Feature Extraction and Analysis of Portable Executable Malicious File

Manojkumar T kamble
Department of computer science
Karnatak university, Dharwad
Dharwad, India
Manojkumarkamble6@gmail.com

Dr. Sridevi
Professor, Department of computer science
Karnatak university, Dharwad
Dharwad, India
devisris@yahoo.com

2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA) | 978-1-6654-5834-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCSEA54677.2022.9936121

Abstract: **Malware is intentionally developed software to harm the working of a computer system. Modern malware is designed with mutation, and encryption characteristics, which increases a large number of different kinds of malware samples every day and these features made malware more active and robust against antivirus software. The portable executable file is a format or data structure of executable, DLL, object, and other files which are used by the Windows operating system. PE file format has a set of information that is used by windows operating system machines for executing the files. The PE file format has a header part and this part acts as metadata for all the tables because this header consists of all the details about the data which is stored in the PE file. The malware authors target the PE file header for spreading maliciousness and hacking the particular system. Malware uses these portable executable files for storing and spreading malicious contents. Most benign or malicious files use portable executable file format for storing and executing their .exe and other supporting files in the windows computer system. So to overcome all these problems, in this paper, dissection, and features have been extracted from malicious PE files.**

Keywords**:** ***benign, encryption, malicious, malware, mutation, portable executable (PE) file.***

I. INTRODUCTION

Malware is computer software that is designed to interrupt system functions, retrieve data, or personal computer devices. Malware is defined by its malevolent behavior, which goes against the wishes of the computer user. The goal is to get access to a computer and network resources, disrupt system control, and capture personal data without the system's owner's authorization [1]. Malware's proliferation had an impact on everyday life, from the government to the digital realm. Malware comes in a variety of forms, including Trojan horses, rootkits, backdoors, botnets, viruses, worms, spyware, and adware.[1].

The file format for executables, DLLs, object files, and other file types is a portable executable format [2]. Windows operating systems employ the Portable executable file structure. For the execution of files and data, the system relied on PE files. The PE file format is divided into subparts such as PE header, DOS stub, PE signature, image optional header, data dictionaries, and sections. The[2] PE header itself contains the metadata that has information about the PE file structure itself. Whenever malware attaches to the PE file that malware harms the system very frequently and Corrupts the whole system activity. The second-generation malware is the most effective malware on PE files because this malware uses features like encryption and obfuscation techniques. Because of this reason second-generation malware is not detected by the traditional antivirus software.

The PE file's structure shown in below Fig.1, begins with the PE header, which gives details required by the operating system to launch the programs. All the PE file starts with the MZ character. The DOS STUB is used for compatibility and it prints" program cannot be run in DOS mode". The PE signature contains the actual signature of that PE file which is used in identification. The image optional header stores the subsystem and entry point of the PE file. The section table contains the information about how to load the executable into memory and at the end of the format is SECTIONS which contains the data which is used by the executables.

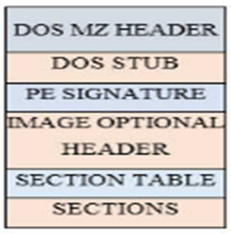


Fig. 1. The basic structure of the PE file

II. MALWARE ANALYSIS TECHNIQUES

A. Static analysis

Static analysis is the process of evaluating harmful software without running it. Signature-based malware detection technique identifies the malware whose signature is already in the database, if any advanced generation malware enters the system then that malware cannot be identified as malicious by anti-virus software. String fingerprint, binary n-grams, lexical module reference, sequence diagram, and opcode relative frequency are some of the identification features utilized in static analysis. Before doing static analysis, the program must be unpacked and decoded [1].

B. Dynamic analysis

Dynamic analysis is the process of observing the behavior of malevolent content as it is being performed in a safe environment. The dynamic analysis of malware is better than static analysis because it is not based on the database matching strategy, it is based on the dynamic feature analysis, extraction, and making decisions on whether the given sample is malware or not [1]. Before doing dynamic analysis, some necessary tools, software, and a safe environment should be created to extract the more relevant features [1].

C. Memory forensics:

Memory forensics is a broader area of forensics studies. There are different types of forensics are there in forensics science like digital, computer, memory, fingerprint, RAM, hard disk, external device, network, malware forensics, and many more. Computer forensics is the process of extracting digital information from computer memory to solve cyber forensics problems. We can store information on hard disks and physical memory. All activities of the user which is performed through a computer are recorded in both hard disk and volatile memory. Malware is a software program that affects the system activity by its malicious behavior. Whenever malware is introduced into the system then it must be gone through random access memory for its execution. So these executed malware traces can be extracted from the memory using memory forensics.

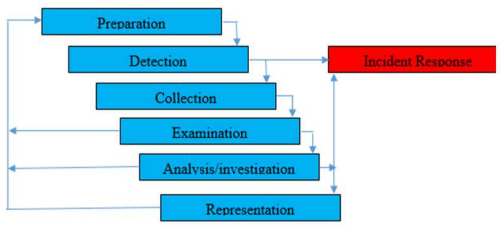


Fig. 2. The process model for malware forensics[3].

III. RELATED WORK

Malware researchers are working continuously to study the evolution of malware and its change in features and behaviors. In 2015 [4], the author proposed the Portable executable program malicious code detection based on the Chi-square test. In this, the author used PE file header features for malware detection and chi-square score and phi coefficients as strategies for deciding the features. 552 PE files were collected in that 338 malware 214 benign programs [4].

In 2016[5], the author acquired a data collection that included 3,130 malevolent programs from the Virus Share malware library, as well as 1,157 harmless program files from places like the Windows System32 region and program sites like File Hippo. Cuckoo was used in this paper for running and analysis of malware in the system and the virus total was used for labeling the files as benign and malicious. Volatility is used as an analysis after extracting the memory image from the sandbox.

In 2017[5], the author uses three features to investigate the presence of unknown malware from the collected samples which are registry activity, imported libraries, and API function calls. In this research, the author used four models and those are data acquisition, feature extraction, feature selection, and model training. In 2017[6], the author used two experiments for malware detection using PE and NON-PE files. The author extracted the log file data from the memory and conducted two experiments. Process, file, registry, and network log details have been extracted. In 2017[7] author experimented on how to detect malicious PE files on the network using a machine learning algorithm. In these 28 features extracting from packing, imported DLL, metadata, and functions.

To improve the previous method and address behavioral and encryption concerns, the author combined static analysis and memory forensics methodologies in 2017 [8]. The author divided the proposed framework into 2 parts. The author uses 200 malware samples for the testing. Process data, records, Portable document excavation, routing information, firmware information processing, firmware objects, functions directories, Injected instructions, database, API hooks, strings, Timelines, log in details, system files, malware configuration, user data, and cryptography are all information obtained by this framework about malicious content. The limitations of this paper are some malware is not detected because they generated encrypted payloads, and polymorphic behavior is another issue not detected by this method.

In 2018 [9] the author used to filter-based feature selection technique in classifying malicious portable executable files from benign files. The collected dataset was a balanced dataset in which 200 benign and 200 malicious files and an unbalanced dataset in which 200 benign and 400 malware files were collected.

In 2019 [10] author uses memory forensics and a dynamic analysis approach for malware detection. Cuckoo sandbox, virtual environment, and volatility tools used for extraction and API feature analysis. In 2019[11] author used a static approach for ransomware detection. PE view and PE parser are used for analyzing PE files and getting DLL used by the ransomware. As a result, the author extracted the PE file features packer status, compile date and time, the cryptographic function used, DLL, and functions used by the Locky ransomware.

In 2019 [12] author used the machine learning technique for differentiating malicious and benign files. The only PE files were collected from the Windows XP and 7 systems. Only the header value of PE files was used in this research and this is not sufficient in real-world scenarios. In 2019[13] Portable executable malware was detected using a supervised binary classifier. The dataset consists of 75% malicious files and 25% benign files. The suggested model's limitations are that it simply detects whether a particular file is malware or not; it does not provide any information about the virus's family of origin, and if the file is encrypted then it's not going to be detected by this model.

In 2019, Metamorphic malware was detected by using the Markov logic networks algorithm[14]. 300 malware samples were collected and that malware is a virus, Trojan, worm, and PUP type. FP growth approach to creating API patterns and Markov logic networks algorithm for verification. In 2019[15] the author developed a portable file malware detection system using static analysis using data mining techniques. The author extracted 60 features and that 34 features are significantly important for malware detection. Based on input samples, the system detects and divides the malicious and benign files.

In 2020 [16] executed an experiment on windows 7 32 and 64-bit memory images. In this paper, the author used the volatility sig check plugin for signature verification of PE files. The virus Total has been used for labeling the malware which is going to be tested. EXE files, DLL files, and SYS files are evaluated from both windows 7 X64 and X86 systems. Limitations of this paper are data changes, incomplete data affected by PE relocation, executable files, catalog signed files, and inconsistencies in the processes. In 2020[2] the author detected the malware using PE header specification with the help of machine learning techniques. The research evaluated using 2460 files with an equal number of malicious and harmless files extracted from the Windows XP machine.

In 2021[17] implemented malware detection using memory forensics technique. The author uses the cuckoo sandbox to execute the malware and volatility for analyzing the malware. In this paper, 2502 malicious files and 966 good files were used in the experiment. API call feature, DLL feature, process handle feature, privilege feature, network feature, and code injection feature. In 2021 [18] author used the static analysis automation technique for malware detection using machine learning. The characteristics are taken from the PE files in this study and an image has been generated for giving input to the system. The author performed the module using PE structure, the module using an image, classification system performance experiment for malicious code classification.

IV. METHODOLOGY

A. Data collection**:** Data collection is the foremost step in malware analysis. In this research data has been collected from various open-source malware repository websites such as tendefence, virusShare, and GitHub.

B. Sample testing**:** This is the second step in malware analysis by ensuring the collected sample is malware or not and knowing what malicious family it belongs to. The collected malware samples are tested on the VirusTotal repository website.

C. Feature extraction**:** Feature extraction is the process of obtaining the characteristics and behavior of the malware samples using static and dynamic analysis with the help of forensics tools. Exinfo PE and PsStudio are used in static malware analysis and Hex editor and Windows PowerShell are used for dynamic analysis.

D. Feature analysis**:** The feature analysis is the last step in the malware analysis framework in which the malware files are decoded into readable text format and malicious characters are examined, results obtained from feature analysis are saved in text format.

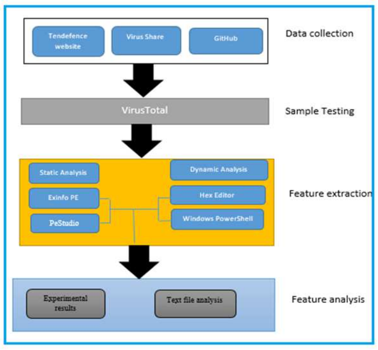
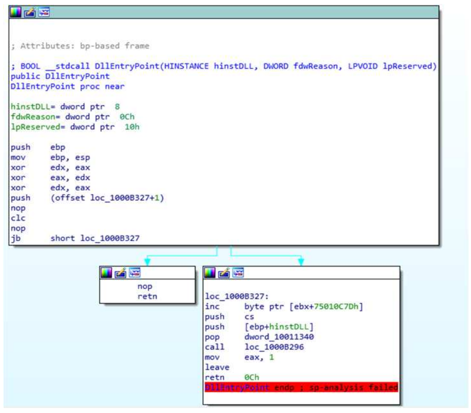


Fig.3. Overview of methodology.

V. EXPERIMENTAL RESULTS

The malware code has been written in any language and executed on any operating system. The decrypted code of the malware sample has been shown in Fig.4. To make the code malicious and difficult to understand, the code has been XOR manipulated with hexadecimal values. The decryption of the malware file has been performed using the IDA freeware forensic tool which is an open-source tool available freely.



```
; Attributes: bp-based frame

; BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
public DllEntryPoint
DllEntryPoint proc near

hinstDLL= dword ptr  8
fdwReason= dword ptr  0Ch
lpReserved= dword ptr  10h

push    ebp
mov     ebp, esp
xor     edx, eax
xor     eax, edx
xor     edx, eax
push    (offset loc_1000B327+1)
nop
clc
nop
jb      short loc_1000B327
```

```
nop
retn
```

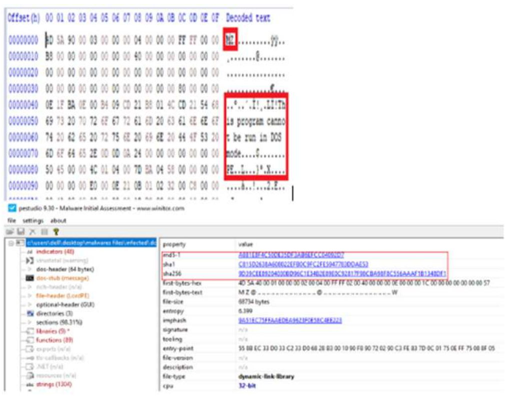
```
loc_1000B327:
inc     byte ptr [ebx+75010C7Dh]
push    cs
push    [ebp+hinstDLL]
pop     dword_10011340
call    loc_1000B296
mov     eax, 1
leave
retn    0Ch
DllEntryPoint endp ; sp-analysis failed
```

Fig.4. Decoded view of PE malicious file

The portable executable file format is widely used in spreading malicious content in windows OS. The PE file format is weak structured so that it is easy to manipulate it and insert malicious content into it. Encryption, polymorphic and metamorphic are the new malicious features of PE files.

In this research, for experimentation, a file has been taken from a malware repository. The sample was initially verified with VirusTotal online malware repository website and this site resulted that 60 security vendors and 2 sandboxes flagging this file as malicious.

The same infected malware sample has been opened in hex editor forensic tool as shown in Fig.5, and by observing some of the data like the PE file header initially starts with “MZ”, the PE signature starts with “PE”, and also the DOS STUB string “This program cannot be run in DOS mode” are the basic features to conclude that it is a Portable executable file. The hash value of the malware was also calculated using the pestudio forensic tool (md5-1, sha1, sha256).



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
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..
```

pestudio 9.30 - Malware Initial Assessment - www.winitor.com

```
c:\users\dell\desktop\malwares files\infected\d...
  indicators (48)
  virustotal (warning)
  dos-header (64 bytes)
  dos-stub (message)
  rich-header (n/a)
  file-header (Lcxd PE)
  optional-header (GUI)
  directories (3)
  sections (98.31%)
  libraries (9) *
  functions (89)
  exports (n/a)
  tls-callbacks (n/a)
  .NET (n/a)
  resources (n/a)
  strings (1304)
```

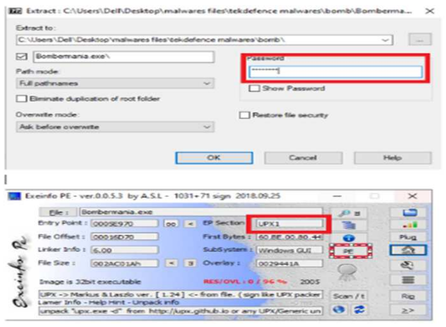
property	value
md5-1	A881E8F4C50DE25DF3AB8EFCC6460207
sha1	C815D2638A600022EFB0C9FC2FE5947783DDAE53
sha256	9D39CEE8920400BD96C1E34B2E89E0C92817F90CBA98F8C556AAAF1B1348DF1
first-bytes-hex	4D 5A 40 00 01 00 00 00 02 00 04 00 FF FF 02 00 40 00 00 00 0E 00 00 00 1C 00 00 00 00 00 00 57
first-bytes-text	M Z @ . . . . @ . . . . W
file-size	68734 bytes
entropy	6.399
imphash	9A51EC75FFAAED6A962BF0E58C4EE223
signature	n/a
tooling	n/a
entry-point	55 8B EC 33 D0 33 C2 33 D0 68 28 83 00 10 90 F8 90 72 02 90 C3 FE 83 7D 0C 01 75 0E FF 75 08 8F 05
file-version	n/a
description	n/a
file-type	dynamic-link-library
cpu	32-bit

Fig.5. Identification of PE file and its hash values using Hex Editor and PEstudio.

A. *Packed or encrypted malicious PE file*

Encryption is another dangerous feature of PE files in which the file is encrypted with the code or password so that any traditional anti-virus software is not able to scan the file and detect it as a malicious PE file. The attacker will utilize the packer's approach to obscure the malware's content, making it impossible to study the file's strings. Packers use the UPX utility to compress an executable, and when it is executed, the packed executable is decompressed, allowing it to infect the machine.

Fig.6. shows how the files get encrypted with encryption keys. The encrypted PE malware was downloaded from tendefence malware repository website which is extracted using the decryption code “infected” and tested against forensic software called Exeinfo PE. By considering the packed malware file, the packed malware file size increases after unpacking it because the packer tools compress the infected data which is present in it. This forensic tool tells that the given file is encrypted using UPX packer’s technique, the author name who have been created this malware, it is a 32 bit executable PE file and UPX tools have been used in packing the file. The Exeinfo PE malware analysis tool also suggests that the “upx.exe –d” tool can help in decompressing the executable file.



Extract : C:\Users\Dell\Desktop\malwares files\tekdefence malwares\bomb\Bomberma...

Extract to: C:\Users\Dell\Desktop\malwares files\tekdefence malwares\bomb\

☑ Bombermania.exe\

Path mode: Full pathnames

☐ Eliminate duplication of root folder

Overwrite mode: Ask before overwrite

Password: ********

☐ Show Password

☐ Restore file security

Exeinfo PE - ver.0.0.5.3 by A.S.L - 1031+71 sign 2018.09.25

File : Bombermania.exe

Entry Point : 0005E970 | EP Section : UPX1

File Offset : 00016D70 | First Bytes : 60.BE.00.80.44

Linker Info : 6.00 | SubSystem : Windows GUI

File Size : 002AC01Ah | Overlay : 0029441A

Image is 32bit executable | RES/OVL : 0 / 96 % | 2005

UPX -> Markus & Laszlo ver. [1.24] <- from file. (sign like UPX packer

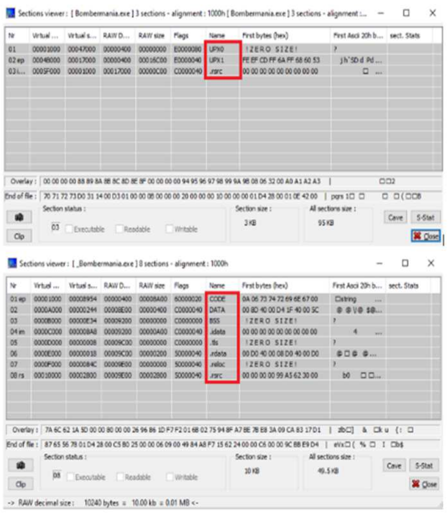
Lamer Info - Help Hint - Unpack info

unpack "upx.exe -d" from http://upx.github.io or any UPX/Generic un

Fig.6. Encrypted PE file using packers UPX technology.

B. *Unpacked or decrypted malicious PE file*

The PE file is encapsulated with many sections which include their data and functions. The PE structure has .text, .data, .rdata or .idata, .reloc, .rsrc, .debug. In these, the .text section is the first section and it contains the executable code for the program. Data sections include the strings. .idata or .rdata contains the import. .reloc contains the relocation information, .rsrc contains the list of resources used by the application, and the .debug section includes the debug information. Fig.7, comparison between compressed and decompressed images. The compressed image has been decompressed using Windows PowerShell. In Fig.7, the first image is a compressed image and the second image is decompressed image. The difference between the names of the section indicates the file compression. In compressed image, the code, data, BSS, .data, .tls, .rdata, .reloc, .rsrc and encapsulated in UPX0 and in UPX1. These two sections are malicious content which includes malicious code to infect the computer in which this file going to be executed.



Sections viewer : [Bombermania.exe] 3 sections - alignment : 1000h [Bombermania.exe] 3 sections - alignment :...

Nr	Virtual ...	Virtual s...	RAW D...	RAW size	Flags	Name	First bytes (hex)	First Ascii 20h b...	sect. Stats
01	00001000	00047000	00000400	00000000	E0000080	UPX0	!ZERO SIZE!	?	
02 ep	00048000	00017000	00000400	00016C00	E0000040	UPX1	FE EF CD FF 6A FF 68 60 53	j h`SD d Pd ...	
03 i...	0005F000	00001000	00017000	00000C00	C0000040	.rsrc	00 00 00 00 00 00 00 00 00	☐ ...	

```
Overlay :     00 00 00 00 88 89 8A 8B 8C 8D 8E 8F 00 00 00 00 94 95 96 97 98 99 9A 9B 08 06 32 00 A0 A1 A2 A3 |   ☐☐2
End of file : 70 71 72 73 D0 31 14 00 D3 01 00 00 08 00 00 00 20 00 00 00 10 00 00 00 01 D4 28 00 01 0E 42 00 | pqrs 1☐ ☐   ☐ ☐(☐☐8
```

Section status : 03 ☐ Executable ☐ Readable ☐ Writable | Section size : 3 KB | All sections size : 95 KB

Sections viewer : [_Bombermania.exe] 8 sections - alignment : 1000h

Nr	Virtual ...	Virtual s...	RAW D...	RAW size	Flags	Name	First bytes (hex)	First Ascii 20h b...	sect. Stats
01 ep	00001000	00008954	00000400	00008A00	60000020	CODE	0A 06 73 74 72 69 6E 67 00	☐string ...	
02	0000A000	00000244	00008E00	00000400	C0000040	DATA	00 8D 40 00 D4 1F 40 00 5C	@ @\@ $@...	
03	0000B000	00000E34	00009200	00000000	C0000000	BSS	!ZERO SIZE!	?	
04 im	0000C000	000008A8	00009200	00000A00	C0000040	.idata	00 00 00 00 00 00 00 00 00	4 ...	
05	0000D000	00000008	00009C00	00000000	C0000000	.tls	!ZERO SIZE!	?	
06	0000E000	00000018	00009C00	00000200	50000040	.rdata	00 D0 40 00 08 D0 40 00 D0	@☐@ @...	
07	0000F000	0000084C	00009E00	00000000	50000040	.reloc	!ZERO SIZE!	?	
08 rs	00010000	00002800	00009E00	00002800	50000040	.rsrc	00 00 00 00 99 A5 62 30 00	b0 ☐☐...	

```
Overlay :     7A 6C 62 1A 5D 00 00 80 00 00 26 96 86 1D F7 F2 01 6B 02 75 94 8F A7 8E 7B E8 3A 09 CA 83 17 D1 | zb☐]  &  ☐k u  {: ☐
End of file : 87 65 56 78 01 D4 28 00 C5 B0 25 00 00 06 09 00 49 84 A8 F7 15 62 24 00 00 C6 00 00 9C 8B E9 D4 | eVx☐( % ☐  I  ☐b$
```

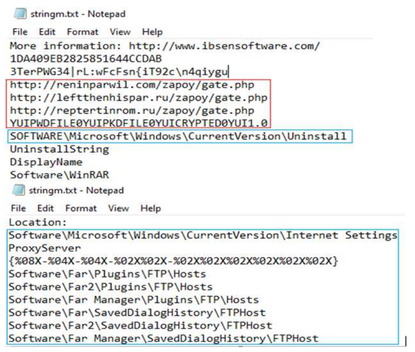
Section status : 08 ☐ Executable ☐ Readable ☐ Writable | Section size : 10 KB | All sections size : 49.5 KB

-> RAW decimal size : 10240 bytes = 10.00 kb = 0.01 MB <-

In Fig.7. Encrypted and decrypted malicious PE file analysis using Exeinfo PE forensic tool.

c. *String analysis for PE malicious code extraction*

Programming codes are represented in binary and hexadecimal language which is not readable by humans and strings are the part of the program read by humans. The technique of retrieving understandable texts from a malicious program is known as string analysis. Strings give insight into the malware's operation. It contains useful as well as garbage strings and is represented in ASCII and Unicode format. File names, URLs, IP addresses, and registry keys are the data that is extracted from the strings. In this research, a Windows power shell is used for data extraction and string analysis. The power shell provides the strings from the malware file in the form of a text file. Fig.8.explains the experimental result of ASCII strings text file extracted using windows power shell. The PE malicious file includes malicious URLs which may be used to export the data or information collected by the malware to the host malware system and also contains some registry keys which are mainly used for storing the infected malware at the root of the system to not detected by the users.

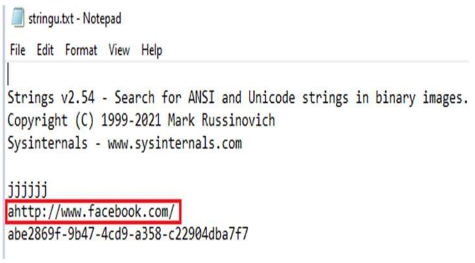


```
More information: http://www.ibsensoftware.com/
1DA409EB2825851644CCDAB
3TerPWG34|rL:wFcFsn{iT92c\n4qiygu
http://reninparwil.com/zapoy/gate.php
http://leftthenhispar.ru/zapoy/gate.php
http://reptertinrom.ru/zapoy/gate.php
YUIPWDFILE0YUIPKDFILE0YUICRYPTED0YUI1.0
SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
UninstallString
DisplayName
Software\WinRAR
```

```
Location:
Software\Microsoft\Windows\CurrentVersion\Internet Settings
ProxyServer
{%08X-%04X-%04X-%02X%02X-%02X%02X%02X%02X%02X%02X}
Software\Far\Plugins\FTP\Hosts
Software\Far2\Plugins\FTP\Hosts
Software\Far Manager\Plugins\FTP\Hosts
Software\Far\SavedDialogHistory\FTPHost
Software\Far2\SavedDialogHistory\FTPHost
Software\Far Manager\SavedDialogHistory\FTPHost
```

Fig.8. String analysis of text file using windows power shell.

The string analysis also contains the Unicode strings shown in Fig.9, which provide more clear results than ASCII strings. Fig.9 is the Unicode string result extracted from the malicious PE file. The Unicode text file Contains the malicious social media website which may be used for secured data transfer from the infected system to the attacker's host malware system.



```
Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

jjjjj
ahttp://www.facebook.com/
abe2869f-9b47-4cd9-a358-c22904dba7f7
```

Fig. 9. Unicode string representation of malicious PE file using windows power shell.

VI. COMPARISON OF FEATURE EXTRACTION OF STATIC PE FILE

Many researchers worked on PE malicious files to extract static features[19][20][21][22] PE header data, strings, import data, OP code, section data, packed data, entropy, n-grams, and DLL. By using these features, one cannot find a malicious PE file in machine learning techniques. In this research paper, extracted features are shown in table 1, by using these features, one can find malicious static PE files using machine learning techniques. Hence the proposed research paper is more efficient than the existing results.

VII. DISCUSSION

Many researchers worked on PE malicious file analysis but with only static and manual analysis. In this research, the PE malware has been analyzed using static, dynamic, and also with forensic tools and methods which provide more features and characteristics.

In this research, the features which are extracted and which are unable to extract using forensics tools are listed in table 1.

Table 1: Extracted different payload format

Features	Results	Features	Results
.exe file	✓	Decrypted data	✓
.DLL file	✓	Strings	✓
.obj file	✓	Unicode and ASCII	✓
.src	✓	Hash values	✓
Offset values	✓	Registry data	✓
Import data	✓	.xl, doc, docx,	✗
Export data	✓	.jpj, .jif, .png	✗
Source code	✓	.mp4,.mpg, mv	✗
Encrypted/packed data	✓		

VIII. CONCLUSION

The malware analysis plays a very important role in any incident response process. In this research paper, the analysis of the malware starts with the malicious code extraction using IDA freeware forensic tool from the malware sample file and analyzing the file extension. The Hex Editor forensic tool was used to detect whether it is a PE file or not and by considering magic bytes, off-set values, and string data features extracted from the malicious file using Hex Editor, it is concluded that the file is a PE file. The file was initially encrypted and using the password the file has been decrypted successfully. The Exeinfo PE forensic tool was used for internal static analysis and the result revealed that the internal content of the malicious file was hidden using the packers tool. The Windows PowerShell command-line tool was used for unpacking the encrypted file and provides the results in ASCII and Unicode text format. From the resulting text file, the malicious contents such as URLs, registry hives, malicious websites, and social media links are extracted. By considering all the above methods and extracted features, it is concluded that this research is more reliable in identifying the PE malware.

IX. FUTURE WORK

The first generation features extracted in this research are used to identify the exact family in which the PE malware belongs and its malicious effect on the computer is determined, and also second-generation malware can be detected with the help of PE malware source code analysis.

ACKNOWLEDGMENT

Mr. Manojkumar T Kamble is thankful to the Department of science and technology of Karnataka (DST) for supporting our work through Ph.D. fellowship No. DST/KSTePS/Ph.D. Fellowship/PHY-04:2021-22/1033.

REFERENCES

[1] S. Talukder and Z. Talukder, "A Survey on Malware Detection and Analysis Tools," *Int. J. Netw. Secur. Its Appl.*, vol. 12, no. 2, pp. 37–57, 2020, doi: 10.5121/ijnsa.2020.12203.

[2] T. Rezaei and A. Hamze, "An Efficient Approach for Malware Detection Using PE Header Specifications," *2020 6th Int. Conf. Web Res. ICWR 2020*, pp. 234–239, 2020, doi: 10.1109/ICWR49608.2020.9122312.

[3] Sudhakar and S. Kumar, "An emerging threat Fileless malware: a survey and research challenges," *Cybersecurity*, vol. 3, no. 1, pp. 1–12, 2020, doi: 10.1186/s42400-019-0043-x.

[4] M. Belaoued and S. Mazouzi, "A Real-Time PE-Malware Detection System Based on CHI-Square Test and PE-File Features," pp. 416–425, 2015, doi: 10.1007/978-3-319-19578-0.

[5] R. Mosli, R. Li, B. Yuan, and Y. Pan, "Chapter 11 A BEHAVIOR-BASED APPROACH FOR MALWARE DETECTION," pp. 187–201, 2017, doi: 10.1007/978-3-319-67208-3.

[6] C. Tien, J. Liao, and S. Chang, "Memory Forensics Using Virtual Machine Introspection for Malware Analysis," pp. 518–519, 2017.

[7] R. Vyas, X. Luo, N. Mcfarland, and C. Justice, "Investigation of Malicious Portable Executable File Detection on the Network using Supervised Learning Techniques," pp. 941–946, 2017.

[8] C. Rathnayaka, "An Efficient Approach for Advanced Malware Analysis using Memory Forensic Technique," pp. 1145–1150, 2017, doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.365.

[9] S. L. S. Darshan and C. D. Jaidhar, "ScienceDirect Performance Performance Evaluation Evaluation of of Filter-based Filter-based Feature Feature Selection Selection Techniques Techniques in in Classifying Classifying Portable Portable Executable Executable Files Files," *Procedia Comput. Sci.*, vol. 125, pp. 346–356, 2018, doi: 10.1016/j.procs.2017.12.046.

[10] R. Sihwail, K. Omar, K. A. Z. Ariffin, and S. Al Afghani, "Malware detection approach based on artifacts in memory image and dynamic analysis," *Appl. Sci.*, vol. 9, no. 18, 2019, doi: 10.3390/app9183680.

[11] S. Poudyal, K. D. Gupta, and S. Sen, "PEFile Analysis : A Static Approach To Ransomware Analysis," no. October, 2019, doi: 10.5769/J201901004.

[12] A. Kumar, K. S. Kuppusamy, and G. Aghila, "A learning model to detect maliciousness of portable executable using integrated feature set," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 31, no. 2, pp. 252–265, 2019, doi: 10.1016/j.jksuci.2017.01.003.

[13] H. Shukla, S. Patil, D. Solanki, L. Singh, M. Swarnkar, and H. K. Thakkar, "On the Design of Supervised Binary Classifiers for Malware Detection Using Portable Executable Files," *Proc. 2019 IEEE 9th Int. Conf. Adv. Comput. IACC 2019*, pp. 141–146, 2019, doi: 10.1109/IACC48062.2019.8971519.

[14] C. Choi, C. Esposito, M. Lee, and J. Choi, "Metamorphic Malicious Code Behavior Detection Using Probabilistic Inference Methods," *Cogn. Syst. Res.*, 2019, doi: 10.1016/j.cogsys.2019.03.007.

[15] M. S. Yousaf, M. H. Durad, and M. Ismail, "Implementation of Portable Executable File Analysis Framework (PEFAF)," *2019 16th Int. Bhurban Conf. Appl. Sci. Technol.*, pp. 671–675, 2019.

[16] D. Uroz and R. J. Rodríguez, "Forensic Science International : Digital Investigation On Challenges in Verifying Trusted Executable Files in Memory Forensics," *Forensic Sci. Int. Digit. Investig.*, vol. 32, p. 300917, 2020, doi: 10.1016/j.fsidi.2020.300917.

[17] R. Sihwail, K. Omar, K. Akram, and Z. Ariffin, "An Effective Memory Analysis for Malware Detection and Classification An Effective Memory Analysis for Malware," no. February, 2021, doi: 10.32604/cmc.2021.014510.

[18] E. Kucharska, "SS symmetry," *Mdpi*, pp. 1–19, 2020.

[19] Z. Chen, X. Zhang, and S. Kim, "A learning-based static malware detection system with integrated feature," *Intell. Autom. Soft Comput.*, vol. 27, no. 3, pp. 891–908, 2021, doi: 10.32604/IASC.2021.016933.

[20] J. Kang and Y. Won, "A study on variant malware detection techniques using static and dynamic features," *J. Inf. Process. Syst.*, vol. 16, no. 4, pp. 882–895, 2020, doi: 10.3745/JIPS.03.0145.

[21] A. Saini, E. Gandotra, D. Bansal, and S. Sofat, "Classification of PE files using static analysis," *ACM Int. Conf. Proceeding Ser.*, vol. 2014-Septe, pp. 429–433, 2014, doi: 10.1145/2659651.2659679.

[22] H. V Nath and B. M. Mehtre, "Static Malware Analysis," pp. 440–450, 2014.