IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

STATIC ANALYSIS & IDENTIFICATION OF MALWARE IN SANDBOX ENVIRONMENT

[1]Supriti Banik,[2]Mr. Toufique Ahammad Gazi

[1]M.Tech Student,[2]Assistant Professor
[1]Computer Science & Engineering,
[1]Supreme Knowledge Foundation Group of Institutions, Hooghly, India

Abstract: Usage of internet is a daily routine nowadays. From office work, e-commerce, banking, education, business, entertainment, stock market, everywhere we depend abundantly on internet. The numbers of internet users are growing largely day by day and so does cyber crimes. Malware is like a weapon used for cyber crime. Analysis of malware is important to prevent malware's action in today's cyber world. This paper is written of static analysis and identification of malware in sandbox environment. Here, in windows 7 ultimate sandbox, static malware analysis is performed using some tools and the malware is identified. Also some details about the malware attached file type; characteristics and functions of the malware, which discover by tools; have been discussed in this research paper.

Index Terms - **Malware,static malware analysis,tools,method of analysis,results,discussion,identification.**

I. INTRODUCTION

Malware is the malicious software or file that is delivered over network for unlawful motive. There are multiple types of malware exist. Section 8 denotes about overview of literature of various researchers. Section 3 denotes about malware and its kinds. From section 4 and 5 malware analysis and variety of analyzing method are known. From data extraction of file through static malware analysis, we come to know about the file or folder acquired by us is malware infected or not. Section 6 defines about the usage of static analysis tools used in this research. Method of analysis performed in research is stated in section 7. In section 8, result of the whole analysis process is discussed. Conclusion and future work are in section 9. Acknowledgement is in section 10. At last the references, which have helped in performing this research by carrying a lot of useful information.

1.1 Objectives

The main objectives of this research are-

- Creating a sandbox environment.
- Downloading some static analysis tools.
- Using the tools, analysis the collected sample in various ways and identifying the malware type.
- Discussing about the characteristics and other details of this malware according to the analysis result.

II. OVERVIEW OF LITERATURE

Narender Solanki and Dr. Neetu Sharma [1] said that the static analysis can be done by some methods like; AV (antivirus) scanning using the hash values; which helps to identify the existing malware and it's nature; checking it's a packed malicious file or not and the packing tool, picking the strings hidden in malware binary. They proposed also the reverse engineering technique using IDA or Ollydbg tool to convert the machine readable language to human readable assembly language to know about the malware functionality.

Dolly Uppal, Vishakha Mehra and Vinod Verma [2] also gave proposal of some static analysis techniques like; reverse engineering, to built the source code from assembly code of software extracted from binary for finding it's maliciousness, file signature analysis for detecting malware, heuristic analysis etc. They expressed four ways of heuristic analysis. Firstly, file based, where the suspected file will undergo a deep analysis to observe the purpose, working process etc. Secondly, weight based, where every sample executable will be weighted by dangerous activity. If the weight value crosses the predefined limit, it will be known as malicious. Thirdly, rule based, where mismatch of some predefined rules, which was made by analyzer for defining the executable, tells about the maliciousness of the executable. Fourthly, generic signature based, where by checking the previous reports of antivirus engines using the signature of malware, can be exposed the new variant of old malware.

S Megira, A R Pangesti and F W Wibowo [3] performed their static analysis by using the tools; CFF explorer for Portable Executable (PE) header analysis, PEview for PE structure analysis, Virustotal for online scanning, IDA for code dissembling or reverse engineering and dependency walker for import analysis.

Syarif Yusirwan S, Yudi Prayudi and Imam Riadi [4] implemented basic static malware analysis by using such tools- Virustotal.com; Md5deep (hash calculating tool); three types of pack/unpack detection tool like, PEid, RDG packer, Exeinfo PE; D4dot (.net unpacker); PEview (structure displayer of PE). And for advanced techniques they used BinText, Dependency walker and IDA.

Omer ASLAN [5] created a comparison between the performances of some static malware analysis tools and malware detection tool in a virtual environment (Windows 7, 8.1 and 10). He used PEid, PEview, PEBrowser professional (disassembler), BinText , MD5deep, UPX, dependency walker, resource hacker, IDA pro, for analysis statically. Norton, Kaspersky, Avast, Bitdefender, Avira, ClamAV, McAfee- these are the most common offline anti-virus softwares, which are used for his research. As per his research, for new coming or unknown malware, static tools are better than anti-virus softwares.

III. What is Malware?

Malware or Malicious software is a program (code) or file, which is designed to hurt computer, network or server deliberately. The malware steals, encrypts and deletes sensitive facts. There are numerous sorts of malware exist. A number of them are computer virus, worm, Trojan horse, spyware, ransomware etc. Malware can infect the system automatically for vulnerable services of network. Downloading files, softwares and so on from web browser invites the malware. Cyber criminals intentionally send malicious files into the sufferer’s device with the aid of unsolicited mail or message. By means of clicking attachment of this form of mail or message spreads over the device.

- Virus: It's far one of the conventional type of malware which can execute by itself and spread by way of infecting other applications or files. Virus especially comes as an attachment of an email. After opening this, device is infected.
- Worm: It's like virus, but it can self-replicate without a host program and commonly spreads from machine to machine, without any interaction of user.
- Trojan Horse: it is a kind of malware is designed to get right of entry to a system as a legitimate software program. After installation, it is able to execute their malicious capabilities on owner’s data or device.
- Spyware: It is a type of malicious software, which installed secretly on device and collects user data from the device and observes user’s activities without their knowledge.[6]
- Ransomware: It is designed to infect user’s system and encrypt its data. After being inaccessible the system, cyber criminals demand money from the victim to restore the system to its previous state.
- Rootkit: It is a malicious software, designed to gain unauthorized access over a system [7].
- Keylogger: Keylogger, also called as keystroke logger, is a malware same like spyware. By monitoring every keystroke of user, steals important data [8].

IV. Malware Analysis

Malware analysis is a process by which all the details about the arrived malware of system like; its identity, characteristics, functions, purpose of use, how to defeat it; can be known. After finding the malicious file, it is analyzed by examining its code or by executing it in a safe controlled environment.

V. Types of Malware Analysis

To analyze malware, there are three ways - Static Malware Analysis, Dynamic Malware Analysis and Hybrid Malware Analysis.

5.1Static Malware Analysis

Analyze malicious software or file (malware) without executing its program, is called static malware analysis. This is the safest way of malware analysis because of not running of the malicious code does not infect the system. There are various ways to extract information from the sample like, examining the file format, string extracting, file fingerprinting, antivirus scanning etc. [9]

5.1.1 Antivirus Scan

Antivirus software scanning is one of the simplest ways to recognize the presence of malware. It’s an essential part of devices safety. But it needs to be updated on every time. Kaspersky, Avast etc. are some popular offline malware scanning and removing software. There are online scanners also available. VirusTotal is a kind of online malware scanning platform.

5.1.2 File Format

It is a data structure, which contains all of the important and necessary data required by the operating system to run the executable file. From here the analyst can known about the sample extension (like, .dll, .exe etc.), target operating system (like, PE file targets windows operating system). Besides it a lot of useful details also can be extracted from the sample like size, compilation date and time, strings, functions, libraries etc.

5.1.3 File Fingerprinting or Hashing

Every suspicious file has a signature. File fingerprinting or file identification is a method of identifying the signature of a binary file. Hash value is a completely unique fingerprint or identifier. Hash is displayed in hexadecimal format. It is used to know that the malware file is changed or not. By searching this value in any online malware scanning website like VirusTotal, Any.Run etc., we come about to know that the malware is previously analyzed or it is a fresh one. There is a variety of different types of hash exist. Some widely used hashes are described below.-

- MD5: MD5 is the Message Digest hashing algorithm of 5th iteration, which creates a 128bit hash value. It is also designed by Ronald Rivest in 1991.

- SHA1: SHA1 refers to Secure Hash Algorithm 1, which is a hashing algorithm produced by United States National Security Agency (NSA). It takes an input and generates 160 bit (20 bytes) hash value as output.
- SHA256: SHA256 is the Secure Hash Algorithm, a part of the SHA2 family of hashing algorithms. It was developed in 2001 by NSA. The hash value, produced by it is 256 bit.
- SHA512: SHA512 is also a part of SHA family, generates 512 bit hash value.[10]

5.1.4 String Extraction

It is a way of extracting readable words and characters from a suspicious file. Strings are made of ASCII (American Standard Code for Information Interchange) and Unicode (Universal Character Encoding) format. Strings consist of filenames, domain names, IP addresses, attacking command, URL (Uniform Resource Locators) etc. For an example; if a malware creates file, this filename will be stored as a string or the domain name, controlled by malware attacker will be stored as domain name in a string. So, extraction of strings reveals valuable data. Strings give a hint about what malware can do. There are also some garbage strings or unwanted strings are obtained in this process. These are used by cybercriminals to keep busy to analyst.

5.2Dynamic Malware Analysis

Analyze to a malware by executing the program in a safe or controlled environment and monitor its behaviour, is called dynamic malware analysis. For obfuscated malware, dynamic malware analysis is used. In this analysis, behaviour of the malware is monitored by monitoring API or system calls, which is used by malware. If programs are not executed in a safe environment by some isolated tools, then malicious programs can affect the system.[11]

5.3Hybrid Malware Analysis

The combination of both static and dynamic malware analysis is called hybrid malware analysis. When the malicious code is more sophisticated, then this type of process is used to examine malware.[12]

Static malware analysis, where malware sample is thoroughly examined by some tools, is performed in this research.

VI. Static Malware Analysis Tools

Some open source tools are used in this research for this static analysis process of malware. These are –

6.1Basic Static Analysis Tools

Table 1 Basic Static Analysis Tools

Tools	Usage
1. Exeinfo PE	It is a software, is used to understand that the sample is executable or not, packed or unpacked. It gives unpacking tips and also more information about the sample.
2. UPX	Cybercriminals use this tool to pack or compress the malicious file for hiding the original functions. Also the executable file can be decompressed by the help of UPX (Ultimate Packer for eXecutables).
3. Free Hex Editor Neo	It is used to extract and edit file’s binary in hexadecimal and ASCII format.
4. HashCalc	To calculate hash value (like, MD5, SHA-1, SHA-256), which is the identification mark or fingerprint of a file, this tool is used.
5. VirusTotal	This is a website, where suspicious URLs or files are analyzed by some antivirus engines and detect the malware or malicious contents.
6. Pestudio	It provides a lot of information about the suspected Portable Executable or PE file like some basic data, VirusTotal report, imports, libraries, strings, etc. which are maliciousness indicator.
7. BinText	It is a tool used for extracting string of a malware sample.

6.2Advanced Static Analysis Tool

Table 2 Advanced Static Analysis Tool

Tool	Usage
1. Dependency Walker	It is a tool shows the .DLLs and their functions, imported by malware.

VII. Method of Analysis

Step 1: To avoid accidental execution of the suspicious file, the first step of analyzing process is ‘Creating a Safe Environment’. So, for creating a sandbox, VMware Workstation Pro 17 Software is downloaded. This software supports installation of multiple operating systems at a time in a single computer (both Windows and Linux operating system).

Then Windows 7 Ultimate (64 bits) virtual machine (sandbox) is installed in it. There are more vulnerability is present in this version of windows operating system, than other version. This makes the process some easier by attracting the malware.

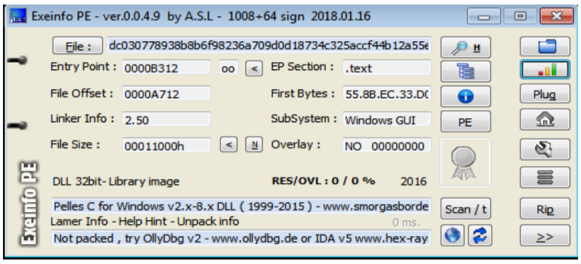
After installing this operating system, windows firewall and windows updates are turned off because both of them prevent malware to come in device. Then previously mentioned tools are downloaded and installed in this virtual machine. The UPX tool is placed in the system32 folder (storage of many sensitive system files). The system32 folder is present at the windows folder of the Local Disc (C:) or C drive in computer. The UPX tool is utilized by opening Windows PowerShell.

Step 2: When the set up is completed, then a sample of suspicious file is collected for static analysis. Here ‘dc030778938b8b6f98236a709d0d18734c325accf44b12a55ecc2d56b8bb9000’ is this sample. Due to malicious function, it is in a zipped folder and the folder is password protected. So, after extracting the information of zipped folder with the given password, it can be used. After extracting the mentioned sample file is then opened (it can also be dragged from its location and dropped in tool) in previously defined tool to analysis that it is malicious or not and if yes, then the type of malware.

VIII. Result and Discussion

8.1Exeinfo PE

After opening the sample file in this tool, it gives some important information as shown in the Fig.1 below.



Exeinfo PE - ver.0.0.4.9 by A.S.L - 1008+64 sign 2018.01.16

File : dc030778938b8b6f98236a709d0d18734c325accf44b12a55e

Entry Point : 0000B312 oo < EP Section : .text

File Offset : 0000A712 First Bytes : 55.8B.EC.33.D(

Linker Info : 2.50 SubSystem : Windows GUI

File Size : 00011000h < N Overlay : NO 00000000

DLL 32bit- Library image RES/OVL : 0 / 0 % 2016

Pelles C for Windows v2.x-8.x DLL (1999-2015) - www.smorgasborde

Lamer Info - Help Hint - Unpack info 0 ms.

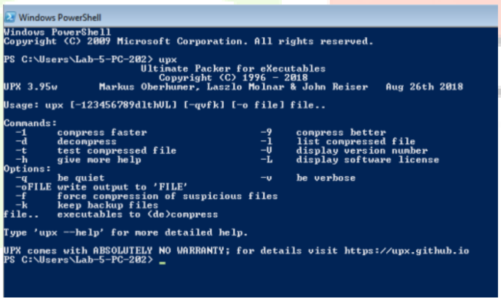
Not packed , try OllyDbg v2 - www.ollydbg.de or IDA v5 www.hex-ray

Figure 1 Sample analysis using Exeinfo PE

Tool confirms that the file isn’t a packed or compressed file. From this tool, it can be known that the suspicious sample file’s extension is .dll, it does imply it is a Dynamic Link Library file. A .dll file is an executable file although it is not directly executable however it contains executable codes. Its architecture is 32 bit and created on 2016. Subsystem is windows GUI (Graphical User Interface). Actually subsystem specifies the environment of the executable file. Exeinfo PE shows it’s a PE or Portable Executable file, which is made for windows operating system. The .text section of file contains the entry point (first bytes are 55 8B EC 33), the place in a program where the execution of the program begins. Exeinfo PE reveals more data about all sections. This tool gives a hint that in case of getting the assembly code from machine readable code of the file, debugging and disassembling tools like OllyDbg version 2 or IDA version 5 can be used. Or, Pelles C, a development kit consists of many editor, debugger etc., made for windows operating system, can be used.

8.2UPX

If the sample file was packed, the UPX tool could have been used to unpack it. Fig.2 shows various usage of this tool. For an instance, to decompress or unpack the packed file, after opening Windows PowerShell, we have to follow the following process-



Windows PowerShell

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Lab-5-PC-202> upx
                        Ultimate Packer for eXecutables
                           Copyright (C) 1996 - 2018
UPX 3.95w       Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

Usage: upx [-123456789dlthVL] [-qvfk] [-o file] file..

Commands:
  -1     compress faster                   -9    compress better
  -d     decompress                        -l    list compressed file
  -t     test compressed file              -V    display version number
  -h     give more help                    -L    display software license
Options:
  -q     be quiet                          -v    be verbose
  -oFILE write output to 'FILE'
  -f     force compression of suspicious files
  -k     keep backup files
file..   executables to (de)compress

Type 'upx --help' for more detailed help.

UPX comes with ABSOLUTELY NO WARRANTY; for details visit https://upx.github.io
PS C:\Users\Lab-5-PC-202> _
```

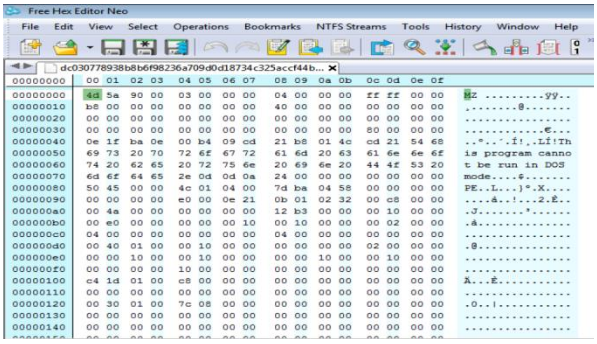
Figure 2 Usage of UPX tool

Type [upx -d (drag and drop the sample file from its current location)] - > Enter

File can also be packed (as the cybercriminals do) by this tool. The command is- upx -9 -qvfk (drag and drop the sample) -> Enter

8.3Free Hex Editor Neo

This tool extracts the file format or binary in hexadecimal and ASCII format. File format is a source of important information about sample file. From ExeinfoPE, it is able to be known that the sample file is a PE file.



```
00000000  00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
00000000  4d 5a 90 00  03 00 00 00  04 00 00 00  ff ff 00 00   MZ ........ÿÿ..
00000010  b8 00 00 00  00 00 00 00  40 00 00 00  00 00 00 00   ¸.......@.......
00000020  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
00000030  00 00 00 00  00 00 00 00  00 00 00 00  80 00 00 00   ............€...
00000040  0e 1f ba 0e  00 b4 09 cd  21 b8 01 4c  cd 21 54 68   ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70  72 6f 67 72  61 6d 20 63  61 6e 6e 6f   is program canno
00000060  74 20 62 65  20 72 75 6e  20 69 6e 20  44 4f 53 20   t be run in DOS 
00000070  6d 6f 64 65  2e 0d 0d 0a  24 00 00 00  00 00 00 00   mode....$.......
00000080  50 45 00 00  4c 01 04 00  7d ba 04 58  00 00 00 00   PE..L...}º.X....
00000090  00 00 00 00  e0 00 0e 21  0b 01 02 32  00 c8 00 00   ....à..!...2.È..
000000a0  00 4a 00 00  00 00 00 00  12 b3 00 00  00 10 00 00   .J.......³......
000000b0  00 e0 00 00  00 00 00 10  00 10 00 00  00 02 00 00   .à..............
000000c0  04 00 00 00  00 00 00 00  04 00 00 00  00 00 00 00   ................
000000d0  00 40 01 00  00 10 00 00  00 00 00 00  02 00 00 00   .@..............
000000e0  00 00 10 00  00 10 00 00  00 00 10 00  00 10 00 00   ................
000000f0  00 00 00 00  10 00 00 00  00 00 00 00  00 00 00 00   ................
00000100  c4 1d 01 00  c8 00 00 00  00 00 00 00  00 00 00 00   Ä...È...........
00000110  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
00000120  00 30 01 00  7c 08 00 00  00 00 00 00  00 00 00 00   .0..|...........
00000130  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
00000140  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
```

Figure 3 Malware file format by Free Hex Editor Neo

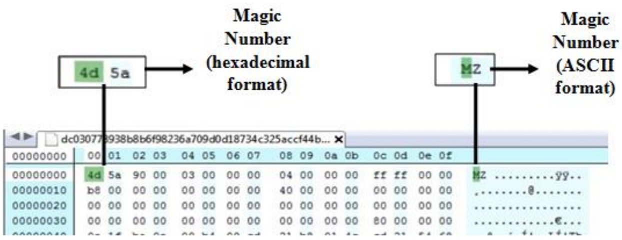
Here the table 3 shows the PE file structure [13] given in the PE file format.

Table 3 PE file structure

DOS Header / MZ Header
DOS Stub
PE / NT Header PE Signature PE File Header Optional Header •Data Directory
Section Table / Section Header .text section header •.text section .rdata section header •.rdata section .data section header •.data section .reloc section header •.reloc section

8.3.1 DOS Header / MZ Header

It contains File Signature or Magic Number, which express the file type or file extension. In the sample file, used in this research, the file signature is ‘4D 5A’ [14] in hexadecimal format and ‘MZ’ in ASCII format and size is 2 bytes. This number is present at the beginning of file format. So, the file is an executable file and the file extension will be .exe or .dll or their other variants.

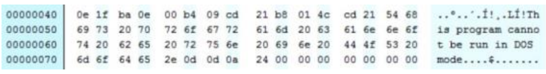

4d 5a
Magic Number (hexadecimal format)
MZ
Magic Number (ASCII format)
dc030778938b8b6f98236a709d0d18734c325accf44b...

```
00000000  00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
00000000  4d 5a 90 00  03 00 00 00  04 00 00 00  ff ff 00 00   MZ ........ÿÿ..
00000010  b8 00 00 00  00 00 00 00  40 00 00 00  00 00 00 00   ¸.......@.......
00000020  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00   ................
00000030  00 00 00 00  00 00 00 00  00 00 00 00  80 00 00 00   ............€...
```

Figure 4 Dos header/MZ header of malware file format and the Magic Number

8.3.2 DOS Stub

DOS stub is present in file format from offset 0×40 (00000040) to 0×7f (00000070). It is a simple program of 64 bytes, which prints ‘This program cannot be run in DOS mode’.



```
00000040 | 0e 1f ba 0e  00 b4 09 cd  21 b8 01 4c  cd 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050 | 69 73 20 70  72 6f 67 72  61 6d 20 63  61 6e 6e 6f  is program canno
00000060 | 74 20 62 65  20 72 75 6e  20 69 6e 20  44 4f 53 20  t be run in DOS 
00000070 | 6d 6f 64 65  2e 0d 0d 0a  24 00 00 00  00 00 00 00  mode....$.......
```

Figure 5 Dos stub of malware file format

8.3.3 PE Header / NT Header

PE header or NT header consists of 3 parts-

8.3.3.1 PE Signature

It confirms that the file is a PE file. ’50 45’ (offset 0×80) in hexadecimal format is the PE signature, which is present in the beginning of PE header and in ASCII format it is ‘PE’.

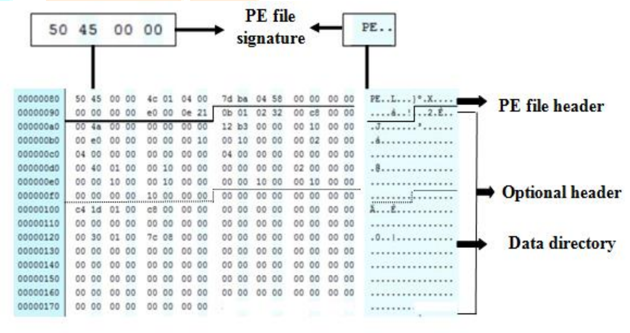
8.3.3.2 PE File Header

It carries some basic information about the file. It is present in the next 20 bytes of the PE signature. It’s also known as Common Object File Format (COFF) file header.

8.3.3.3 Optional Header

It also carries some useful details about the file. It is present after the PE file header in file format.

- Data Directory- Data directory, which is present in last some bytes of optional header, contains useful data about the executable file.

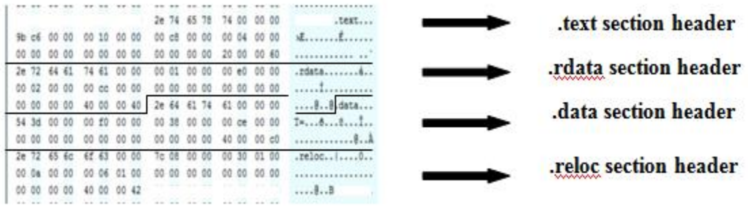


```
00000080 | 50 45 00 00  4c 01 04 00  7d ba 04 58  00 00 00 00  PE..L...}º.X....
00000090 | 00 00 00 00  e0 00 0e 21  0b 01 02 32  00 c8 00 00  ....à..!...2.È..
000000a0 | 00 4a 00 00  00 00 00 00  12 b3 00 00  00 10 00 00  .J.......³......
000000b0 | 00 e0 00 00  00 00 00 10  00 10 00 00  00 02 00 00  .à..............
000000c0 | 04 00 00 00  00 00 00 00  04 00 00 00  00 00 00 00  ................
000000d0 | 00 40 01 00  00 10 00 00  00 00 00 00  02 00 00 00  .@..............
000000e0 | 00 00 10 00  00 10 00 00  00 00 10 00  00 10 00 00  ................
000000f0 | 00 00 00 00  10 00 00 00  00 00 00 00  00 00 00 00  ................
00000100 | c4 1d 01 00  c8 00 00 00  00 00 00 00  00 00 00 00  Ä...È...........
00000110 | 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000120 | 00 30 01 00  7c 08 00 00  00 00 00 00  00 00 00 00  .0..|...........
00000130 | 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000140 | 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000150 | 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000160 | 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
00000170 | 00 00 00 00  00 00 00 00                            ........
```

Figure 6 PE header / NT header of malware file format

8.3.4 Section Table / Section Header

It is present after data directory from offset 0×170 in file format. It consists of .text section header, .rdata section header, .data section header, .reloc section header. These carry a lot of important information about four sections present in executable file.



```
                         2e 74 65 78  74 00 00 00          .text...
9b c6 00 00  00 10 00 00  00 c8 00 00  00 04 00 00  .Æ......È.......
00 00 00 00  00 00 00 00  00 00 00 00  20 00 00 60  ............ ..`
2e 72 64 61  74 61 00 00  00 01 00 00  00 e0 00 00  .rdata.......à..
00 02 00 00  00 cc 00 00  00 00 00 00  00 00 00 00  .....Ì..........
00 00 00 00  40 00 00 40  2e 64 61 74  61 00 00 00  ....@..@.data...
54 3d 00 00  00 f0 00 00  00 38 00 00  00 ce 00 00  T=...ð...8...Î..
00 00 00 00  00 00 00 00  00 00 00 00  40 00 00 c0  ............@..À
2e 72 65 6c  6f 63 00 00  7c 08 00 00  00 30 01 00  .reloc..|....0..
00 0a 00 00  00 06 01 00  00 00 00 00  00 00 00 00  ................
00 00 00 00  40 00 00 42                            ....@..B
```

Figure 7 Section table / Section header of malware file format

8.3.4.1 Sections

There are 4 sections are present in the file.

Table 4 Sections in the file format

2e 74 65 78 74 00 00 00	.text section
2e 72 64 61 74 61 00 00	.rdata section
2e 64 61 74 61 00 00 00	.data section
2e 72 65 6c 6f 63 00 00	.reloc section

i. .text section- It is present at first 8 bytes of .text section header. It contains executable code.

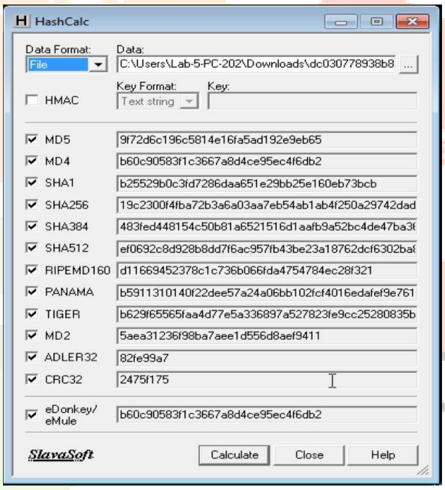
ii. .rdata section- It is present at first 8 bytes of .rdata section header. It contains read only data within the program. This data is globally accessible.

iii. .data section- It is present at first 8 bytes of .data section header. It contains data which is both readable and writable and accessed by the program.

iv. .reloc section- It is present at first 8 bytes of .reloc section header. It contains image relocation information.

8.4HashCalc

Hash calculator shortly HashCalc calculates the file’s Hash value as previously mentioned as tool list. Figure 8 shows different types of the hash values of the suspicious sample .dll file, calculated by this tool. Like MD5 hash-9F72D6C196C5814E16FA5AD192E9EB65, SHA256 hash- 19C2300F4FBA72B3A6A03AA7EB54AB1AB4F250A29742DAD. Any of hash value of the file can be taken for analyzing purpose in online antivirus scanning website. File names should not be used directly in online scanning platform because as a subscribed member of this platform, the attacker will know about the capture and analysis of his files. This may force him to turn to other methods of attack.



HashCalc

Data Format: File

Data: C:\Users\Lab-5-PC-202\Downloads\dc030778938b8

□ HMAC Key Format: Text string Key:

☑ MD5 9f72d6c196c5814e16fa5ad192e9eb65

☑ MD4 b60c90583f1c3667a8d4ce95ec4f6db2

☑ SHA1 b25529b0c3fd7286daa651e29bb25e160eb73bcb

☑ SHA256 19c2300f4fba72b3a6a03aa7eb54ab1ab4f250a29742dad

☑ SHA384 483fed448154c50b81a6521516d1aafb9a52bc4de47ba3

☑ SHA512 ef0692c8d928b8dd7f6ac957fb43be23a18762dcf6302ba

☑ RIPEMD160 d11669452378c1c736b066fda4754784ec28f321

☑ PANAMA b5911310140f22dee57a24a06bb102fcf4016edafef9e761

☑ TIGER b629f65565faa4d77e5a336897a527823fe9cc25280835b

☑ MD2 5aea31236f98ba7aee1d556d8aef9411

☑ ADLER32 82fe99a7

☑ CRC32 2475f175

☑ eDonkey/ eMule b60c90583f1c3667a8d4ce95ec4f6db2

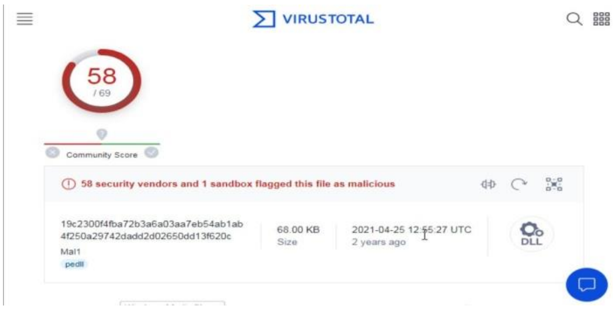
Figure 8 Different types of hashes using HashCalc

VirusTotal website is used for online scanning of the sample file in this research.

8.5VirusTotal

VirusTotal, a free online tool, is used to analysis suspected files, Urls or IP addresses or domains. It provides a report of many of antivirus engines and website scanners, which detect the sample file as malware or a threat. VirusTotal platform also displays the file type, file size, last analysis time of file, several types of file name given by many researchers etc.

In this research, one of hash value, the MD5 (9F72D6C196C5814E16FA5AD192E9EB65) is submitted in VirusTotal website for analysis. Here the figure 12 shows that the sample is a DLL file, sized 68 kb, which is malicious, confirmed by 58 security vendors and one sandbox, out of 69. The last analysis of this file is in 25th April, 2021 at the time of 12:55:27 UTC (Coordinated Universal Time), its mean the malware is not a new one. The SHA256 hash is displaying in the Fig.9, which is19c2300f4fba72b3a6a03aa7eb54ab1ab4f250a29742dadd2d02650dd13f620c. Mal1 is a name of this file, which was given by other researchers.



58 security vendors and 1 sandbox flagged this file as malicious

19c2300f4fba72b3a6a03aa7eb54ab1ab4f250a29742dadd2d02650dd13f620c
Mal1
pedll

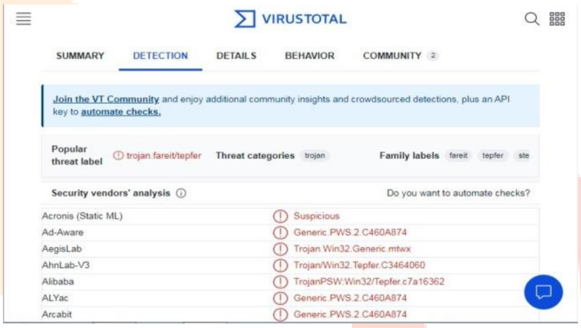
68.00 KB
Size

2021-04-25 12:55:27 UTC
2 years ago

Figure 9 Various types of security vendors flagged the sample as malicious

As per antivirus engines' report in VirusTotal after analysis, the threat in the file is Trojan Horse (or Trojan) malware. Maximum engines confirmed that the malware, present in the sample file is fareit or tepfer or ste, which are the variants of trojan.

- Fareit- It is a malware belongs to trojan family, used to steal password and sensitive data of user and send it to hacker. By clicking on malicious link, attached with spam mail or message, this malware affects the user computer.
- Tepfer- Like fareit, it also belongs to trojan family, steals personal data like accounts details of cloud storage or email id, cookies data from infected computer.
- Ste- It is a malware of trojan family, performs malicious actions in victim's computer in cybercriminal's direction.
  Figure 10 displays some of the antivirus engines that analyzed the file, used in this research.



Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label: trojan.fareit/tepfer
Threat categories: trojan
Family labels: fareit tepfer ste

Security vendors' analysis — Do you want to automate checks?

Acronis (Static ML)	Suspicious
Ad-Aware	Generic.PWS.2.C460A874
AegisLab	Trojan.Win32.Generic.mtwx
AhnLab-V3	Trojan/Win32.Tepfer.C3464060
Alibaba	TrojanPSW:Win32/Tepfer.c7a16362
ALYac	Generic.PWS.2.C460A874
Arcabit	Generic.PWS.2.C460A874

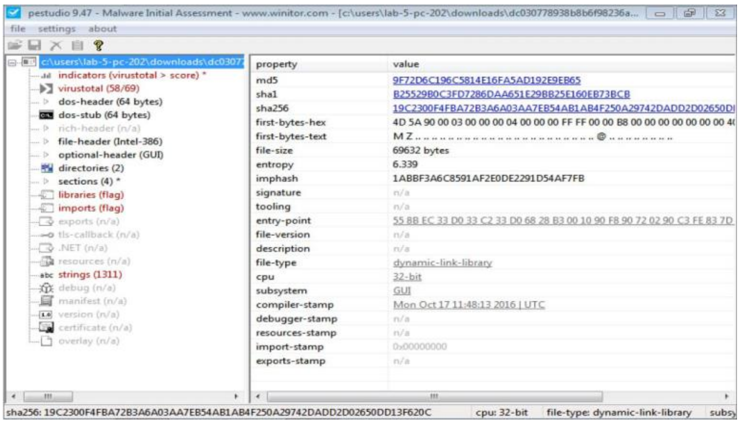
Figure 10 Some of antivirus engines and website scanners analysis the sample file

VirusTotal reveals the different types of hash like MD5, SHA-1, SHA-256 etc and also imphash, which is used to identify malware binary of same family. The file is a Portable Executable file, its target is Microsoft Windows (GUI) machine of 32 bit Intel 80386 microprocessor or later compatible version of this processor. The compilation (or creation) date was 17th October, 2016 at 11:48:13 UTC. VirusTotal displays the entry point of the malicious dll file, which is 45842 and gives some basic details like name, virtual address, raw size, virtual size, entropy level, MD5 hash value about the four sections present in file.

The imported DLLs as per VirusTotal report, are urlmon.dll, wininet.dll, kernel32.dll, wsock32.dll, advapi32.dll, ole32.dll, shlwapi.dll, user32.dll, userenv.dll (will be discussed below). Besides these VirusTotal gives some additional information of submitted file.

8.6Pestudio

Pestudio is a free tool used to analyze a windows executable binary statically. After dropping the sample file in this tool, it displays many useful data about file using some tabs.



pestudio 9.47 - Malware Initial Assessment - www.winitor.com - [c:\users\lab-5-pc-202\downloads\dc030778938b8b6f98236a...

c:\users\lab-5-pc-202\downloads\dc03077
- indicators (virustotal > score) *
- virustotal (58/69)
- dos-header (64 bytes)
- dos-stub (64 bytes)
- rich-header (n/a)
- file-header (Intel-386)
- optional-header (GUI)
- directories (2)
- sections (4) *
- libraries (flag)
- imports (flag)
- exports (n/a)
- tls-callback (n/a)
- .NET (n/a)
- resources (n/a)
- strings (1311)
- debug (n/a)
- manifest (n/a)
- version (n/a)
- certificate (n/a)
- overlay (n/a)

property	value
md5	9F72D6C196C5814E16FA5AD192E9EB65
sha1	B25529B0C3FD7286DAA651E29BB25E160EB73BCB
sha256	19C2300F4FBA72B3A6A03AA7EB54AB1AB4F250A29742DADD2D02650DI
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 4(
first-bytes-text	M Z .. .. @ .. ..
file-size	69632 bytes
entropy	6.339
imphash	1ABBF3A6C8591AF2E0DE2291D54AF7FB
signature	n/a
tooling	n/a
entry-point	55 8B EC 33 D0 33 C2 33 D0 68 28 B3 00 10 90 F8 90 72 02 90 C3 FE 83 7D
file-version	n/a
description	n/a
file-type	dynamic-link-library
cpu	32-bit
subsystem	GUI
compiler-stamp	Mon Oct 17 11:48:13 2016 | UTC
debugger-stamp	n/a
resources-stamp	n/a
import-stamp	0x00000000
exports-stamp	n/a

sha256: 19C2300F4FBA72B3A6A03AA7EB54AB1AB4F250A29742DADD2D02650DD13F620C cpu: 32-bit file-type: dynamic-link-library subsy

Figure 11 Malware analysis using Pestudio

The above Fig.11 of pestudio is representing some the initial information about research sample. The widely used hashes (md5, sha-1, sha-256), the first bytes of malware binary in hexadecimal & ASCII format, which contains the file signature (4D 5A in hexadecimal and MZ in ASCII, described previously), file size, entropy value (6.339), entry point of the code, imphash value, CPU architecture (32 bit), subsystem (Windows GUI), compiler stamp (compilation time and date) are revealed by pestudio. Here, entropy is the measure of the random nature of file's data. Higher entropy value signifies more randomness. A file with high entropy value may be malicious.

The all extracted data from the tabs of pestudio are-

8.6.1 Indicators tab

This tab highlights data within the sample file that may be malicious. Table is representing that pestudio tool has identified some indicators and classified them on a scale of 1-3. Those marked as 1 are definitely highly malicious indicators.

Table 5 Malicious indicators listed by Pestudio

Indicators	Detail	Level
VirusTotal score	58 out 69	**1**
3 URLs	http://reninparwil.com/zapoy/gate.php	**1**
	http://leftthenhispar.ru/zapoy/gate.php	
	http://reptertinrom.ru/zapoy/gate.php	
3 Libraries	OLE32 Extensions for Win32	**1**
	Internet Extensions for Win32 Library	
	Windows Socket 32-Bit Library	
Imports	30	**1**
File Hash	19C2300F4FBA72B3A6A03AA7EB54AB1A...	3
File Size	69632 bytes	3
File Subsystem	GUI	3
13 APIs	services, security, rdp, network, cryptography, execution, setup, file, reconnaissance, memory, dynamic-library, registry, exception	3
Imphash	1ABBF3A6C8591AF2E0DE2291054AF7FB	3

8.6.2 Virustotal tab

This tab highlights some antivirus engines or website scanners (58/69) that already analyzed the file and identified as malicious. The dates of every analysis are also mentioned.

8.6.3 Dos-header tab

This tab displays the hash values, size, entropy value and file ratio of dos-header in malware binary. It also shows the location of file header.

8.6.4 Dos-stub tab

This tab shows the hash values, size, entropy value, file ratio and the error message (!This program cannot be run in DOS mode) of dos-stub in malware binary.

8.6.5 File-header tab

This tab shows some characteristics of file header revealed by pestudio, as shown in table below-

Table 6 Characteristics of file header in Pestudio

Property	Value	Detail
Dynamic-link-library	0×2000	true
32-bit word support	0×0100	true
File-can-be-executed	0×0002	true
Line-stripped-from-file	0×0004	true
Local-symbols-stripped-from-file	0×0008	true

Pestudio also gives some general information like compiler stamp (Mon Oct 1711:48:13, 2016 UTC), optional header's size (224 bytes), PE signature (PE00, here 00 means null), the PE signature value is 0×00004550, targeted machine (Intel 386), number of sections (4) etc.

8.6.6 Optional header tab

This tab highlights some general information about optional header of the sample file format written in table below-

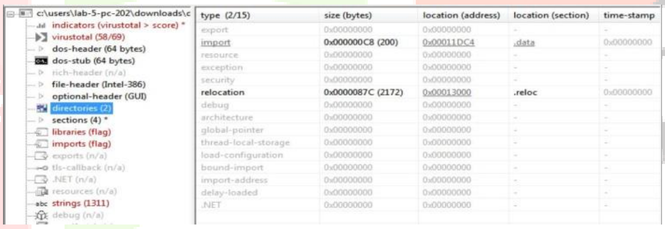
Table 7 Optional header details in Pestudio

Property	Value	Detail
Subsystem	0×0002	GUI
Magic	0×010B	PE
File checksum	0×00000000	0×0001EA07 (expected)
Entry point	0×0000B312	section: .text
Base of code	0×00001000	section: .text
Base of data	0×0000E000	section: .rdata
Size of code	0×0000C800	51200 bytes
Size of initialized data	0×00004A00	18944 bytes
Size of image	0×00014000	81920 bytes
Size of headers	0×00001000	4096 bytes
Size of stack reserve	0×00100000	1048576 bytes
Size of stack commit	0×00001000	4096 bytes
Size of heap reserve	0×00100000	1048576 bytes
Size of heap commit	0×00001000	4096 bytes
Section alignment	0×00001000	4096 bytes
File alignment	0×00000200	512 bytes
Directories count	0×00000010	16
Image base	0×10000000	0×10000000
Linker version	2.50	2.50
OS version	4.0	Windows NT 4.0
Subsystem version	4.0	4.0

Here, in the table, the Subsystem is GUI (Graphical User Interface) version of 4.0, is required to run the file. File is a Portable Executable file as previously said. Entry point location, where the executable code is present, is .text section and the beginning of data is present in .rdata section. Initialized data means the writable data. The image base specifies the preferred address to which the executable must be mapped to memory.

8.6.7 Directories tab

The Fig.12 shows this tab, where 2 directories and their basic details are highlighted. The directories are import and relocation. Import directory is present in .data section and relocation is in .reloc section.



type (2/15)	size (bytes)	location (address)	location (section)	time-stamp
export	0x00000000	0x00000000	-	-
import	0x000000C8 (200)	0x00011DC4	.data	0x00000000
resource	0x00000000	0x00000000	-	-
exception	0x00000000	0x00000000	-	-
security	0x00000000	0x00000000	-	-
relocation	0x0000087C (2172)	0x00013000	.reloc	0x00000000
debug	0x00000000	0x00000000	-	-
architecture	0x00000000	0x00000000	-	-
global-pointer	0x00000000	0x00000000	-	-
thread-local-storage	0x00000000	0x00000000	-	-
load-configuration	0x00000000	0x00000000	-	-
bound-import	0x00000000	0x00000000	-	-
import-address	0x00000000	0x00000000	-	-
delay-loaded	0x00000000	0x00000000	-	-
.NET	0x00000000	0x00000000	-	-

Figure 12 Pestudio is highlighting directories

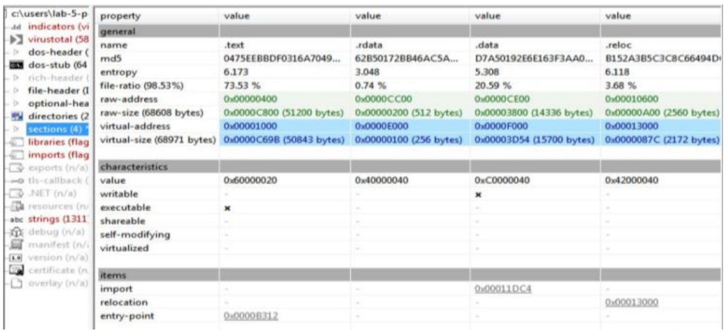
8.6.8 Section tab

This tab shows the sections present in the sample file and their details. Figure 13 is highlighting that-

i. The executable code is present in .text section. Its mean this section has execute permission. This section contains entry point information.

ii. The writable data is present in .data section. Its mean this section has write permission. This section contains import information.

iii. The .reloc section contains relocation information.



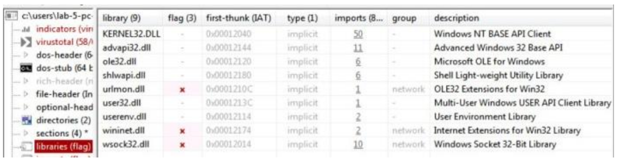
property	value	value	value	value
general				
name	.text	.rdata	.data	.reloc
md5	0475EEBBDF0316A7049...	62B50172BB46AC5A...	D7A50192E6E163F3AA0...	B152A3B5C3C8C66494D
entropy	6.173	3.048	5.308	6.118
file-ratio (98.53%)	73.53 %	0.74 %	20.59 %	3.68 %
raw-address	0x00000400	0x0000CC00	0x0000CE00	0x00010600
raw-size (68608 bytes)	0x0000C800 (51200 bytes)	0x00000200 (512 bytes)	0x00003800 (14336 bytes)	0x00000A00 (2560 bytes)
virtual-address	0x00001000	0x0000E000	0x0000F000	0x00013000
virtual-size (68971 bytes)	0x0000C69B (50843 bytes)	0x00000100 (256 bytes)	0x00003D54 (15700 bytes)	0x0000087C (2172 bytes)
characteristics				
value	0x60000020	0x40000040	0xC0000040	0x42000040
writable	-	-	x	-
executable	x	-	-	-
shareable	-	-	-	-
self-modifying	-	-	-	-
virtualized	-	-	-	-
items				
import	-	-	0x00011DC4	-
relocation	-	-	-	0x00013000
entry-point	0x0000B312	-	-	-

Figure 13 Details of every section in Pestudio

8.6.9 Libraries tab

Libraries help to recognize the capabilities of the sample. This tab consists of the libraries, which are flagged (blacklisted) as malicious by pestudio tool. This are-

i. Urlmon.dll
ii. Wininet.dll
iii. Wsock32.dll



c:\users\lab-5-pc-
indicators (vir
virustotal (58/
dos-header (6
dos-stub (64 b
rich-header (n
file-header (In
optional-head
directories (2)
sections (4) *
libraries (flag)

library (9)	flag (3)	first-thunk (IAT)	type (1)	imports (8...	group	description
KERNEL32.DLL	-	0x00012040	implicit	50	-	Windows NT BASE API Client
advapi32.dll	-	0x00012144	implicit	11	-	Advanced Windows 32 Base API
ole32.dll	-	0x00012120	implicit	6	-	Microsoft OLE for Windows
shlwapi.dll	-	0x00012180	implicit	6	-	Shell Light-weight Utility Library
urlmon.dll	×	0x0001210C	implicit	1	network	OLE32 Extensions for Win32
user32.dll	-	0x0001213C	implicit	1	-	Multi-User Windows USER API Client Library
userenv.dll	-	0x00012114	implicit	2	-	User Environment Library
wininet.dll	×	0x00012174	implicit	2	network	Internet Extensions for Win32 Library
wsock32.dll	×	0x00012014	implicit	10	network	Windows Socket 32-Bit Library

Figure 14 Blacklisted libraries are highlighted by Pestudio

The above Fig.14 shows the blacklisted libraries. These libraries will be described below.

8.6.10 Imports tab

Imports are saved in libraries. Pestudio flagged some imports (30) with their libraries, which are the malicious indicator of sample file, are listed with their functions [15] [16] below.-

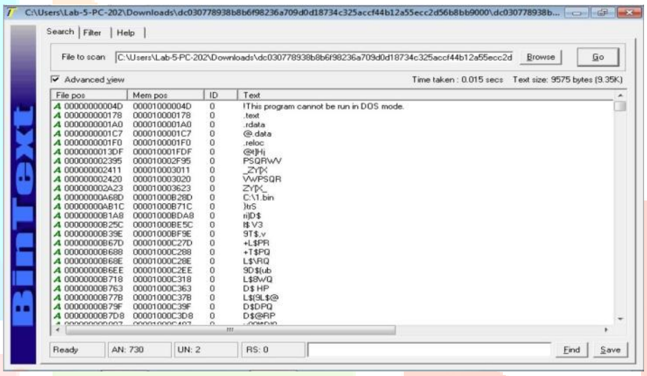
Table 8 Flagged 30 imports by Pestudio

Imports	Flag	Group	Usage	Library
LoadUserProfileA	×	Security	When a user logs on, the system automatically loads the user's profile. If a file or software tries to log in user profile illegally, the system does no longer load the profile. Then the file or software loads the user's profile by using LoadUserProfileA function.	userenv.dll
UnloadUserProfile	×	Security	This function is used to unload a user profile, which was loaded via the LoadUserProfileA function.	userenv.dll
GetPrivateProfileSectionName	×	Registry	This function is used to recover the names of every the sections of an initialization file, which stores the settings of operating system.	Kernel32.dll
RegCreateKeyA	×	Registry	The RegCreateKeyA function is used to create a specific registry key. Already existed key in registry can be opened by using this.	advapi32.dll
RegSetValueExA	×	Registry	This function is used to set the data and type of a value under a registry key.	advapi32.dll
RegOpenCurrentUser	×	Registry	It is used to open a specific registry key when the profile of user is already loaded by impersonated user.	advapi32.dll
ObtainUserAgentString	×	Network	The malware uses ObtainUserAgentString function to retrieve the user agent.	urlmon.dll
InternetCrackUrlA	×	Network	Malware uses this function to break (crack) a URL into parts.	wininet.dll
InternetCreateUrlA	×	Network	Malware uses this function to create a URL from its broken (cracked) parts.	wininet.dll
inet addr	×	Network	It is the function used in networking purpose.	wsock32.dll
gethostbyname	×	Network	This function is used to retrieve host information.	wsock32.dll
socket	×	Network	It is used to create a new socket.	wsock32.dll
connect	×	Network	This function is used to connect to a remote socket. Malware frequently uses low level functionality to hook up with a command and control server. It is usually used by malware to communicate with their command and control server.	wsock32.dll
closesocket	×	Network	It is used to close a socket.	wsock32.dll
send	×	Network	It is used to send data to a socket.	wsock32.dll
select	×	Network	This function is used to control any process like an original user.	wsock32.dll
recv	×	Network	This function is used to control any process like an original user.	wsock32.dll
setsockopt	×	Network	It is used to control socket behaviour.	wsock32.dll
WSAStartup	×	Network	It initializes the use of winsock.dll for calling program.	wsock32.dll
WriteFile	×	File	This function is used to write data to a specified file or input/output device.	Kernel32.dll
MapViewOfFile	×	File	This function is used to gain access to the memory-mapped object. A memory-mapped file holds the	Kernel32.dll

			contents of a file in the virtual memory.	
UnmapViewOfFile	×	File	It is used to unmap the map view.	Kernel32.dll
DeleteFileA	×	File	This function is used to destroy the data completely.	Kernel32.dll
FindFirstFileA	×	File	It is used to search via a directory and calculate the file system.	Kernel32.dll
FindNextFileA	×	File	It is used to search via a directory and calculate the file system.	Kernel32.dll
CreateToolhelp32Snapshot	×	Execution	CreateToolhelp32Snapshot is used to create a snapshot of processes, threads and modules. This function is typically used by malware to enumerate processes before process injection.	Kernel32.dll
Process32First	×	Execution	This function is used to retrieve the data about the first process of a system snapshot.	Kernel32.dll
OpenProcess	×	Execution	The OpenProcess function is used to open an existing local process object.	Kernel32.dll
Process32Next	×	Execution	This function is used to retrieve the data about the next process of a system snapshot.	Kernel32.dll
SetCurrentDirectoryA	×	-	By calling this function a file comes to know that, which directory is current.	Kernel32.dll

8.7BinText

Its extracted strings carry a more information about sample. Figure 15 is showing how BinText works.


C:\Users\Lab-5-PC-202\Downloads\dc030778938b8b6f98236a709d0d18734c325accf44b12a55ecc2d56b8bb9000\dc030778938b...
Search
Filter
Help
BinText
Browse
Go
Find
Save

File to scan: C:\Users\Lab-5-PC-202\Downloads\dc030778938b8b6f98236a709d0d18734c325accf44b12a55ecc2d

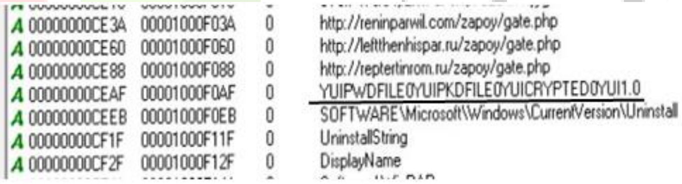
☑ Advanced view   Time taken : 0.015 secs   Text size: 9575 bytes (9.35K)

```
File pos          Mem pos           ID   Text
A 00000000004D    00001000004D      0    !This program cannot be run in DOS mode.
A 000000000178    000010000178      0    .text
A 0000000001A0    0000100001A0      0    .rdata
A 0000000001C7    0000100001C7      0    @.data
A 0000000001F0    0000100001F0      0    .reloc
A 0000000013DF    000010001FDF      0    @t]Hj
A 000000002395    000010002F95      0    PSQRWV
A 000000002411    000010003011      0    _ZY[X
A 000000002420    000010003020      0    VWPSQR
A 000000002A23    000010003623      0    ZY[X_
A 00000000A68D    00001000B28D      0    C:\1.bin
A 00000000AB1C    00001000B71C      0    }trS
A 00000000B1A8    00001000BDA8      0    ri)D$
A 00000000B25C    00001000BE5C      0    I$ V3
A 00000000B39E    00001000BF9E      0    9T$,v
A 00000000B67D    00001000C27D      0    +L$PR
A 00000000B688    00001000C288      0    +T$PQ
A 00000000B68E    00001000C28E      0    L$\RQ
A 00000000B6EE    00001000C2EE      0    9D$(ub
A 00000000B718    00001000C318      0    L$8WQ
A 00000000B763    00001000C363      0    D$ HP
A 00000000B77B    00001000C37B      0    L$(9L$@
A 00000000B79F    00001000C39F      0    D$DPQ
A 00000000B7D8    00001000C3D8      0    D$@RP
```

Ready | AN: 730 | UN: 2 | RS: 0

Figure 15 Some strings extracted by BinText

From the extracted strings an unorganized line is discovered, which obviously indicates that the sample is malicious also the malware is Trojan Horse.



```
A 00000000CE3A    00001000F03A    0    http://reninparwil.com/zapoy/gate.php
A 00000000CE60    00001000F060    0    http://leftthenhispar.ru/zapoy/gate.php
A 00000000CE88    00001000F088    0    http://reptertinrom.ru/zapoy/gate.php
A 00000000CEAF    00001000F0AF    0    YUIPWDFILE0YUIPKDFILE0YUICRYPTED0YUI1.0
A 00000000CEEB    00001000F0EB    0    SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
A 00000000CF1F    00001000F11F    0    UninstallString
A 00000000CF2F    00001000F12F    0    DisplayName
```

Figure 16 Unorganized line is showing in the list of strings

This line is- YUIPWDFILEOYUIPKDFILEOYUICRYPTEDOYUI1.0, shown in figure 16. The line shows some hidden words and a number- PWD FILE, PKD FILE, CRYPTED, 1.0.

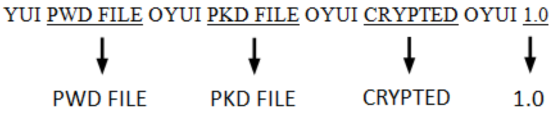


Figure 17 Discovering the hidden words present in line

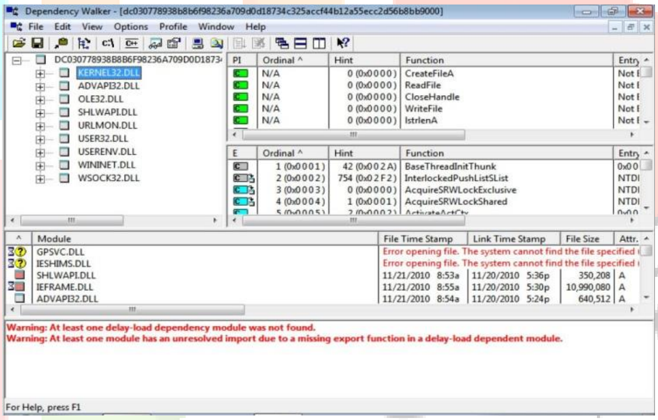
Here PWD FILE means .DOC file & PKD FILE means VIDEO file or MP4 file. A CRYPT file is the file which is encrypted via CryptXXX [17] virus. This CryptXXX virus is a Trojan Horse, used by cybercriminals, typically entered through spam email attachments, fake updates, downloads etc. This virus is a type of ransomware because it is used for demanding ransom forcefully by encrypting victim’s data. Mainly victim’s .DOC or .MP4 file is encrypted to prevent access. CRYPT files became conventional in 2016. Version of the virus is 1.0.

Some functions [18] in the strings, which are may be used by malware are listed below-

- AdjustTokenPrivileges: AdjustTokenPrivileges function is utilized to enable or disable particular access privileges. Malware uses it to gain supplemental sanction in a process injection attack.
- Accept: This function is utilized to listen for incoming connections. Malware uses it to communicate with their command and conversation server in many cases.
- CertOpenSystemStore: It is utilized to access the certificates stored on the local system.
- CreateFileA: It is used to create a new file or open an already existing file.
- CreateFileMappingA: CreateFileMappingA is the function, that is used to create a handle to a file mapping that loads a file into memory and makes it accessible via memory addresses. Loaders, launchers, and injectors use this to read and modify the PE files.
- CreateProcessAsUserA: It is used to create a new process. If malware creates a new process, it must be analyzed as well.
- GetProcAddress: This is utilized to retrieve the address of a function in a DLL loaded into memory.
- GetTempPathA: It returns the temporary file path. If malware call this function, it must be examined that whether it reads or writes any files in the temporary file path.
- GetModuleHandleA: This function is used to gain a handle to an already loaded module. Malware may use it to find and modify code in a loaded module or to look for a good location to inject code.
- GetVersionExA: This function returns data about which version of windows is currently running. This can be used to select between different objects for undocumented structures that have modified between different versions of windows.
- GetWindowsDirectoryA: It returns the file path to the windows directory. Sometimes malware use this function to decide that in which directory, additional malicious codes can be installed.
- ImpersonateLoggedOnUser: This function is used to impersonate some other user to run the process. The attackers ensure that the impersonated user has all the essential permissions to run the process.

8.8Dependency Walker

The dependency walker exposes those DLLs on which the runnable sample file depends on. The imported DLLs and their functions, of the research sample are listed and described below. Figure 18 is highlighting those DLLs. The usages of DLLS [19] are discussed below.-



Dependency Walker - [dc030778938b8b6f98236a709d0d18734c325accf44b12a55ecc2d56b8bb9000]
File
Edit
View
Options
Profile
Window
Help

DC030778938B8B6F98236A709D0D1873
KERNEL32.DLL
ADVAPI32.DLL
OLE32.DLL
SHLWAPI.DLL
URLMON.DLL
USER32.DLL
USERENV.DLL
WININET.DLL
WSOCK32.DLL

PI	Ordinal ^	Hint	Function	Entry
C	N/A	0 (0x0000)	CreateFileA	Not I
C	N/A	0 (0x0000)	ReadFile	Not I
C	N/A	0 (0x0000)	CloseHandle	Not I
C	N/A	0 (0x0000)	WriteFile	Not I
C	N/A	0 (0x0000)	lstrlenA	Not I

E	Ordinal ^	Hint	Function	Entry
C	1 (0x0001)	42 (0x002A)	BaseThreadInitThunk	0x00
C	2 (0x0002)	754 (0x02F2)	InterlockedPushListSList	NTDI
C	3 (0x0003)	0 (0x0000)	AcquireSRWLockExclusive	NTDI
C	4 (0x0004)	1 (0x0001)	AcquireSRWLockShared	NTDI
C	5 (0x0005)	2 (0x0002)	ActivateActCtx	0x00

^	Module	File Time Stamp	Link Time Stamp	File Size	Attr.
?	GPSVC.DLL	Error opening file. The system cannot find the file specified			
?	IESHIMS.DLL	Error opening file. The system cannot find the file specified			
	SHLWAPI.DLL	11/21/2010 8:53a	11/20/2010 5:36p	350,208	A
	IEFRAME.DLL	11/21/2010 8:55a	11/20/2010 5:30p	10,990,080	A
	ADVAPI32.DLL	11/21/2010 8:54a	11/20/2010 5:24p	640,512	A

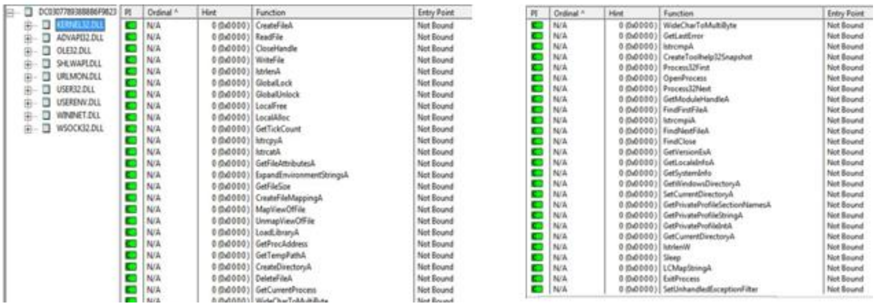
Warning: At least one delay-load dependency module was not found.
Warning: At least one module has an unresolved import due to a missing export function in a delay-load dependent module.

For Help, press F1

Figure 18 Imported DLLs by malware disclosed by Dependency Walker

8.8.1 KERNEL32.DLL

Normally, Kernel32.dll, a 32 bit dynamic link library file, is the core part of operating system. It manages the memory operations, (i/o) input/output operations, process interruption etc. If the kernel32.dll is found in import list of a file, its means that the file wants to get access on the user's computer. So, the sample must be a malicious file.



DC030778938B8B6F9823
KERNEL32.DLL
ADVAPI32.DLL
OLE32.DLL
SHLWAPI.DLL
URLMON.DLL
USER32.DLL
USERENV.DLL
WININET.DLL
WSOCK32.DLL

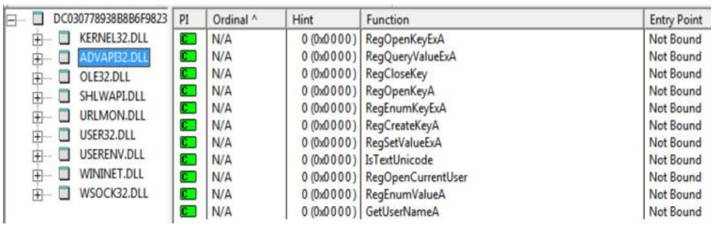
PI	Ordinal ^	Hint	Function	Entry Point
C	N/A	0 (0x0000)	CreateFileA	Not Bound
C	N/A	0 (0x0000)	ReadFile	Not Bound
C	N/A	0 (0x0000)	CloseHandle	Not Bound
C	N/A	0 (0x0000)	WriteFile	Not Bound
C	N/A	0 (0x0000)	lstrlenA	Not Bound
C	N/A	0 (0x0000)	GlobalLock	Not Bound
C	N/A	0 (0x0000)	GlobalUnlock	Not Bound
C	N/A	0 (0x0000)	LocalFree	Not Bound
C	N/A	0 (0x0000)	LocalAlloc	Not Bound
C	N/A	0 (0x0000)	GetTickCount	Not Bound
C	N/A	0 (0x0000)	lstrcpyA	Not Bound
C	N/A	0 (0x0000)	lstrcatA	Not Bound
C	N/A	0 (0x0000)	GetFileAttributesA	Not Bound
C	N/A	0 (0x0000)	ExpandEnvironmentStringsA	Not Bound
C	N/A	0 (0x0000)	GetFileSize	Not Bound
C	N/A	0 (0x0000)	CreateFileMappingA	Not Bound
C	N/A	0 (0x0000)	MapViewOfFile	Not Bound
C	N/A	0 (0x0000)	UnmapViewOfFile	Not Bound
C	N/A	0 (0x0000)	LoadLibraryA	Not Bound
C	N/A	0 (0x0000)	GetProcAddress	Not Bound
C	N/A	0 (0x0000)	GetTempPathA	Not Bound
C	N/A	0 (0x0000)	CreateDirectoryA	Not Bound
C	N/A	0 (0x0000)	DeleteFileA	Not Bound
C	N/A	0 (0x0000)	GetCurrentProcess	Not Bound
C	N/A	0 (0x0000)	WideCharToMultiByte	Not Bound

PI	Ordinal ^	Hint	Function	Entry Point
C	N/A	0 (0x0000)	WideCharToMultiByte	Not Bound
C	N/A	0 (0x0000)	GetLastError	Not Bound
C	N/A	0 (0x0000)	lstrcmpA	Not Bound
C	N/A	0 (0x0000)	CreateToolhelp32Snapshot	Not Bound
C	N/A	0 (0x0000)	Process32First	Not Bound
C	N/A	0 (0x0000)	OpenProcess	Not Bound
C	N/A	0 (0x0000)	Process32Next	Not Bound
C	N/A	0 (0x0000)	GetModuleHandleA	Not Bound
C	N/A	0 (0x0000)	FindFirstFileA	Not Bound
C	N/A	0 (0x0000)	lstrcmpiA	Not Bound
C	N/A	0 (0x0000)	FindNextFileA	Not Bound
C	N/A	0 (0x0000)	FindClose	Not Bound
C	N/A	0 (0x0000)	GetVersionExA	Not Bound
C	N/A	0 (0x0000)	GetLocaleInfoA	Not Bound
C	N/A	0 (0x0000)	GetSystemInfo	Not Bound
C	N/A	0 (0x0000)	GetWindowsDirectoryA	Not Bound
C	N/A	0 (0x0000)	SetCurrentDirectoryA	Not Bound
C	N/A	0 (0x0000)	GetPrivateProfileSectionNamesA	Not Bound
C	N/A	0 (0x0000)	GetPrivateProfileStringA	Not Bound
C	N/A	0 (0x0000)	GetPrivateProfileIntA	Not Bound
C	N/A	0 (0x0000)	GetCurrentDirectoryA	Not Bound
C	N/A	0 (0x0000)	lstrlenW	Not Bound
C	N/A	0 (0x0000)	Sleep	Not Bound
C	N/A	0 (0x0000)	LCMapStringA	Not Bound
C	N/A	0 (0x0000)	ExitProcess	Not Bound
C	N/A	0 (0x0000)	SetUnhandledExceptionFilter	Not Bound

Figure 19 Dependent functions of kernel32.dll in Dependency Walker

8.8.2 ADVAPI32.DLL

Advapi32 (Advanced Application Programming Interface 32 bit) is the dll; which is used to get access on advanced functionalities. For example; computer restarting or shutting down; starting, creating or stopping a process; windows registry etc.



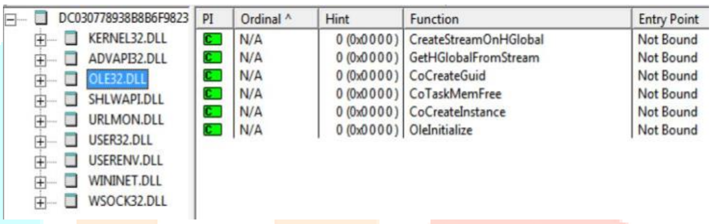
- DC030778938B8B6F9823
  - KERNEL32.DLL
  - ADVAPI32.DLL
  - OLE32.DLL
  - SHLWAPI.DLL
  - URLMON.DLL
  - USER32.DLL
  - USERENV.DLL
  - WININET.DLL
  - WSOCK32.DLL

PI	Ordinal ^	Hint	Function	Entry Point
C	N/A	0 (0x0000)	RegOpenKeyExA	Not Bound
C	N/A	0 (0x0000)	RegQueryValueExA	Not Bound
C	N/A	0 (0x0000)	RegCloseKey	Not Bound
C	N/A	0 (0x0000)	RegOpenKeyA	Not Bound
C	N/A	0 (0x0000)	RegEnumKeyExA	Not Bound
C	N/A	0 (0x0000)	RegCreateKeyA	Not Bound
C	N/A	0 (0x0000)	RegSetValueExA	Not Bound
C	N/A	0 (0x0000)	IsTextUnicode	Not Bound
C	N/A	0 (0x0000)	RegOpenCurrentUser	Not Bound
C	N/A	0 (0x0000)	RegEnumValueA	Not Bound
C	N/A	0 (0x0000)	GetUserNameA	Not Bound

Figure 20 Dependent functions of advapi32.dll

8.8.3 OLE32.DLL

Ole32.dll is a dll, which is used for Object Linking and Embedding in computer operating system. Object linking and embedding is a feature, which allows the user to create document in other application or modify them.



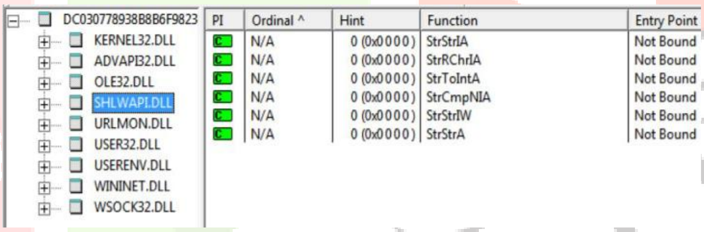
- DC030778938B8B6F9823
  - KERNEL32.DLL
  - ADVAPI32.DLL
  - OLE32.DLL
  - SHLWAPI.DLL
  - URLMON.DLL
  - USER32.DLL
  - USERENV.DLL
  - WININET.DLL
  - WSOCK32.DLL

PI	Ordinal ^	Hint	Function	Entry Point
C	N/A	0 (0x0000)	CreateStreamOnHGlobal	Not Bound
C	N/A	0 (0x0000)	GetHGlobalFromStream	Not Bound
C	N/A	0 (0x0000)	CoCreateGuid	Not Bound
C	N/A	0 (0x0000)	CoTaskMemFree	Not Bound
C	N/A	0 (0x0000)	CoCreateInstance	Not Bound
C	N/A	0 (0x0000)	OleInitialize	Not Bound

Figure 21 Dependent functions of ole32.dll

8.8.4 SHLWAPI.DLL

Shlwapi.dll (Shell Light Weight Utility Library) which manages important settings like; URL paths, registry settings etc.



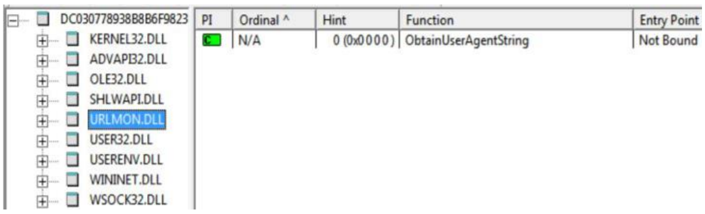
- DC030778938B8B6F9823
  - KERNEL32.DLL
  - ADVAPI32.DLL
  - OLE32.DLL
  - SHLWAPI.DLL
  - URLMON.DLL
  - USER32.DLL
  - USERENV.DLL
  - WININET.DLL
  - WSOCK32.DLL

PI	Ordinal ^	Hint	Function	Entry Point
C	N/A	0 (0x0000)	StrStrIA	Not Bound
C	N/A	0 (0x0000)	StrRChrIA	Not Bound
C	N/A	0 (0x0000)	StrToIntA	Not Bound
C	N/A	0 (0x0000)	StrCmpNIA	Not Bound
C	N/A	0 (0x0000)	StrStrIW	Not Bound
C	N/A	0 (0x0000)	StrStrA	Not Bound

Figure 22 Dependent functions of shlwapi.dll

8.8.5 URLMON.DLL

It is used at the time of object linking and embedding operation.



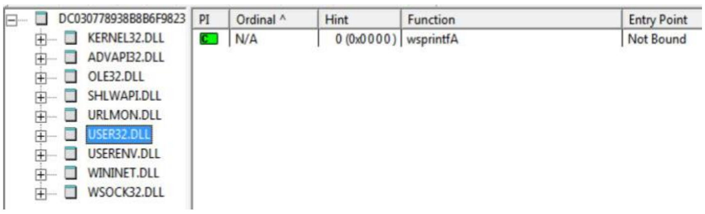
- DC030778938B8B6F9823
  - KERNEL32.DLL
  - ADVAPI32.DLL
  - OLE32.DLL
  - SHLWAPI.DLL
  - URLMON.DLL
  - USER32.DLL
  - USERENV.DLL
  - WININET.DLL
  - WSOCK32.DLL

PI	Ordinal ^	Hint	Function	Entry Point
C	N/A	0 (0x0000)	ObtainUserAgentString	Not Bound

Figure 23 Dependent functions of urlmon.dll

8.8.6 USER32.DLL

User32.dll is a dll file where graphical elements of windows like, windows, dialogue box are stored.



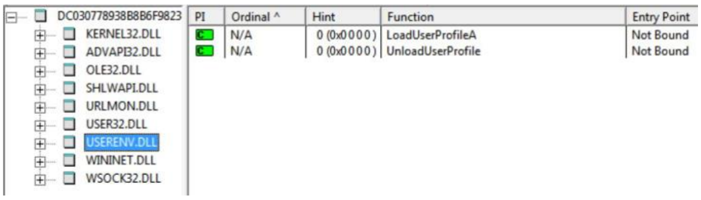
- DC030778938B8B6F9823
  - KERNEL32.DLL
  - ADVAPI32.DLL
  - OLE32.DLL
  - SHLWAPI.DLL
  - URLMON.DLL
  - USER32.DLL
  - USERENV.DLL
  - WININET.DLL
  - WSOCK32.DLL

PI	Ordinal ^	Hint	Function	Entry Point
C	N/A	0 (0x0000)	wsprintfA	Not Bound

Figure 24 Dependent functions of user32.dll

8.8.7 USERENV.DLL

It is used for creating or managing user profile.



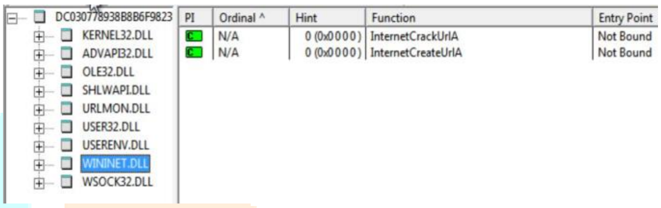
- DC030778938B8B6F9823
  - KERNEL32.DLL
  - ADVAPI32.DLL
  - OLE32.DLL
  - SHLWAPI.DLL
  - URLMON.DLL
  - USER32.DLL
  - USERENV.DLL
  - WININET.DLL
  - WSOCK32.DLL

PI	Ordinal ^	Hint	Function	Entry Point
🟩	N/A	0 (0x0000)	LoadUserProfileA	Not Bound
🟩	N/A	0 (0x0000)	UnloadUserProfile	Not Bound

Figure 25 Dependent functions of userenv.dll

8.8.8 WININET.DLL

It is used to access internet using HTTP (Hypertext Transfer Protocol) or FTP (File Transfer Protocol).



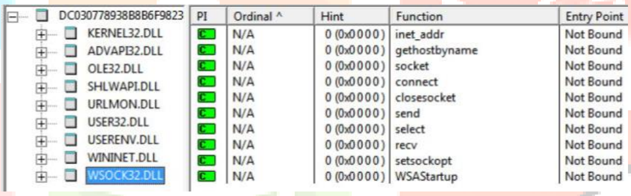
- DC030778938B8B6F9823
  - KERNEL32.DLL
  - ADVAPI32.DLL
  - OLE32.DLL
  - SHLWAPI.DLL
  - URLMON.DLL
  - USER32.DLL
  - USERENV.DLL
  - WININET.DLL
  - WSOCK32.DLL

PI	Ordinal ^	Hint	Function	Entry Point
🟩	N/A	0 (0x0000)	InternetCrackUrlA	Not Bound
🟩	N/A	0 (0x0000)	InternetCreateUrlA	Not Bound

Figure 26 Dependent functions of wininet.dll

8.8.9 WSOCK32.DLL

Wsock32.dll is used at the time of TCP/IP network communication.



- DC030778938B8B6F9823
  - KERNEL32.DLL
  - ADVAPI32.DLL
  - OLE32.DLL
  - SHLWAPI.DLL
  - URLMON.DLL
  - USER32.DLL
  - USERENV.DLL
  - WININET.DLL
  - WSOCK32.DLL

PI	Ordinal ^	Hint	Function	Entry Point
🟩	N/A	0 (0x0000)	inet_addr	Not Bound
🟩	N/A	0 (0x0000)	gethostbyname	Not Bound
🟩	N/A	0 (0x0000)	socket	Not Bound
🟩	N/A	0 (0x0000)	connect	Not Bound
🟩	N/A	0 (0x0000)	closesocket	Not Bound
🟩	N/A	0 (0x0000)	send	Not Bound
🟩	N/A	0 (0x0000)	select	Not Bound
🟩	N/A	0 (0x0000)	recv	Not Bound
🟩	N/A	0 (0x0000)	setsockopt	Not Bound
🟩	N/A	0 (0x0000)	WSAStartup	Not Bound

Figure 27 Dependent functions of wsock32.dll

IX. CONCLUSION AND FUTURE WORK

In today's digital world, malware is the one of big threat. So, the malware analysis is the biggest part of cyber security. To protect important virtual data of everywhere, malware analysis is necessary. This research paper is written, based on various types of static analysis where the detailed information about analysis of file's maliciousness, identification of the malware and its characteristics are present. Knowledge about characteristics of malware prevents to get infected by malware. But malware authors are developing new types of malware and new strategy of cyber attack day by day. The malware researchers are trying to get more efficient way to prevent malware attack. This research will continue in future and malware analysis will be done using reverse engineering and dynamic approach.

X. ACKNOWLEDGMENT

My deep of gratitude and thanks to all the sir and madam of the department of Computer Science & Engineering of Supreme Knowledge Foundation Group of Institutions, for their generous guidance, continuous encouragement, and supervision during the preparation of this research paper. Their support helped me to write this paper successfully.

REFERENCES

[1] Narender Solanki, Dr. Neetu Sharma, May 2019. Malware Analysis: Types & Tools. International Journal of Engineering Science and Computing, 9(5): 22664-22667.

[2] Dolly Uppal, Vishakha Mehra and Vinod Verma, February 2014. Basic survey on Malware Analysis, Tools and Techniques. International Journal on Computational Sciences & Applications (IJCSA), 4(1): 103-112, DOI: 10.5121/ijcsa.2014.4110.

[3] S Megira, A R Pangesti and F W Wibowo, December 2018. Malware Analysis and Detection Using Reverse Engineering Technique. Journal of Physics: Conference Series, 1140(1): 1-12, id.012042, DOI:10.1088/1742-6596/1140/1/012042.

[4] Syarif Yusirwan S, Yudi Prayudi, Imam Riadi, May 2015. Implementation of Malware Analysis using Static and Dynamic Analysis Method. International Journal of Computer Applications, 117(6): 11-15, ISSN: 0975-8887.

[5] Omer ASLAN, 25-26 November, 2017. Performance Comparison of Static Malware Analysis Tools Versus Antivirus Scanners To Detect Malware. International Multidisciplinary Studies Congress, Akdeniz University, Antalya/Turkey, 1-6.

[6] Meet Parekh, Gaurav Kulkarni, Aug 2021. A Survey on "Malware Analysis Techniques, its Detection and Mitigation.". International Research Journal of Engineering and Technology (IRJET), 8(8): 512-514, e-ISSN: 2395-0056, p-ISSN: 2395-0072.

[7] Aru Okereke Eze and Chiaghana Chukwunonso E., Jul–Aug 2018. Malware Analysis and Mitigation in Information Preservation. Journal of Computer Engineering (IOSR-JCE), 20(4): 53-62, e-ISSN: 2278-0661, p-ISSN: 2278-8727.

[8] Swathi Edem, Jan-March 2019. A Study on the Malware Analysis with Machine Learning Methods. International Journal of Research and Analytical Reviews (IJRAR), 6(1): i564-i569, e-ISSN: 2348-1269, p-ISSN: 2349-5138.

[9] Nirav Bhojani, October 2014. Malware Analysis. Conference: Ethical Hacking, Nirma University, DOI: 10.13140/2.1.4750.6889.

[10] https://en.m.wikipedia.org

[11] Aziz Makandar, Anita Patrot, 2015. Overview of Malware Analysis and Detection. International Journal of Computer Applications, National Conference of Knowledge, Innovation in Technology and Engineering (NCKITE), 35-40, ISSN: 0975-8887.

[12] Felina Simon Menezes, Felomina Jancy, Hanan Saleem Baji, Ponica J, March 2022. Malware Detection and Analysis. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), 2(2): 220-225, DOI: 10.48175/568, ISSN (Online) 2581-9429.

[13] Sundeep Varma, Jonnadula Narasimharao, June 2022. Malware Analysis with Machine Learning: Classifying Malware based on PE Header. International Journal for Research in Applied Science & Engineering Technology (IJRASET), 10(VI): 3583-3590, ISSN: 2321-9653.

[14] https://handwiki.org/wiki/List of file signatures

[15] https://learn.microsoft.com

[16] https://gist.github.com

[17] https://fileinfo.com/extension/crypt

[18] https://resources.infosecinstitute.com/topic/windows-functions-in-malware-analysis-cheat-sheet-part-1/

[19] https://www.silurian.com/inspect/imports.htm