the analysis.

## II. MALWARE DETECTION TOOLS

During the analysis the sample, following tools were used:

A. *VirusTotal:* VirusTotal is a web based malware scanning tool, which allows to upload a hash of a file or the file itself, which then shows the result of the all the Antivirus machines that have detceted the given file ass malicious.

B. *Peid*: Peid can be useful in detecting the packer or compiler used in buliding the program. It has ability to detect most common packers, crypters and compilers for PE files, as of now it can detect approximately 600 signatures in PE file.

C. *Process Monitor:* It is an advanced monitoring tool which shows the malware interaction of the processes run by the ptogram with filesystem, process activity, thread activity and resgistry.

D. *Process Explorer:* It is a task manager which is used in dynamic analysis that gives insight processes currently running on the system.

E. *OllyDBG:* OllyDbg is a debbuging tool that is used in analyzing binary code. It used in tracing registers, identifying API calls, switches, tablws, constants and strings and procedures.

F. *IDA Pro*: IDA Pro is a disassembler, a tool that can be used for translating machine code to assembly code, which can be useful in static analysis.

G. *Hex editor:* A tool that allows inspection od each byte of the file.

H. *FaKeDNS:* FakeDNS responds to DNS queries, logsthe details of received request and sent responses.

I. *TCPView:* It a tool for displaying a detailed list of all UDP an dTCP endpoints on a system. It can help track down a process name when the victim machine connects over a port and gives a better understading about which process is responsible for a particular connection.

## III. Lab Environment

Analysis of a suspicious program needs a safe and secure lab environment so that the analysts do not infect their systems. In order to create a safe lab environment, it is important to keep in mind that the malware does not escape the virtualized environment and infect the host system. This can be ensured by using *host-only* network configuration mode, or network traffic withing the lab environment should be restricted while using the simulated services. In this lab setup, a physical system *Windows* as base operating system and with virtualization software *VMware workstation* was installed. A Windows based virtual machine (victim machine) was installed within the *VMware* for the analysis of malware. The benefit of using a VM is that, once the analysis is done, one can revert to a clean state by taking a snapshot of the clean state of the machine. A snapshot is an image of the memory and disk that captures the state and data of the VM.

## IV. Static Analysis

Static analysis is the process of analyzing a suspicious file without the execution of the sample. This type of analysis is usually done in the initial steps of analysis to extract useful information from the suspect binary, which helps in making decision on how to classify the subsequent analysis efforts. In the remaining of this section various techniques that were used will be discussed.

### A. Determining the Type of File

Attackers use various tricks to hide the file by modifying the file extension, that changes the appearance of the file that tricks the user to execute it. Relying on the file extension is not sufficient, that's why it is important that we determine the file type by file *signature*. It is a unique sequence of bytes that are written to header of the file. The Windows executables have a file signature as MZ in the first two bytes. However, on opening the given sample *winhlp.exe* in the Hexplorer Hex editor, it was observed that the file signature is NZ instead of MZ. This PE signature needs to be modified to reflect a proper PE file, which is changing NZ to MZ. After making the modification, the file was saved as *winhlp_fixed.exe*, the sample was placed in PeiID, within Peid it was observed that the executable is compiled with LCC Win32 1.x -> Jacob Navia [Overlay]. Also,
Entrypoint: 00001219 EP Section: .text
FileO Offset: 00000619 First Bytes: 64, A1, 00, 00
Linker Infor: 2.55 Subsystem: Win32GUI
Looking at the time and date in hex edit, we see that it was created at 47552E52h which converts to Tuesday, December 4, 2007 10:39:14 AM.

### B. Fingetprinting the Executable

Fingerprinting is a technique of generating the cryptographic hash values of the suspect binary. The hashing algorithm such as SHA1, SHA256, MD5 are usually considered for generating the file hashes for malware samples. Having a hash value of a malware can help in identifying if the newly dropped sample is the same as the original or not. File hash is also frequently used as an indicator that is shared with other malware analysts to help them identify the malware. File hash can also be used in determining whether the malware has been detected as malicious file by Anti-virus engines, which can be done by using web services like *VirusTotal.* If a suspect file is not detected as malicious by the Anti-Virus engines, that does not necessarily mean that the suspect binary is not malicious. The malware authors modify the code and use obfuscation techniques to avoid the detection. Following. is the list of hash values of the winhlp.exe that were generated during the analysis:

1) *MD5:* 3b683acbc0dc96e3fbf8bca715333789
2) *SHA*1:3928ee05b2de3ba8dc0d1733b7575b0dcf90bbef
3) *SHA-256:* 6538e8175bbf3aab7b56603b5ca4ea5c2e898c7b4d818f2ca025d7fd477a13b7
4) *Vhash:* 0640466c0d5d5058z2dhz13z83z19z
5) *Imphash:* 5f5a1e6dd08eeda296f384994007c5ed

### C. Results from VirusTotal:

Upon taking the MD5 hash value of the executable and scanning it on VirusTotal, the resulsts showed that the winhlp.exe has been detected as malicious by 57 out 68 Anti-Virus engines, which is significantly a high number of engines. Following is the table that shoes the list of various Anti-Virus engines that flagged this as malicious[4]:

| Antivirus Engines | Malware Type |
|---|---|
| Ad-Aware | Generic.Keylogger.2.51F09BFA |
| Alibaba | Worm:Win32/SpyBot.97c29cf8 |
| BitDefenderTheta | AI:Packer.E4EB78041E |
| Cyberreason | Malicious.bc0dc9 |
| Cynet | Malicious (score: 100) |
| McAfee | W32/Spybot.worm.gen.a |
| Microsoft | Backdoor:Win32/Spybot.gen!B |
| Palo Alto Networks | Generic.ml |
| Rising | Worm.SpyBot!1.984D (CLASSIC) |
| TACHYON | Trojan-Spy/W32.KeyLogger.63008 |

| | |
|---|---|
| TrendMicro-HouseCall | WORM_SPYBOT.GEN1 |
| VIPRE | Trojan.Win32.Ircbot!cobra (v) |
| Webroot | W32.Malware.Gen |
| Ikarus | P2P-Worm.Win32.SpyBot |
| ClamAV | Win.Spyware.ot-2 |
| Cyren | W32/IRCBot-basedD_DET!Eldorado |
| ESET-NOD32 | A Variant Of Win32/SpyBot.APV |
| Yandex | Worm.P2P.SpyBot!8KnhkVNx64Q |
| Tencent | Win32.Worm-p2p.Spybot.Llho |
| VBA32 | Worm.SpyBot |
| Panda | Generic Malware |
| McAfee-GW-Edition | BehavesLike.Win32.PWSZbot.kh |
| Nano-Antivirus | Trojan.Win32.SpyBot.dwglso |
| CrowdStrike | Win/malicious_confidence_80% (W) |
| Cylane | Unsafe |
| Emsisoft | Generic.Keylogger.2.51F09BFA (B) |
| Lionic | Worm.Win32.Generic.kYNq |
| MaxSecure | Trojan.Malware.300983.susgen |
| Panda | Generic Malware |

Most the Antivirus vendors have identifying the executable as it has ability of either keylogger, spybot, backdoor and trojan.

*D. Extacting Strings*

Extracting the strings can be used to get clues about the functionality of malware and indicators of compromise that are associated with the binary. Strings are Unicode and ASCII chain of characters that are embedded in a file. Strings obtained from the binary can have references to URL, IP addresses, attacks commands, filenames, registry keys *etc.*, Although strings does not have the capability to provide a clear picture of a file, they can provide good leads about the malware and what it is capable of doing. The following ASCII strings extracted from the binary show its key logging capabilities:

*Keylogger logging to C:\WINDOWS\system32\keylog.txt*
*keylog.txt*
*Keylogger Started*
*HH:mm:ss]*
This indicates that the malware is logging the key strokes to file named keylog.txt, after it is launched it logs the time when it was started.
*Keylogger logging to %s*
*Keylogger active output to: DCC chat*
*Keylogger active output to: %s*
*error already logging keys to %s use "stopkeylogger" to stop startkeylogger*

DCC stands for Direct Client-to-Client which is an IRC related sub protocol that enables peers to interconnect by using IRC server in order to exchange files and/or perform non relay chats. The keylogger store the content in characters are %s is defined here. PRIVMSG %s :%s, suggests that the executable is trying to send private message to the server with no restrictions from the target. Upon execution the malware looks for open ports, once it is successful the target machine is connect to a Server named SpyBot1.2, as given in the ASCII strings
*Found poort %i open at ip:%s*
*PRIVMSG %s :Found poort %i open at ip:%s*.
The following ASCII string suggest that malware tries to fetch the information about the target machine, such as version, CPU, RAM, Operating System, uptime, Current User, IP address, Hostname,
*Version:%s cpu: %dMHz. ram: %dMB total, %dMB free %d%s in use os: Windows %s (%d.%d, build %d). uptime: %dd %dh %dm. Date: %s Time: %s Current user: %s IP address: %s Hostname: %s Wi*
*ndir: %s\ Systemdir: %s*

*HTTP/1.0 200 OK*
*Server: SpyBot1.2*
*Date: %s %s GMT*
*Content-Type: %s*
*Accept-Ranges: bytes*
*Last-Modified: %s %s GMT*
*Content-Length: %i*
*Connection: close*
The malware tries to establish a HTTP connection with the server named SpyBot1.2.

The malware also attempts to open CDAudio door and closes it, as represented in the following strings:
*set CDAudio door open*
*cd-rom drive opened*
*set CDAudio door closed*
*cd-rom drive closed*

Some more interesting strings were notices during the dynamic analysis such as *Process32Next*
*Process32First*
*CreateToolhelp32Snapshot*
*RegisterServiceProcess,* this set of strings indicates that the malware is gathering the name of the executable as it is writing to the log file. This is accomplished by using these functions.

*E. Imports and Exports*

Malware typically uses Windows API functions for interaction with the OS to perform process memory, filesystem and network operations. Windows exports most of its functions needed for these interactions in *Dynamic Link Library* files. Using the *Peid* Imports Viewer a list of all the imports was displayed, such as Wsock32.dll, SHELL32.dll, winmm.dll, KERNEL32.dll, USER32.dll, Advapi32.dll, CRTD.dll.

- *Kernel32.dll*: Kernel32.dll exports functions that are related to process, hardware, memory and filesystem operations. Malware imports functions from DLLs to perform these operations.
- *Wsock32.dl*l: functions for communicating on the network are stored in it, malware import functions from these DLLs to perform network-related tasks.
- *Advapi32.dll*: Malware uses API functions that are contained in this DLL to operations related to service and registry.
- *USER32.dll:* User32.dll implements functions which create and manipulate Windows user interface such as menus,, message boxes, etc. Some malwares use functions from this DLL to perform DLL injections and to monitor keystrokes and mouse events.
- *Winmm.dll:* winmm.dll is used for communicating or controlling multimedia devices.

### *F. Examinig Sections*

The content of the PE file is divided into sections, these sections represents code or data. It also represents they have in-memory attributes like read/write. The instructions that are executed by the processor are contained in the section. The section that has data represents different types of data such as import/export tables, read/write program data. Each section has a name that carries the purpose of the section. The given sample has four sections, .text, .bss, .data and .idata.

- .text: .text contains executable part of the code.
- .data: This section typically contains global variables and read/write data.
- .idata: This section is present if the executable contains import table.
- .bss: .bss section is used to store local common variable. When the program starts running the contents of this section are zeroed bytes.

### *G. File Dependecies and Imports Inspection*

To perform interaction with the file, registry, network, the malware depends on the functions by operating system. Windows exports most of the functions API (*Application Programming Interfaces)* that are required for interactions in *Dll (Dynamic Link Library)*[1]. Malware imports and call these functions from different DLLs to provide different functionality. The following table represents the summary of the potentially suspicious function calls and the Dll they are associated with:

| Dynamic Link Library | API Functions |
|---|---|
| Wsock32.dll | WSACleanup<br>connect<br>closesocket<br>listen<br>send<br>gethostbyname<br>getsocketname |
| SHELL32.DLL | ShellExecuteA |
| winmm.dll | mciSendStringA |
| Kernel32.dll | DeleteFileA<br>GetCommandLineA<br>GetSystemDirectoryA<br>GetProcAddress<br>CopyFileA, ReadFileA<br>CreateDirecetoryA<br>LoadLibraryA<br>CreateProcessA<br>CreateThread |
| USER32.dll | GetWindowsTextA<br>GetForegroundWindow<br>GetKeyState<br>GetAsynKeyState<br>MapVirtualkeyA<br>ExitWindowsEx |
| ADVAPI32.dll | GetUserNameA<br>RegCreateKeyA<br>RegSetValueExA<br>RegQueryValueExA<br>RegCloseKey |
| CRTDLL.Dll | Memcpy, malloc, memset, raise, signal, fopen |

## V. DYNAMIC ANALYSIS

To monitor the activities, effect on the system and how a sample interacts with the system, dynamic analysis is done, which involves analyzing the sample by executing it which is done in an isolated environment to prevent the host system from getting infected. Upon execution, the malware can interact with the system in different forms. For example, it can release a child process, drop files on the filesystem, create values for its persistence, create registry keys, take command and control server and it can also download other components. Dynamic malware analysis will help in getting a better understanding of the purpose of the malware. While performing the dynamic malware analysis, the different types of monitoring were carried out:

*A.Process Explorer:* Upon execution of winhlp_fixed.exe, a file named wind0ws.exe quicly appeared the the process explorer main menu wind0ws.exe which later changed to upwcfbg.exe. This executable was further investigated by right-clicking and selectin Properties for the upwcfbg.exe and then Strings which shows the strings, both in the executable image on **Disk** and **Memory**, the strings in the Memory “Keylogger logging to C:\WINDOWS\system32\keylog.txt” suggets that

this malware is keylogger, and it is logging key strokes to a file named keylog.txt. To test this, a short message was typed in the Notepad to check if this malware performs keylogging. By opening keylog.txt reveals that the keystrokes that were entered in Notepad. Below is the text that was captured by the keylogger.
[21:50:44] Untitled - Notepad hello work[Del[Num Lock]ld!!! (Changed window)

This also verifies the type of strings that are being captured along with the all the alphabet (lower case and upper case), as it can be seen from the strings such as[ Num Lock],[Down],[Right],[Up],[Left],[PgDn],[End],[Del],[Pg Up],[Home],[Insert],[ScrollLock],[PrintScreen],[WIN],[CTRL],[TAB],[F12],[F11],[F10],[F9],[F8],[F7],[F6],[F5],[F4],[F3],[F2],[F1],[ESC].

A. *FakeDNS:* As soon as the winhlp_fixed.exe was excuted, FakeDNS window shows that it attempts to connect to irc.efnet.org.

B. *TCPView:* Results from TCPview show that the malware is trying to establish a TCP connection from Local Address 192.168.21.128:1053 to a Remote Address 192.168.21.128:6667, the state of the connection stays on TIME_WAIT until it fails to connect and then disappears.

C. *Process Monitor*: During the dynamic malware analysis, the different types of monitoring was carried out:

- Process Monitoring: It involves the examining the properties and process activity of the result process
- File System Monitoring: It involves monitoring real time system activity
- Registry Monitoring: Register monitoring is done to monitor the register keys and registry data which is being written/read by the malware.
- Network Monitoring: The live traffic to the system and from the system is monitored in this process.

Utilizing the filters in Process Monitor, process name is winhlp_fixed.exe and upwcfbg.exe (the file that malware dropped on execution), all the activity by these processes can be seen. Process Monitor generates a huge number of events which makes it impossible to go through each event manually. To make it easier for the rest of the analysis in *process monitor,* the filters will be used.

- *Process create:*
  *Path and command line*
  *C:\WINDOWS\system32\upwcfbg.exe by winhlp_fixed.exe*

- *CreateFile:*
  *upwcfg.exe*
  *C:\WINDOWS\system32\keylog.txt*

*Generic write, Read Attributes and Delete* permissions are assigned to the file, which indicated that the malware has the ability to drop more than one malicious file with read and write privileges.

- *WriteFile:*
  *winhlp_fixed.exe*
  *C:\WINDOWS\system32\upwcfbg.exe*
  *upwcfg.exe*
  *C:\WINDOWS\system32\keylog.txt*

After the execution of winhlp.exe the malware deletes itself immediately, this indicates that the malware is trying to cover its tracks by deleting the file after compromising the system.

To maintain the presence of malware on the compromised machines even when the machine reboots, malware authors use various persistence mechanisms that allows the malware to stay on the compromised machines without having to reinfect it. The most common way of maintaining persistence is by adding an entry to run registry keys. The malware adds itself to different auto-start locations along with the ones mentioned below:

- *RegSetValue:*

*HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce*

*HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run*

- *RetCretaeKey:* winhlp_fixed.exe and upwcfg.exe

*HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run*

*HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce*

- *RegCloseKey:*

*HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run*

*HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce*

Windows store personal folders of each user in the keys that are mentioned below, this includes music, documents, downloads. However a malware can change this this behavior and load a malicious executable rather than a legitimate content. These registry keys were found with using filters as, *RegCreateKey, RegSetValue, RegQueryKey and regCloseKey:*

*HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer \Shell Folders*

*HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer \Shell Folders\Personal*
The above-mentioned keys are trying to read data from C:\Documents and Settings\Administrator\My Documents with desired access as maximum allowed.

Dynamic analysis is used in determining the network based and host-based indicators, it also validates the findings obtained in static analysis. The combination of both dynamic analysis and static analysis gives a good understanding of the malware. However, basic dynamic analysis has some limitations, and to get a deeper insight of the malware, code analysis or malware reverse engineering will be examined in the next section.

## VI. REVERSE ENGINEERING

Malware authors write the code in high-level language such as C++ or C, that is compiled into an executable using a compiler. During the analysis of a malware, the analysts only have the malicious executable without the source code. From the analysis done so far, it is safe determine that the malware has the capability of a keylogger, and it also has the capability of maintaining persistence, and the ability to take a snapshot of the running system. Malware carries out various operations, which includes various functionalities. In this section, the primary focus of the discussion will be on understanding how this malware uses various API functions to perform certain functionality.

A keylogger is malware which is designed to log keystrokes and intercept. Attackers use this functionality to steal confidential information such as passwords, credit card information usernames. The most common methods of logging key use Windows API functions which by either checking the key state or by installing Hooks.

Upon loading *winhlp_fixed.exe* in IDA Pro, *GetKeyState* can be seen in the names window. To determine the status of the *Shift* key whether it is up or down, the keylogger calls the *GetKeyState()* API at address 0x40CDC4. At address 0x40CDD0, it calls *GetAsynKeyState()* which is a part of loop and within each iteration of the loop, a virtual key code is passed as an argument that determines the status of key.[1] At address 0x40305B, a test operation is performed to determine if the most significant bit is set on the return value of *GetAsyncKeyState().* If it is set, it indicates that the key is being pressed. If a key is pressed, then the keylogger calls *getkeyState* at address 0x403062 that checks the status of Caps lock (14h converts to VK_CAPITAL in virtual key codes) [10]. By using this technique, a malware can determine if the upper-case, lower-case letter, number, or a special character was typed. At the end of the loop, malware iterates through 5Ch key codes that coverts to 92 key codes, which means that it monitors 92 keys. Var_4 acts as an index into array of key codes, which is incremented at the end of the loop, and loop is continued if the var_4 value is below 92. The subroutine at location *.text:00402FA5 push offset aSChangedWindow ; "%s (Changed window"* indicates that the string "%s(Changed window, the substituted strings are running process from *GetForegroundWindow* and ASCII values for the *GetAsynKeyState*. These values are written to DLL with a call to *GetWindowTextA.*

Malware can open handle to registry key by calling either *RegCreatekey* or *RegOpenKey.* After opening the handle to the Run registry by calling the *RegCreateKey*, the returned handle which is stored in phkResult variable is moved to eax register and is then passed as 1st parameter by *RegSetValueExA* at loc_402A4 in cross reference by sub_402933.. Malware is using the *RegSetValueEx* API to set value in registry key to maintain persistence. After adding entry, the malware closes the handle by passing the handle it had previously to the *RegClosekey* API function at loc_4029BC in cross reference by sub_402933.

The malware leaves no choice for the target process but to load a malicious DLL in its process memory space through the *LoadLibrary()* API. The kernell32.dll exports *LoadLibrary()* which takes a single argument and which is path to the DLL. The DLL pathname is copied to the target process so that later when the remote thread is created and when the *LoadLibrary* is called in the target process the DLL path will be passed as argument. Malware must determine the address oof *LoadLibrary* at location *loc_401433* in cross reference by sub_40129C in kernell32.dll which is done by calling *GetModuleHandle* API and then it passes kernel32.dll as argument. This returns the base address of kernel32.dll, once it receives the base address of kernel32.dll, the address of *LoadLibrary* is determines by calling *GetProcessAddress.* In the same loop code injection technique is being used. In this three API calls are being used in .data section, 1) CreateToolhelp32Snapshot, which is used to obtain the snapshot of all the running processes; 2) Process32First, which gets information about the first process in the snapshot; 3) Process32Next, is used loop to pass through all the processes. The Process32First and Process32Next APIs get the information about processes, such as process ID, the parent process ID, executable name. This information is used by the malware to determine whether it is target process. [2]

GetWindowsDirectory returns the file path to the directory, malware sometimes uses this to determine which directory to install additional malware program. GetWindowsDirectory is being called at two subtroutines, sub_404d75 and sub_406472. When the code executes, .text 00404F6F call GetWindowsDirectoryA pushes offset aDdMmmyyy, from

this it can be determined that the malwar is trying to store the date of execution to the windows directory. In the same loop the malware calls sub_409A21 to store aVersionSCpuDmh, which grabbing all the information about the running operating system, such as version, cpu, ram, the bulit of windows system along with the version, current user, hostname, windows directory, space available in memory, date and time.

The presence of API call such as ShellExecuteA suggests that the malware has the capability of invoking other programs. In this malware, the function ends at .text 004013D2 with a call to ExitProcess, that terminates the malware. The call to ShellExecuteA at .text 004013CB removes the malware from the disk by launching wind0ws.exe.

## VII. PATCHING

Malware often times contain defense mechanisms to prevent reverse-engineering. Analysts can fix these defense mechanisms by using patching technique. This requires an analyst to identify a defense mechanism that is used in the code, modify it and save as a new executable which is analyzed for further analysis. In this sample, it was observed that the malware sleeps for a certain period of time after execution. This may prevent an analyst from understanding the functionality of the malware while doing the dynamic analysis. As mentioned earlier, during the dynamic analysis we learnt that the malware is creating a file named C:\WINDOWS\system32\keylog.txt, also as noted in strings it was mentioned that Keylogger logging to C:\WINDOWS\system32\keylog.txt, from this we can determine that the malware will log keystrokes to the mentioned file path on execution. However, this was not the case when the program was executed. The malware was logging only the time when the keylogger started. This could mean that the malware sleeps for some time after the execution. To fix this, the malware executable was loaded into OllyDbg. At address 0x402921 the value PUSH 7530 is given, that means the malware sleeps 30000ms, and at address 0x402A76 the value is assigned as PUSH DWORD PTR [416690] which converts to 1800000 ms, that means it sleeps for 30 minutes. Modifying these PUSH instructions to PUSH1 and adding nops, this modified executable was saved as *winhlp_patched.exe*. The purpose of adding NOP instruction is that CPU does nothing, which can be useful when removing the original code, that means it just cannot be just deleted but it must be replaces with valid instructions. To verify if this works, the modified file was executed. Following are the results that were observed in the keylog.txt file.

[03:May:2022, 12:44:08] Keylogger Started
[12:44:55] Run r[WIN] (Changed window)
[12:45:09] Untitled - Notepad hello world!! (Changed window)

Looking at the timestamps of when the keylogger started and when it starts logging keystrokes, verifies that the malware was designed in a way to trick the analyst in believing that the malware just logs the time of execution in keylog.txt.

## VIII. MALWARE CLASSIFIACTION USING YARA RULES

A malware contains a number of strings and/or binary indicators that recognize the strings or binary that are unique to malware or a particular type of malware family that be used in malware classification. Malware analysts classify malwares based on these unique strings and binary indicators that are present in the binary. Malware analysts can create YARA rules based on the binary information that is contained in malware sample. These YARA rules contain a set of strings and Boolean expressions, that determines its logic. The YARA rule can be used to scan files using the YARA utility.

```
rule wind0ws

{
        meta:
                description = "detects the presence of keylogger"
                MD5 = "3b683acbc0dc96e3fbf8bca715333789"

        strings:
                $a = " Found poort %i open at ip:%s PRIVMSG %s "
                $b = " WNetEnumCachedPasswords"
                $c = " Keylogger active output to: DCC chat"
                $d = " MODE $CHAN +k spybot"

        condition:

                $a and $c and $d and $b

}
```

To perform the YARA rule scanning an analyst will need to use the yara rule mentioned above against the target machine or folder. If the YARA scan is successful, that should mean that the malware from the similar malware family is present on that system.

## IX. CONCLUSION

After analyzing the *winhlp.exe*, later saved as *winhlp_fixed.exe*, with various malware techniques, we observed that the given malware sample has extensive functionalities of a keylogger, along with retrieving system information, injecting DLL. First, it was identified that a keylogger was installed on the victim system and what type of information is being targeted. The malware also tries to transfer the collected information to a remote user.

## REFERENCES


[1] A. Monnappa, “Learning Malware Analysis- Explore the concepts, tools, and techniques to analyze and investigate Windowa malware”, June 2018

[2] S. Michael, H. Andrew, “Practical Malware Analysis- The hands on guide to dissecting malicious software”

[3] E. Chris, “The IDA Pro Book- the Unofficial Guide to World’s Most Popular Disassembler”

[4]VirusTotal homepage, https://www.virustotal.com/gui/file/6538e8175bbf3aab7b56603b5ca4ea5c2e898c7b4d818f2ca025d7fd477a13b7/details

[5] Microsoft Syinternals Process Explore, https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer

[6] Microsoft Sysinternals Process Monitor, https://docs.microsoft.com/en-us/sysinternals/downloads/procmon

[7] S. Megira, A. R. Pangesti, F. W. Wibowo, “Malware Detection And Analysis Using Reverse Engineering Technique”, https://iopscience.iop.org/article/10.1088/1742-6596/1140/1/012042/pdf

[8] B. Supreeth,”Reverse Engineering of Malware Eyeing the Future of Security”, https://etd.ohiolink.edu/apexprod/rws_etd/send_file/send?accession=akron1247447165&disposition=inline , August 2009.

[9] b. Ahmet, V. Dan, V. Jaramir, “Malware reverse Engineering Handbook”, https://ccdcoe.org/uploads/2020/07/Malware_Reverse_Engineering_Handbook.pdf

[10] https://github.com/Eazybright/Irvine32/blob/master/VirtualKeys.inc

[11] C. Raymond, “What is difference between GetKeyState and GetAsynKeyState” https://devblogs.microsoft.com/oldnewthing/20041130-00/?p=37173 November 30th, 2004