



Feature Extraction of Malware Infected Files and Malicious Datasets

Hemant J. Chaudhari, Prof. M. S. Mahindrakar

M.Tech Student, Dept. of CSE., Shri Guru Gobind Singhji Institute of Engineering and Technology Vishnupuri,
Nanded, Maharashtra, India

Assistant Professor, Dept. of CSE., Shri Guru Gobind Singhji Institute of Engineering and Technology Vishnupuri,
Nanded, Maharashtra, India

ABSTRACT: In this paper, we introduce a simple approach to extract malware features of malware infected files and malicious datasets by using signature based approach and IDA pro tool. We extract various malware features and attributes from the PE-headers using the structural information of executable files. We use the following three methodologies: (1) First of all we collect large amount of malicious executable files or datasets and benign executable file or datasets for testing purpose. (2) Then we scan both sample dataset and executable file through the virus total online tool to predict file behaviour. Virus-Total is a free to analyse suspicious files and facilitates the quick detection of viruses, worms, Trojans, and all kinds of malware.(3) then we used Kfngam tool to convert suspected file into the N-grams for measurement of used symbols. (4)Then we collect large amount of information from the PE header by using IDA pro tool to extract set of features and various properties to understand malware files.

We have evaluated our approach on a malware infected executable file which contains positive detection ratio. The result of our experiments shows that the PE-header-Based approach achieves more than 79% detection rate with Threat Score: 100/100(Labelled as: Trojan.VIZ.Gen)

KEYWORDS: Malware, Feature Extraction of Malware, Malware features, PE-Headers, Metadata, Malicious Files, Sections, Entropy.

I. INTRODUCTION

Malware refers to software programs that build to damage or do unwanted and suspicious actions on a computer system. Malicious code is used to describe any code in part of a software system that is intended to cause undesired effects or damage to a computer system. Malicious code describes a broad category of system security terms that includes viruses, worms, Trojan horses, backdoors, and malicious active content. Analysing of malware files and datasets continues to be a challenge as attacker's device new techniques to evade from the detection methods. Most of the anti-virus software uses signature based detection techniques which is ineffective in the current scenario due to the rapid increase in the number malware samples. The signature is a unique identification for a binary file, which is created by binary file using static analysis methods. Dynamic analysis uses the behaviour and actions in execution to identify whether the executable is a malware or not. Both methods have own advantages and disadvantages. Modern malware detector uses a various types of detection techniques. Most of the detectors must determine the behaviour of the file and then extract the contents of file to find the embedded item. In this paper, we introduce a simple approach to extract malware features of malware infected files and malicious datasets by using signature based approach and IDA pro tool. This paper is structured as follows: The II section presents related work. Section III gives an overview of PE file format. Section III presents details on the design and implementation of PE-Header-Based detection. Section IV presents the experimental results. Section V discusses the limitation and Section VI concludes the paper.

Our goal in this research is to introduce a malware features to understand the properties and attributes of malware files to understood malware features and Experimental evaluations on a standard malware data collection are performed to evaluate the proposed technique.

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

II. RELATED WORK

Several possible techniques have been implemented in the past for malware detection. Chatchai Liang-boonprakong Ohm Sornil, Bangkok, Thailand Classification of Malware Families Based on N-grams Sequential Pattern Features [8], in this paper, they have proposed n-grams sequential pattern features for classifying malware into 10 families. N-grams are created from the binary content of files; n-gram sequential patterns are formed; and patterns are reduced to a minimal set by sequential floating forward selection procedure. Mansour Ahmadi Dmitry Ulyanov, University of Cagliari, Italy Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification [9]. They have presented a malware classification system characterized by a limited complexity both in feature design and in the classification mechanism employed. To attain this goal, a number of novel features to represent in a compact way some discriminate characteristics between different families. They both have shown general techniques of feature extraction of malware and classify them through the malware families. In [11], Smita Ranveer and Swapnaja Hiray, Comparative Analysis of Feature Extraction Methods of Malware Detection, This paper gives an overview of malware detection techniques based on static, dynamic and hybrid analysis of executable. They have been presented a comparative assessment of features and illuminated their effect on performance of the system. They have found that, high accuracy and TPR can be achieved by selecting an appropriate feature extraction method. Although op-code and PE features enhanced the speed and accuracy of malware detection system, they give rise to false positives.

III. OVERVIEW OF PE FILE FORMAT

The PE file format is a data structure that contains the information necessary for the Windows OS loader to manage the wrapped executable code. Before PE file there was a format called COFF used in Windows NT systems. The portable executable (PE) format is a file format for executable, object code, DLLs, FON font files, and core dumps. A PE executable file basically contains two sections, which can be subdivided into several sections. One is Header and the other is Section. The overview of PE format is showing as the following figure: 1.

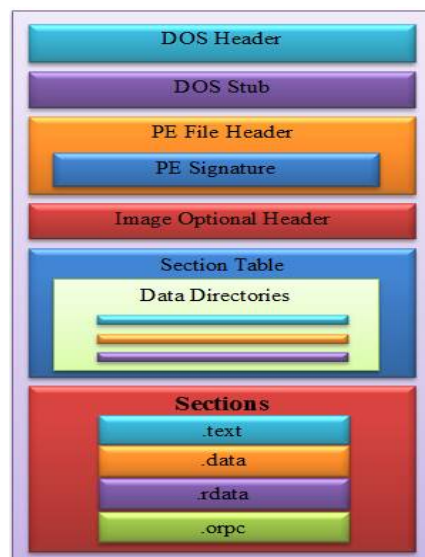


Fig.1. Basic structure of PE file

A. DOS Header: DOS header starts with the first 64 bytes of every PE file. It's there because DOS can recognize it as a valid executable and can run it in the DOS stub mode.



International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirccce.com

Vol. 5, Issue 6, June 2017

- B. DOS Stub:** The DOS stub usually just prints a string, something like the message, “This program cannot be run in DOS mode.” It can be a full-blown DOS program. When building applications on Windows, the linker sends instruction to a binary called winstub.exe to the executable file. This file is kept in the address 0x3c, which is offset to the next PE header section.
- C. PE File Header:** Like other executable files, a PE file has a collection of fields that defines what the rest of file looks like. The header contains info such as the location and size of code, as we discussed earlier. The first few hundred bytes of the typical PE file are taken up by the MS-DOS stub.
- D. PE Signature:** It only contains the signature so that it can be easily understandable by windows loader.
- E. Image Optional Header:** This optional header contains most of the meaningful information about the image, such as initial stack size, program entry point location, preferred base address, operating system version, section alignment information, and so forth.
- F. Section Table:** This defines the size of the section table, which immediately follow the header.
- G. Data Directories:** This is another sub-section in the header section. It is nothing but the array of 16 IMAGE_DATA_DIRECTORY structures, each relating to an important data structure in the PE file, namely the Import Address Table. In the current PE file, out of 16 only 11 are used.
- H. Sections:** This section contains the main content of the file, including code, data, resources and other executable files. Each section has a header and body. The **.bss** section represents the uninitialized data for the application. The **.rdata** represents the read-only data on the file system, such as strings and constants. The **.rsrc** is a resource section, which contains resource information of a module. In many cases it shows icons and images that are part of the file’s resources. The **.edata** section contains the export directory for an application or DLL. When present, this section contains information about the names and addresses of exported functions. We will discuss these in greater depth later. The **.idata** section contains information about import directory and import address table.

IV. DATASETS

We collect malicious executable files and malicious datasets from various malware hosting websites like www.vxheaven.org, malware.lu, csmining.org/index.php/malicious-software-datasets-.html, and virusshare.com. We used CSDMC2010 API sequence corpus, which is one of the datasets for the data mining competition associated with ICONIP 2010. This dataset is composed of a selection of Windows API/System-Call trace files, intended for testing on classifiers treating with sequences.

- **The corpus description:**

The dataset contains two parts:

1. **TRAINING:** 388 logs which there are 320 malware traces labeled as '1' and 68 benign software traces labeled as '0'.
2. **TESTING:** 378 traces with unknown labels -- labeled as all 0's in the file.

V. PROPOSED APPROACH

Proposed approach is based on disassembled PE-Header and extracted signature. We first took classified sets of malicious and benign executable files and malicious datasets and extract various information by using HxD tool [3] and IDA pro tool [1]. HxD tool is hex editor used to extract embedded feature from the executable file. The Interactive disassembler (IDA) is a disassembler for computer software which generates assembly language source code from machine-executable code. It supports a variety of executable formats for different processors and operating systems. It also can be used as a debugger for Windows PE, Mac OS X Mach-O, and Linux ELF executable. In our approach first of all we scanning sample dataset or executable file through the virus total online tool [4] then disassemble given file

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirccce.com

Vol. 5, Issue 6, June 2017

by using IDA pro tool [1]; then we measure number of used symbols by using kfngram tool [2], sections, metadata and finally calculate the entropy. Our goal is to introduce a noble set of features to understood malware features and Experimental evaluations on a standard malware data collection are performed to evaluate the proposed technique. Then these models are used to classify unknown given file as malicious or benign. We analysed all the datasets and executable files (benign and malicious) in training set manually for the several features.

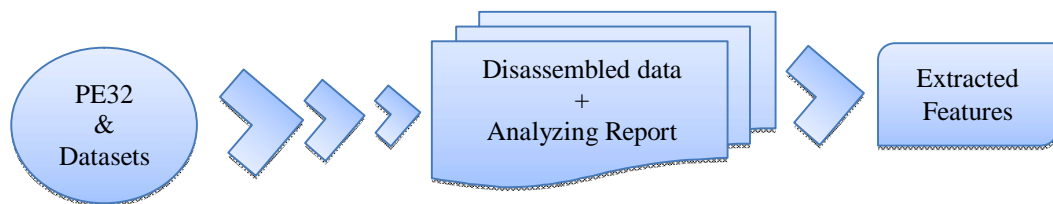


Fig.2. Technical design for feature Extraction

VI. RESULTS

In first stage we scanning sample dataset or executable file through the virus total online tool [4] then disassemble given file by using IDA pro tool [1]; then we used Kfngram tool [2] to convert suspected file into the N-grams for measurement of used symbols. And finally we measure number of detected symbols, sections, metadata and finally calculates the entropy. Our goal is to introduce a set of features to understand malicious files.

Stage I: In first stage we scan the given executable file through the virus total online tool as shown in figure 3. Table.1 shows metadata about the sample executable file. We used 39Uvmv.exe file with the 305.0 kb size.

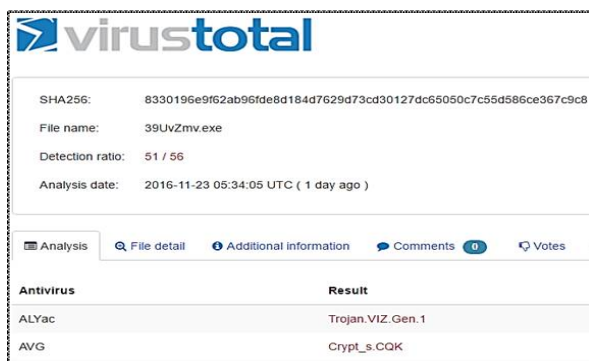


Fig: 3. Virus total result

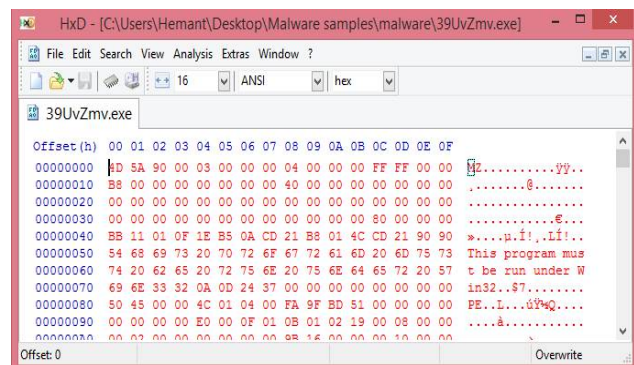


Fig: 4. File Signature in Hexadecimal view

Stage II: In second stage we extract the signature of file by using HxD tool, as shown in figure 4. as we know we can't see which data embed in executable file, but after import given file into HxD tool we can extract file signature as well as see embedded data in executable file. We used 2 different executable files to understand major difference between benign file and malware infected file. The statistical report easily differentiates benign file and malicious file as shown in figure 5 and figure 6.

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirccce.com

Vol. 5, Issue 6, June 2017

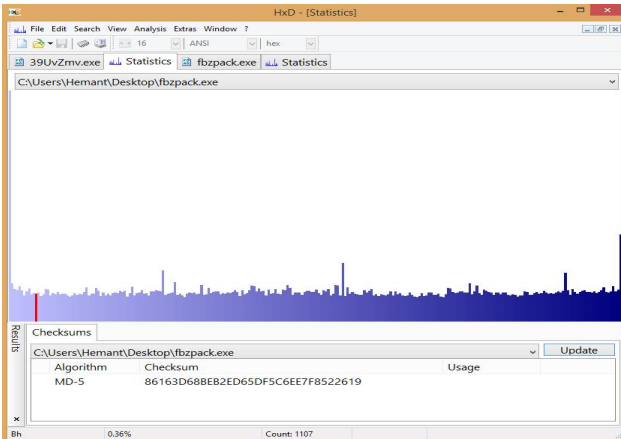


Fig: 5. Benign file Statistic report

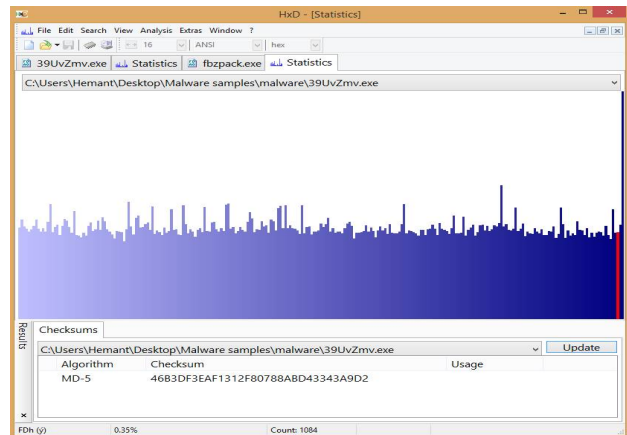


Fig: 6. Malicious file Statistic report

Stage III: In third stage we used IDA pro tool to collect information about the malicious files. Table 1 shows metadata about the malicious file. We have extracted the following features of malware from contents of file.

1. Metadata:

Metadata is data that describes other data. Metadata summarizes basic information about data, which can make finding and working with particular instances of data easier. We collect the information of our sample malicious executable file from the analysis report of virus total online tool and payload security as shown in Table 1.

File Name	39UvZmv.exe
Size of File	305.0 KB (312320 bytes)
Address of the offset	311808
Number of sections	4
File type	Win32 EXE
Code Size	2048

Table: 1. Sample file metadata

2. Set of Symbols:

The frequencies of the following set of symbols (SYM), -, +, *,], [, ?, @, are taken into account as a high frequency of these characters is typical of code that has been designed to evade detection, for example by resorting to indirect calls, or dynamic library loading. Here we are using Kfngam tool [2] to convert suspected file into the N-grams for measurement of used symbols. N-gram is a contiguous sequence of n items from a given sequence. N-gram is intensively used for characterizing sequences in different areas, e.g. computational linguistics, and DNA sequencing.



International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirccce.com

Vol. 5, Issue 6, June 2017

Used Symbols	Count
###	77
ã ü ì	32
ÿü ì ì	26
ÿü ì e	22
ÿü ÿü ÿü	20
r ###	18
à à à	14

Table: 2. Malicious symbols with count

3. Entropy:

Entropy (ENT) is a measure of the amount of disorder, and can be used to detect the possible presence of obfuscation. Entropy is computed on the byte-level representation of each malware sample and the goal is to measure the disorder of the distribution of bytes in the byte-code as a value between 0 (Order) and 8 (Randomness).

Sample File: 39UvZmv.exe

Detection ratio: Positive

File size: 312320 bytes

Name	Virtual address	Virtual size	Raw size	Entropy
.text	4096	1776	2048	5.18
.rdata	8192	2864	3072	1.78
.orpc	12288	305664	305664	7.86
.adata	319488	4096	0 0.	00

Table: 3. Section wise Entropy

4. Resources:

Resources are data items in modules which are difficult to be stored or described using the chosen programming language. This requires a separate compiler or resource builder, allowing insertion of dialog boxes, icons, menus, images, and other types of resources, including arbitrary binary data. Although a number of Resource API calls can be used to retrieve resources from a PE file we are going to look at the resources without the use of those APIs.

Once the resource section is found we can start looking at the structures and data contained in that section. As shown in following figure 7.

In figure 7 we can show number of resources are used by using repeatedly called instructions.

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijirccce.com

Vol. 5, Issue 6, June 2017

```

Directory Information----- Section Information-----
0 RVA: 0 Size: 0
1 RVA: 9620 Size: 80
2 RVA: 20480 Size: 3512
3 RVA: 16384 Size: 204
4 RVA: 0 Size: 0
5 RVA: 24576 Size: 32
6 RVA: 8832 Size: 56
7 RVA: 0 Size: 0
8 RVA: 0 Size: 0
9 RVA: 0 Size: 0
10 RVA: 8208 Size: 208
11 RVA: 0 Size: 0
12 RVA: 8416 Size: 272
13 RVA: 0 Size: 0
14 RVA: 0 Size: 0
15 RVA: 0 Size: 0

Section Name: .rsrc
Misc: 3512
Virtual Address: 20480
Size Of RawData: 3584
Pointer To Raw Data: 7168
Pointer To Relocations: 0
Pointer To Linenumbers: 0
Number Of Relocations: 0
Number Of Linenumbers: 0
Characteristics: 1073741888
  
```

Fig. 7. Repeatedly called instructions

```

00405177  C9          LERU
00405178  C3          RETI
00405179  > 0FB745 0A MOVZX EAX,WORD PTR SS:[EBP+9]
0040517D  > 2345 0C   AND  EAX,DWORD PTR SS:[EBP+C]
00405180  C3          LEAVE
00405181  C3          RETI
00405182  < FF25 44604000 JMP  DWORD PTR DS:[(KERNEL32.RtlUnwind) kernel32.RtlUnwind]
00405189  00         DB 00
0040518A  00         DB 00
0040518B  00         DB 00
0040518C  00         DB 00
0040518D  00         DB 00
0040518E  00         DB 00
0040518F  00         DB 00
00405190  00         DB 00
00405191  00         DB 00
00405192  00         DB 00
00405193  00         DB 00
00405194  00         DB 00
00405195  00         DB 00
00405196  00         DB 00
00405197  00         DB 00
00405198  00         DB 00
00405199  00         DB 00
0040519A  00         DB 00
0040519B  00         DB 00
0040519C  00         DB 00
0040519D  00         DB 00
0040519E  00         DB 00
0040519F  00         DB 00
004051A0  00         DB 00
004051A1  00         DB 00
004051A2  00         DB 00
004051A3  00         DB 00
004051A4  00         DB 00
004051A5  00         DB 00
004051A6  00         DB 00
004051A7  00         DB 00
  
```

Code caves (Blank space)

Fig. 8. Code injection space

5. Code Injection Space :

A disassemble executable file has some empty space referred to as Code Caves where malware creators can place or inject any external binary code. As shown in fig 8, you will easily identify the blank area named as DB 00 or NOP in the assembly code. In that the DB 00 instruction starts from the 00405188 offset. So, malware creators place our external spyware code in these code caves.

6. Mutexs:

A mutex, also called a lock is a program object commonly used to avoid simultaneous access to a resource, such a variable. Mutexs are usually used by malware creators to avoid the infection of a system by different instances of the same malware. When the Trojan infects a system, the first step is to obtain a handle to a “named” mutex, if the process fails, then the malware exits.

Created Mutexs:
Global\{CB561546-E774-D5EA-8F92-61FCBA8C42EE} (successful)
Local\{744F300D-C23F-6AF3-8F92-61FCBA8C42EE} (successful)
Global\{9D49E756-1564-83F5-0508-B06D3016937F} (successful)
Global\{9D49E756-1564-83F5-7109-B06D4417937F} (successful)
Global\{9D49E756-1564-83F5-490A-B06D7C14937F} (successful)
Global\{9D49E756-1564-83F5-610A-B06D5414937F} (successful)

Table. 4.Created Mutexs (Resource: Virus total Analysis report)

7. Overlays:

The process of transferring a block of program code or other data into internal memory, replacing what is already stored. A block of code or other data transferred during the overlay process.

File Name	39UvZmv.exe
MD5	bf619eac0cdf3f68d496ea9344137e8b
File type	ASCII text
Offset	311808
Size	512
Entropy	0.00

Table. 5. Overlays Report

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

8. Embedded Section:

A PE consists of some predefined sections like .text, .data, .bss, .rdata, .edata, .idata, .rsrc, .tls, and .reloc. Because of evasion techniques like packing, the default sections can be modified, reordered, and new sections can be created. We extract different characteristics from sections (SEC). The .orpc section is code section inside rpcrt4.dll files. This section holds dereference remote procedure calls, and calling number of resources repeatedly and make unwanted events to disrupt users systems.

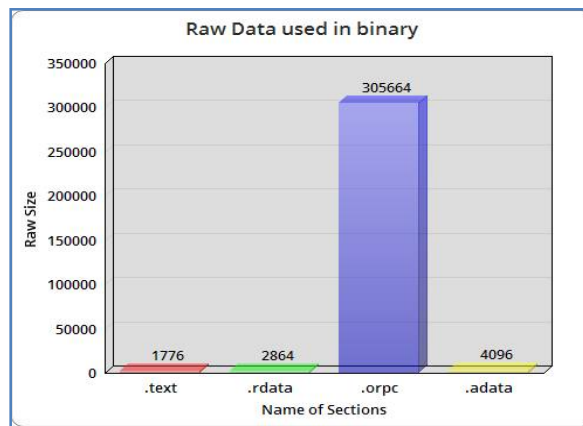


Fig: 9.Raw data report

9. Byte-Plot:

Byte Plot is a graphical representation to represent and detect meaningful sections of a given portable executable or binary file. As shown in figure 10. In following figure left block shows Byte-Plot graphical representation with the visible ASCII, invisible ASCII and non ASCII. In this representation blue pixels shows visible ASCII text data, green pixels shows invisible ASCII and yellow pixel shows non-ASCII text. According to PE-header file representation invisible ASCII code used to inject malicious code in embedded file and then wrapped into the core file. In short 39UvZmv.exe file is encoded with another two files to make malicious event.

File Name : 39UvZmv.exe

Wrapped Files: aCQKcf.exe & Vti-reSCAN.exe

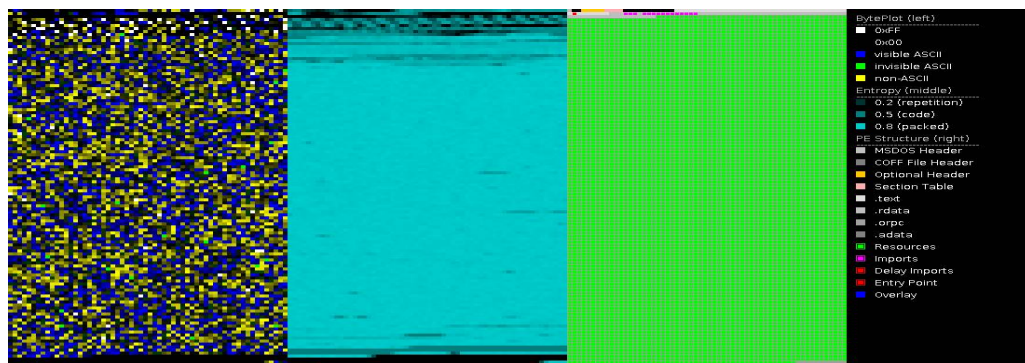


Fig: 10.Graphical Representation of payload

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

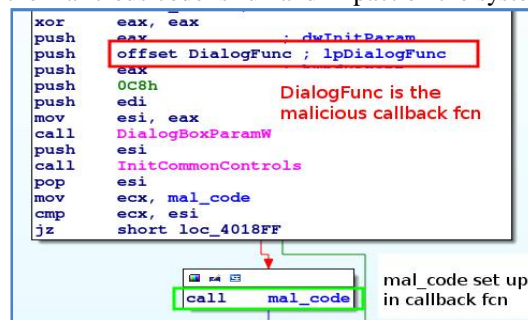
Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

10. Offsets (Call-Back Function):

Callback functions are routines that are passed to Windows API functions as a parameter, and are called later by the API to perform some type of functionality, such as processing messages or handling events. Callbacks are used in a number of places within the Windows API, and a quick MSDN search finds hundreds of Windows functions that use them.

Malware will often set up own malicious callback functions and pass them to APIs. When the API function is called, the callback function is executed and the malicious code is run and impact on the system resources.



```
xor     eax, eax
push   eax           ; dwInitParam
push   offset DialogFunc ; lpDialogFunc
push   eax
push   0C8h
push   edi           ; DialogFunc is the
mov     esi, eax     ; malicious callback fcn
call   DialogBoxParamW
push   esi
call   InitCommonControls
pop    esi
mov    ecx, mal_code
cmp    ecx, esi
jz     short loc_4018FF
call   mal_code
mal_code set up
in callback fcn
```

Fig: 11.Callback function

Callback functions normally used by the Windows API to handle program operations are used by malware to redirect the flow of execution and increase analysis difficulty. The two examples examined in this post only represent a small percentage of what and how malware uses malicious callback functions. However, by understanding the purpose of callback functions, and how to find and analyse those, analysts will be better prepared when they come across malware samples that use them.

VIII. CONCLUSION

We presented a feature extraction of Malware infected files and malicious datasets; we have extracted ten features of malware file from signature and PE-header disassembled file. To attain this goal, we used online tools namely as Virus Total, Hex to ASCII Text converter, Count Number of lines in Text and count symbols. The tools used in the project are freely available, and the overall feature extraction process is significantly depends on the analysis report of virus total and IDA pro tools and signature retrieved from HxD tool. IDA Pro is freeware commercial software/tool. The used numbers of samples are different for testing and comparison. Malware family identification is a complex process but feature extraction report helps to identify malware type. From the different experiments considering feature extraction, we observed that a number of features are involved into the creation of malware file by the hackers.

We have extract the 10 features namely as Metadata, Set of symbols, Entropy, Resources, Code injection space, Mutexes, overlays, Embedded sections, Byte-plot and offsets. Recreating the experiment is generally feasible for the most part. All used tools are free to use, and so is the software required to carry out the experiments itself. Another significant challenge for recreating the experiment is the availability of the dataset itself, which, at the time of writing, is not publicly available. All the tools are free and open-source, except from Virus Total. Virus Total offers an API and provides analysis report which can connect to perform analysis for a range of files. The public API is limited, both in terms of scans per minute and scans per-day.

REFERENCES

1. IDA-Pro tool, available at [http:// www.hex-rays.com](http://www.hex-rays.com)
2. KfNgram tool available at <http://www.kwicfinder.com/kfNgram>
3. HxD Tool, available at <https://mh-nexus.de/en/hxd/>



ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

4. Virus-Total online tool available at, <https://www.virustotal.com>
5. Hex-to-text Converter online tool, available at <http://string-functions.com/hex-string.aspx>
6. Count number of lines online tool, available at <https://www.tools4noobs.com/onlinetools>
7. VXheavens Website for Datasets <http://vx.netlux.org>
8. Chatchai Liangboonprakong Ohm Sornil, Bangkok, Thailand Classification of Malware Families Based on N-grams Sequential Pattern Features IEEE 2013.
9. Mansour Ahmadi Dmitry Ulyanov, University of Cagliari, Italy Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification CODASPY 16, March 09-11, 2016, New Orleans, LA, USA.
10. Smita Ranveer Swapnaja Hiray, Sinhgad College of Engineering, Pune Comparative Analysis of Feature Extraction, Methods of Malware Detection International Journal of Computer Applications (0975 8887) Volume 120 No. 5, June 2015.
11. ROBERT LYDA, Sparta JAMES HAMROCK, McDonald Bradley Using Entropy Analysis to Find Encrypted and Packed Malware in 1540-7993/07 2007 IEEE SECURITY PRIVACY.