[1]Manojkumar T Kamble,
[2]Dr. Sridevi

Identification of Desirable Traits in Malicious Portable Executable Files

JES Journal of Electrical Systems

Abstract: Malware is software that has been purposefully created to interfere with the system's operation. Contemporary malware is built with mutation, encryption, and other elements that make it more active and resistant which made it undetected by modern anti-virus software and result in a daily growth in the variety of malware samples. The Windows operating system makes use of the Portable executable file, a format or data structure for the executable file. Computers on windows OS employ a certain set of information in the PE file format for executing files. These PE files are used by malware to store and disseminate dangerous information. On Windows systems, the majority of files, whether good or evil, store and execute their.exe and other related documents in the PE file format. To identify the malicious content in the existing PE file, in this research paper, to identify the desirable characters from malicious PE files extracted using some forensic tools. Also implemented for malware identification and detection.

Index Terms—Encrypted files, Good files, malicious software, Malware, Portable executable (PE) file.

I. INTRODUCTION

Software known as malware attacks computer systems or certain devices to steal data or degrade system performance. Malware is identified by its harmful conduct, which contravenes the user's intentions. The objectives are to get access to the network and computer resources, interfere with system performance, and collect private data without the computer owner's awareness [1]. The spread of malicious information affected all aspects of daily life, including business and the internet. Key loggers, rootkits, backdoors, botnets, ransomware, worms, spyware, and malicious advertising files are just a few examples of the many different types of malware [1]. A PE format is used to hold several file kinds, including .exe files, Dynamic link libraries, object files, and others [2]. Windows os uses a Portable Executable file format, for the system to execute files. The PE file format is broken down into smaller components like PE header, sections, data dictionaries, DOS stubs, PE signatures, and image optional headers. The metadata stored in DOS MZ HEADER which gives details about the PE file structure [2]. The PE file damages the system and corrupts all system activities when it contains malware. The PE header provides information needed by the OS to execute the applications. The following figure 1 depicts the PE file format structure.

Figure 1. The PE file's fundamental structure.

[1]Research Scholar, Department Of Computer Science, Karnatak University, Dharwad. India, Manojkumarkamble6@gmail.com
[2]Professor, Department Of Computer Science, Karnatak University, Dharwad, India sridevi@kud.ac.in

In the above figure 1, the DOS MZ HEADER, the MZ character begins in every PE file. The function of DOS STUB gives an error message when execuble is not compatible with DOS and exits. The PE signature is used to identify the real signature of a file. Image optional header is used to find entry point and subsystem of the PE file. Section Table is an array of IMAGE SECTION HEADER structures and contains information related to the various sections such as SizeOfRawData, VirtualSize, PointerToRawData, Virtual Address, characteristics available in the image of an executable file. The sections in the image are sorted by the Relative virtual address (RVA) rather than alphabetically. Section Table is an array of IMAGE SECTION HEADER structures and contains information related to the various sections such as SizeOfRawData, VirtualSize, PointerToRawData, Virtual Address, characteristics available in the image of an executable file. The sections in the image are sorted by the Relative virtual address (RVA) rather than alphabetically. The section table also gives instruction to launch an application in the memory PE file sections header specify the section name using a simple character array field called Name. The various common sections are .text, .data,. rdata or .idata, reloc, .rsrc and .debug available from an executable file[3].

A. Techniques of Malware Analysis

Malware analysis is the process of examining malicious software to understand its behaviour, functionality, and impact. The analysis helps in identifying the type of malware, the attacker's intent, and the extent of the damage it can cause. Based on these characters, malware can be detected using the following three techniques. 1. Static analysis. 2. Dynamic analysis. 3. Memory forensics. Overall, these techniques are used by security researchers to understand the behavior and functionality of malware and protect computer systems from malicious attacks. Malware can be analysed based on its characteristics and behaviour. This information can be used to develop countermeasures to prevent and mitigate the effects of malware. This malware analysis in this research paper is implemented on 64 bit windows systems.

Static analysis

In static analysis malicious software assessed without actually executing the file. Static analysis looks for signs of malicious intent in a file. It may be useful to recognise malicious libraries, packaged files, or infrastructure. Antivirus software cannot recognize any new malware as malicious since it lacks a signature in the database. This analysis includes the signatures like string fingerprints, binary n-grams, lexical module references, sequence diagrams, and opcode relative frequencies. This analysis has to be applied to unpack and decoded before this analysis can be performed [1].

Dynamic analysis

This technique of seeing how malicious material behaves while operating in a secure setting. Since it is independent of a database filtering approach, performance testing of a virus is better than static analysis but rather on the dynamic selecting features, extraction, and assessment of whether the presented sample is malicious or benign [1]. The main important properties should be extracted before performing dynamic analysis using the necessary equipment, software, and secure surroundings [1]. Dynamic analysis is vulnerable to recent new malware because new variants of malware are known about the dynamic analysis strategies.

Memory forensics

The substantial subfield of research in forensics is memory forensics. In the field of forensic science, there are many distinct forms of forensics, including memory forensics, malware forensics, digital forensics, computer forensics, fingerprint forensics, RAM forensics, hard disc forensics, and more. The practice of extracting electronic data via system storage to solve cyber investigative concerns is known as computer forensics. Information can be kept in physical memory and on hard drives. Every action a user takes while using a computer is documented on the hard drive and in volatile memory. Malware is a type of software that, by its malicious behaviour, interferes with system operations. Malware must always be executed using random access memory when it is introduced into the system. As a result, memory forensics may be used to retrieve these virus execution traces from the memory. Memory forensics is the post process where it is useful to extract traits of malware when already malware attacks the system.

II. RELATED WORK

In 2017[4], the author merged static analysis and memory forensics approaches to enhance the prior approach and solve behavioural and encryption problems. The proposed framework was split up into two sections as section

A and section B. Section A identify the malware and section B verifies the malware identified in section A. Total 200 malware samples were tested. The following file type characteristics were tested to collect information about malicious content. Data set, documents, portable document excavation, resources to generate, system software information processing, boot loader objects, features repositories, infused directions, databases, application programming interface latches, thread, Timeframes, authentication information, operating system, spyware structure, user information, and data encryption are all sources that this framework uses to collect information about malicious content. Some malware is not identified because it developed using a encryption packages, and adaptive activity is yet one more problem that is not picked up by such research. The author utilized a static method for ransomware detection in 2019[5]. For examining PE files and obtaining the ransomware DLL, PE view and PE parser are employed. As a result, the author was able to extract the Locky Ransomware Features, Package Condition, Preconfigure Time and Date, Data encryption Feature, dynamic link libraries, and other information from the PE file. The drawback in this research is, extracts the header data only for identification of ransomware. The author [6] employed machine learning in 2019 to distinguish between harmful and good documents. The PE files from Windows XP and 7 devices that have been gathered. The header content of executable files is used to identify the malware, although it is inadequate in actual world situations in identifying malicious files.

The limitation is, this research only extracts the header data for malware identification which leads to less accuracy. In 2019[7], a supervised binary classifier was used to find portable executable malware. 25% of the files are benign, while 75% of the files are malicious. The proposed model has a few shortcomings, including the fact that it only evaluates if a particular document is malicious or not. It provides no information about the malware family, and if the file is encoded, this model won't be able to establish whether it is malware or not.

In 2021[8], The author used unprocessed header data to implement ransomware detection. The technique is excellent for the rapid identification of suspicious samples since it can accurately identify ransomware files with 93.25% efficiency without executing the file and utilizing a raw PE header. The limitation is that it provides less accuracy because the author used only a single feature while doing machine learning to detect malicious files.

III. METHODOLOGY

The methodology of malware feature extraction starts with collecting the malware file and ends by collecting desirable malicious features. In between some forensic tools are used to perform the experiments. The proposed methodological steps are shown in below figure 2.

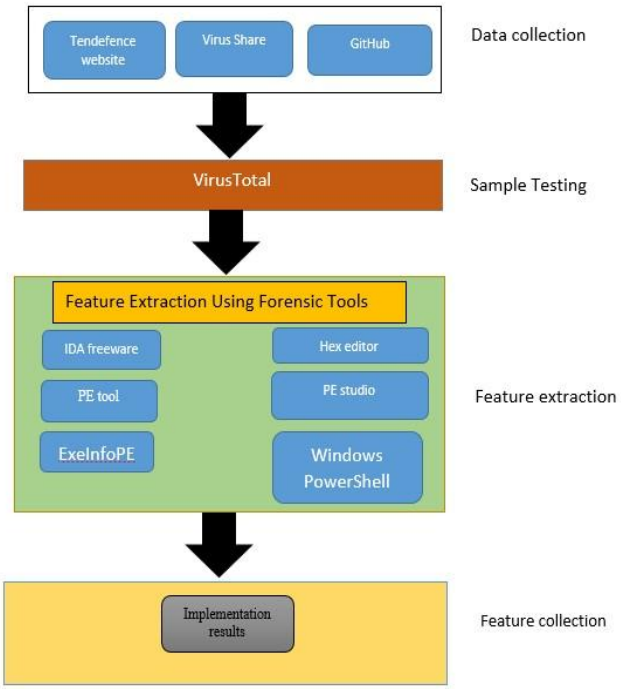

Figure 2. Outline of the proposed approach.

A. Data collection

The first stage in a malicious code investigation is data gathering. An open-source malware repository website, like tekdefence, virusShare, and GitHub, provided the information for this study [3]. While collecting the data, only portable executable files are collected not its features. In this stage, both benign and malware files have been collected from the data repository sites.

B. Sample testing

The second stage of malware feature extraction involves validating whether the sample was malware or not and identifying which dangerous family it belonged to. On the VirusTotal repository website, the malware samples that have been collected are evaluated. The results from virus total are recorded for future work.

C. Feature extraction

Using forensics tools, feature extraction is the process of learning the traits and behaviours of malware samples. In this research IDA freeware, PE tool, Exeinfo PE, Hex editor, PEStudio, and Windows PowerShell forensic tools are used to extract the features from the PE malicious file. The tools mentioned above are briefly explained below.

IDA freeware

IDA freeware, as a disassembler, has the capability to produce maps of their processing in order to show the binary instructions that are actually carried out by the processor in a symbolic representation. To make this complex code more readable, IDA Pro employs advanced techniques to translate machine-executable code into assembly language source code [9].

PE tool

PE tools enable active investigation of PE files and procedures. Also this tool included Process Explorer, PE file Editor, Dumper, Rebuilder, Comparator, and Analyser. An outdated reverse engineering instrument with a lengthy history dating back to 2002 is called PE Tools.

ExeInfoPE

Exeinfo PE allows to verify.exe files and examine all of their features, rename the file, and run the .exe manually, or remove it. The precise dimensions and place of entry are additional pieces of the information given. To assess a wide range of choices for editing any Windows executable file.

Hex Editor

HxD is a painstakingly designed and speedy hex editor that can modify RAM and raw disc data in addition to working with files of any size. The intuitive user interface offers several features, such as finding and replacing, exporting, checking for checksums and digests, adding byte patterns, shredding files, concatenating or splitting files, and much more.

PE studio

The goal of PeStudio is to find executable file traces in order to assist and hasten the first analysis of malware. The programme is used by Computer Emergency Response Teams (CERT), Security Operations Centers, and Digital-Forensic Labs globally. This tool is completely portable and works on any Windows platform; no software is necessary. No system changes or leftovers are made by PeStudio [10]

Windows PowerShell

The windows powershell is a debugger tool used to get the forensically useful information in the form of strings. The strings include Unicode's, texts, URL's, registry hives and locations of the registry files.

D. Feature collection

The malware identification framework's final stage is feature detection. Examines harmful characters and decodes malware files into readable text format. In this stage, several forensic tools have been used to extract the malicious features from the PE file. At last all the features are collected from various forensic tools.

IV. IMPLEMENTATION AND RESULTS

Any programming language may be used to write malicious code, which can run on any operating system. Figure 3 displays the malware sample's encrypted source code. The code has been altered using hexadecimal numbers to be harmful and challenging to understand. The malware file was decoded using the publicly accessible free-source IDA freeware investigative tools as shown in figure 3.

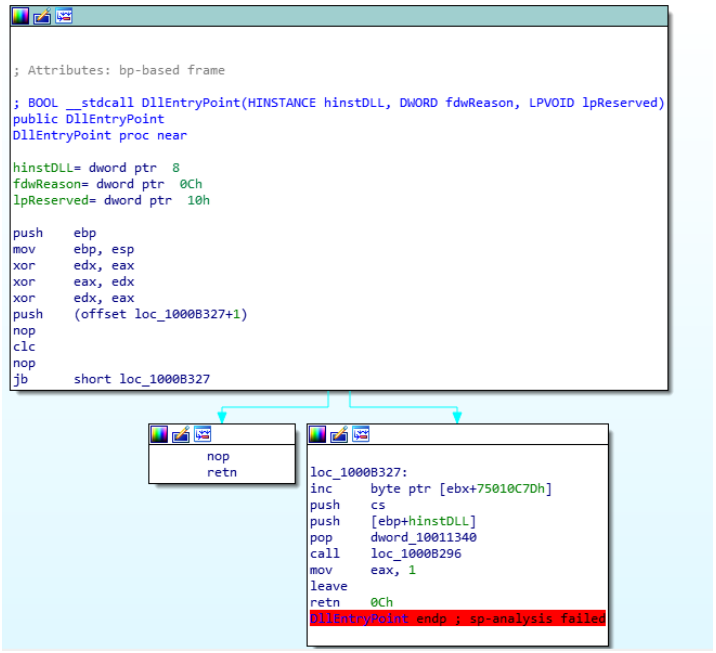

Figure 3. View of the decoded malicious PE file.

In Windows OS, PE malware is a type of malware that is frequently used to disseminate PE format. Because of the fragile nature of the PE file format, it is simple to alter and add harmful material to it. The new harmful properties of PE files include encryption, polymorphism, and metamorphism. In this study, files from a malware repository have been used for testing. The VirusTotal internet malware repository website was used to confirm the sample as benign or malicious. As seen in below Figure 4, The forensic hex editor programme was used to view the infected malware sample, and it was determined that the file was a portable executable by examining some of the data, such as the PE file header initially beginning with" MZ," the PE signature initially beginning with" PE," and also the DOS STUB string "This program cannot be run in DOS mode" [3].

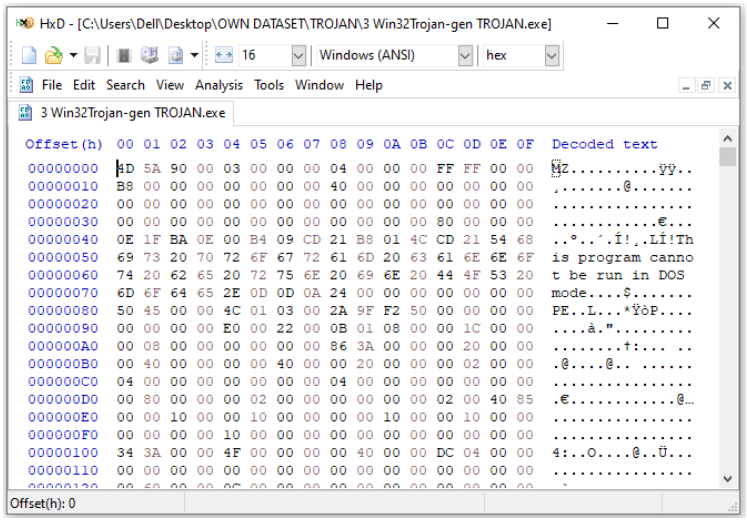

Figure 4. Identifying a PE file Using Hex Editor.

Using the Pestudio forensic program, the malware's hash value was also determined as shown in figure 5 (md5-1, sha1, sha256). Figure 5. Hash value of PE file extraction using PE studio.

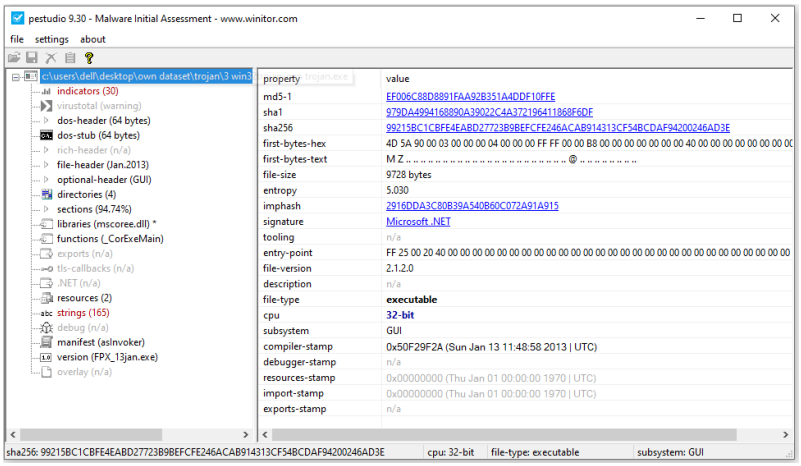

Figure 5. Hash value of PE file extraction using PE studio.

A. Encrypted malicious PE file

Encryption is a hazardous PE malicious file feature as well, which encrypts the file with a code or password, making it impossible for typical antivirus software to scan the file and identify it as dangerous. To conceal the malicious information, the attacker will use the packer's approach, leaving it hard to inspect the document's contents. 6 An executable that has been compressed using the UPX packer can infect a computer when it is run since it is decompressed. Figure 6 illustrates the use of encryption keys to encrypt the files. The encrypted PE malware was tested using the Exeinfo PE forensic tool after it was downloaded from the tendefence malware GitHub web page and decrypted with the decryption key" infected," as seen in below figure 6.

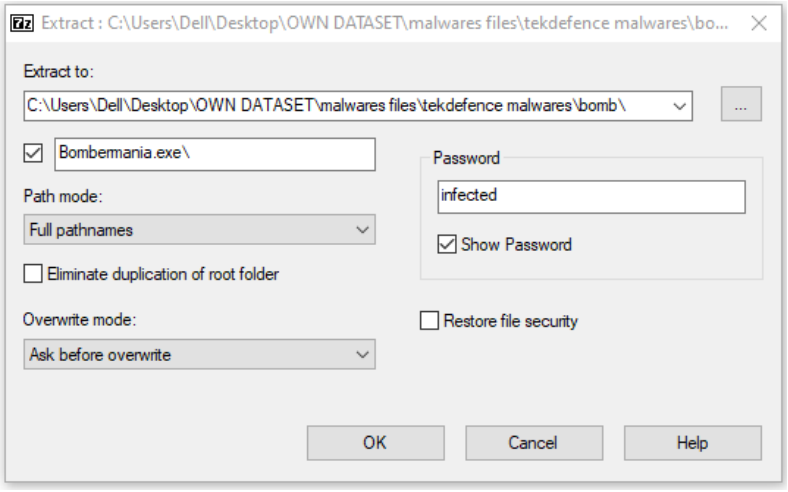

Figure 6. Decrypting PE file using a password.

The infected data that is stored in the packed malware file is compressed by the packer tools, increasing the packed malware file size. Figure 7 below illustrates how this forensic tool reveals the name of the malware's creator, the fact that the given file is encrypted using the UPX packer's technique, that it is a 32-bit executable PE file, and that UPX tools were used in the file's packing. The "upx.exe -d" utility, according to the Exeinfo PE malware analysis program, can assist in unzipping the executable file.

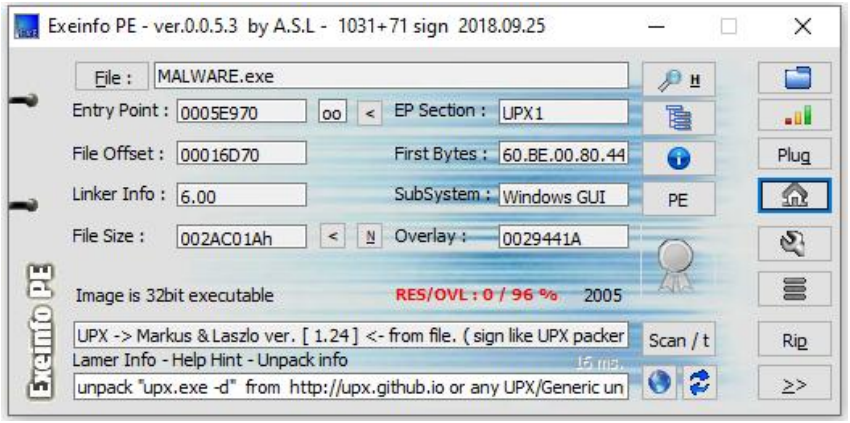

Figure 7. PE file sections that use packers UPX tool to pack data.

B. Unpacking the packed harmful PE file

The portable executable file is divided into several parts, each of which has its data and functionalities. In addition to text and data, the PE structure also contains. reloc, rsrc, and .debug sections. The first part of these is called. text, and it includes the program's executable code. The strings are part of the data portions. The import is included in .idata or .rdata files. The relocation information is in.reloc, The resources required by the programme are listed in the .rsrc file, and debug information is found in the .debug section. Figure 8 and 9 shown below, contrasts compressed versus uncompressed files.

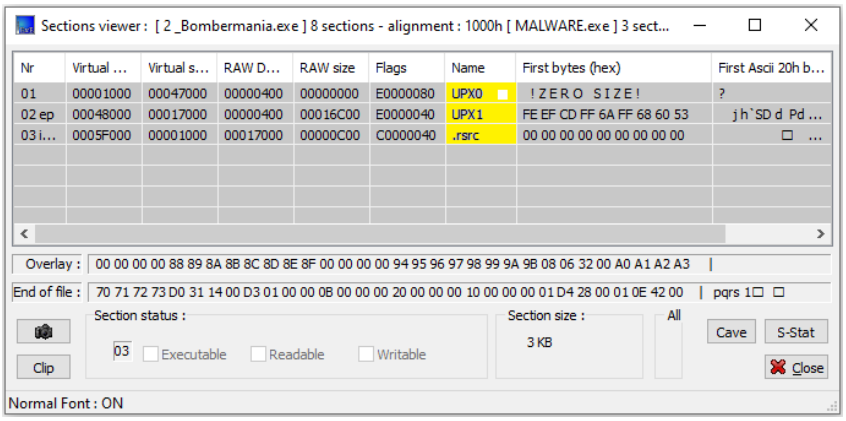

Figure 8. PE file obfuscated with UPX technology.

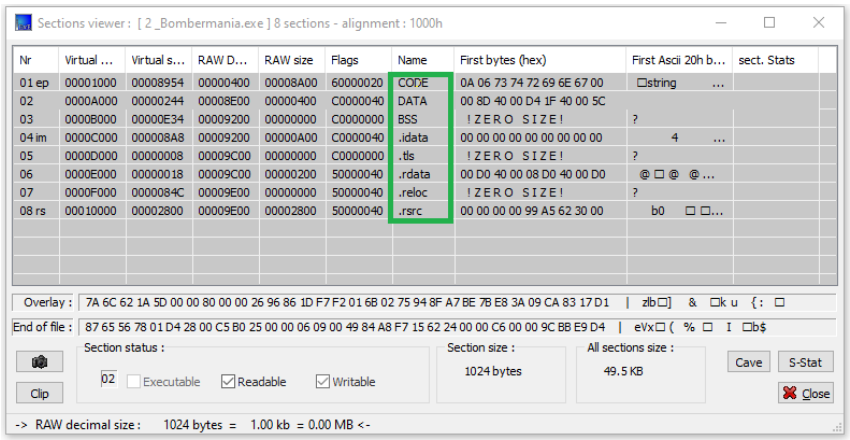

Figure 9. DE obfuscated PE malicious file.

Windows PowerShell has been used to decompress the zipped image. The first picture in Figure 8 is compressed, whereas figure 9is uncompressed. In compressed PE file, the sections are obfuscated by UPX packer

and represented as UPX0 and UPX1 as shown in figure 8. The file compression is shown by the variation in the section names. The PE file sections compressed images are contained in UPX1 and UPX0, respectively. These two parts contain malicious code that will cause the machine on which these file(UPX0 and UPX1) will be run to become infected. The Uncompressed sections with their section names code, data, .BSS,.data,.tls,.rdata,.reloc, and .rsrc is shown in the figure 9[3].

C. String analysis for extracting harmful PE code

String analysis is a method for extracting comprehensible text from harmful software. Strings provide information about how the infection functions. It comprises both useful and pointless strings and is expressed in ASCII and Unicode forms. The data extracted from the strings comprises registry keys, file names, URL, and Web addresses. In this study, data extraction and string analysis are performed using a Windows power shell. The malware file's strings are made available via the power shell as a text file. Figure below 10 explains the experimental outcome of an ASCII string extraction from a text file using Windows Power Shell [3].

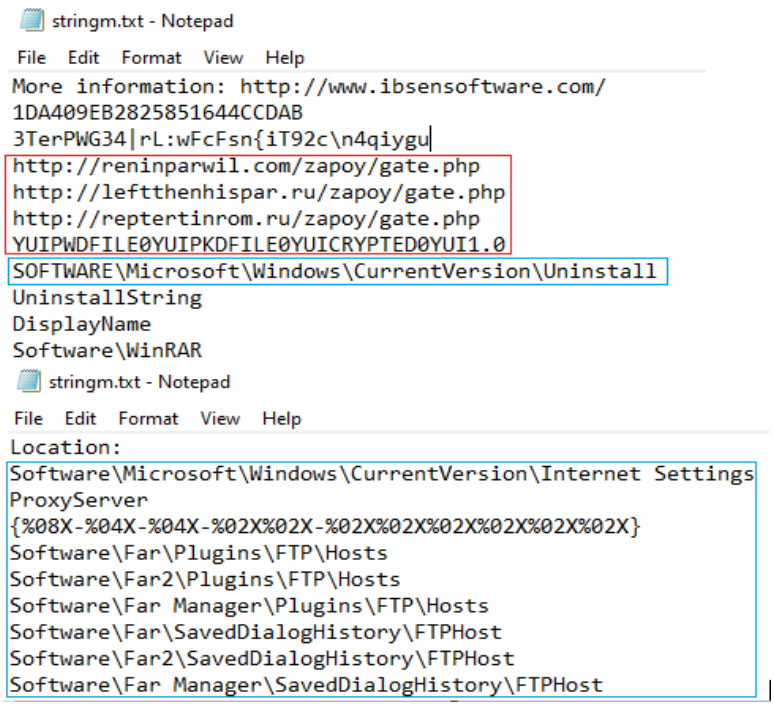

Figure 10. Windows Power Shell string analysis of a text file.

The Unicode strings shown in Figure.11 are also included in the string analysis since they provide clearer results than ASCII strings do. The below Figure 11 displays the Unicode string produced by the infected PE file. The text file in Unicode Includes the harmful social networking platform that might be used to send data securely from an infected machine to the malware creator's system.

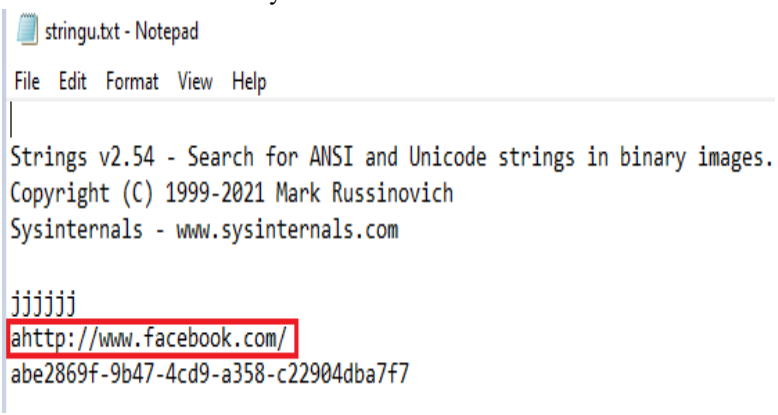

Figure 11. Windows Power Shell representation of a malicious PE file as a Unicode string.

D. PE file malicious section table

There are multiple sections, each with a specific function, where the real contents of the file are placed. These contents include data and resources that the programme utilises as well as the program's actual code. The sections are included in the header part of the section table. The various sections included in the section table are discussed below [11].

Text

The executable code for the software is found in this section [11]. The file is benign or malicious, code included in the .text section of PE file. When any PE file executed, the first executable part is .text section of PE file. This part also contains the location of the first programme instruction that will be executed, which serves as the application's entry point. There may be more than one chunk of executable code in an application [12]. Malware author's always search for the .text section of PE file to manipulate and make the benign file to malicious file.

Block starting symbol (.bss)

Uninitialized data is present in this section. The section of an object file, executable, or assembly language code in computer coding known as the block starting symbol (.bss) comprises statically allocated variables that are defined but have not yet been given a value. The " bss sector" is a common term used to describe it [13].

Read-only data (.rdata)

Read-only initialized data is contained here. The essential information regarding the portable executable file is placed in the read only data section. In this section, the data is stored in the form of text documents.

Export data (.edata)

The export tables are in this file. An application's or DLL export directory may be found in the .edata section. Essentially, it stores data about the names and locations of exported functions. Applications may be made modular with the use of dynamic linked libraries, making it easier for the functions to be reused. Programs that import the DLL can access the functions' address and offset using an export directory found in this section [14].

Import data (.idata)

The .idata file includes import tables. The. idata section includes information about imported functions, such as the Import Directory and Import Address Table. Implementing modularity is made simpler by the import section. The import section guards the programme against such modifications because the DLL can be changed at any moment [14].

Relocation (.reloc)

.reloc gives an information about picture relocation. Position-independent code is usually absent from PE files. Instead, they are built to a preferred base address, and all of the addresses that the compiler and linker produce are already fixed. The operating system will rebase a PE file if it can't be loaded at its desired location because it's already occupied by something else. This entails revising the code to incorporate the updated data and recalculating each absolute address [15].

Resource data (.rsrc)

This file contains application resources, such as pictures, icons, or even embedded binaries [11]. The necessary files and resources used by the PE file to get executed inluded in the resourse section. If the PE file has section names other than the given above standard names, then that PE file is disturbed and can be identified as a malicious portable executable file. The benign PE file's sections are shown below in figure12.

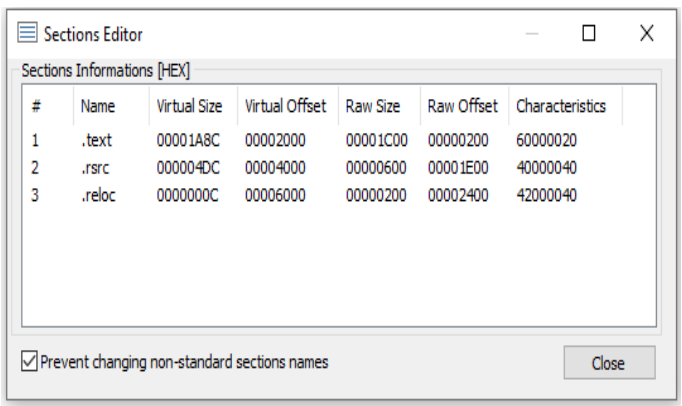

Figure 12. Benign PE file sections.

The malicious PE file's sections are obfuscated with UPX technology as shown in below figure 13. The Ultimate packer for executables(UPX) is a method of packing the file. A sophisticated executable file compressor is UPX. Program and DLL file sizes will often be reduced by UPX by upto 70 percent, saving on distribution and storage costs, network load times, download time, and storage capacity [16]. For the majority of the available formats, programmes and libraries compressed by UPX operate totally independently and without any runtime or memory overhead. The files packed with UPX mechanism are also unprocessed from the anti-virus programs.

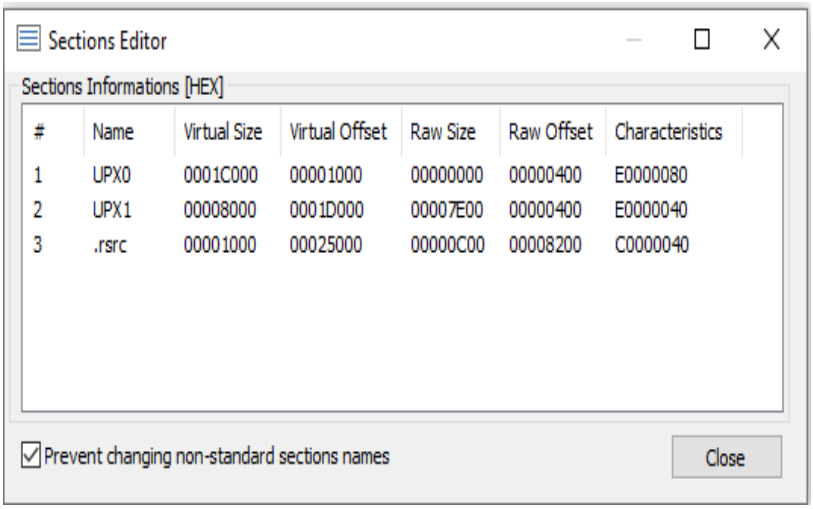

Figure 13. Obfuscated malicious PE file.

The malicious PE files with unknown sections are shown below in figure14.

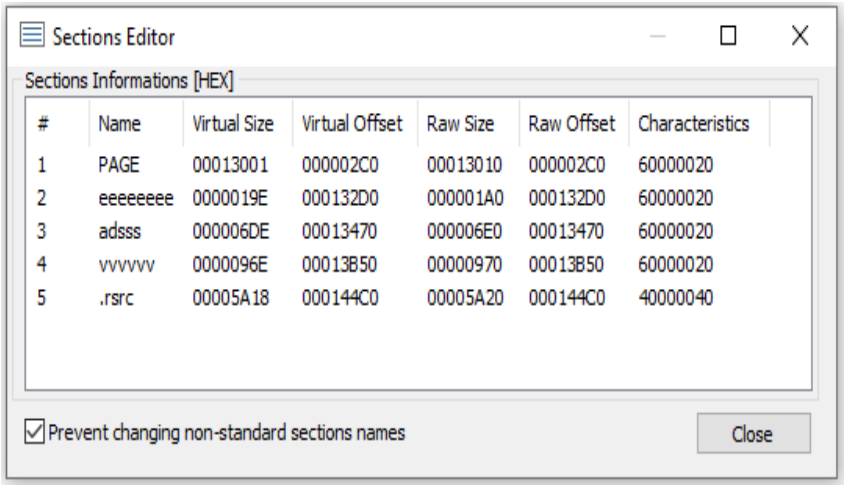

Figure 14. Malicious PE file with unknown sections.

E. PE file header malicious features extraction

In PE file, the checksum was created to find the originality of the file. The compiler generates the checksum after creating the executable, and it will become invalid if the binary is changed after compilation [17]. The checksum of the benign PE file has some value greater than 0. If the value of the checksum is 0, then the file is considered as malicious file. The malicious checksum is shown below in figure 15. The major image version of the PE file, size of initial data, and dynamic link library (DLL) value of the benign file are always more than zero. If the values of the above features are zero, then that file is considered a malicious PE FILE. The implementation results are shown in below Figure 15.

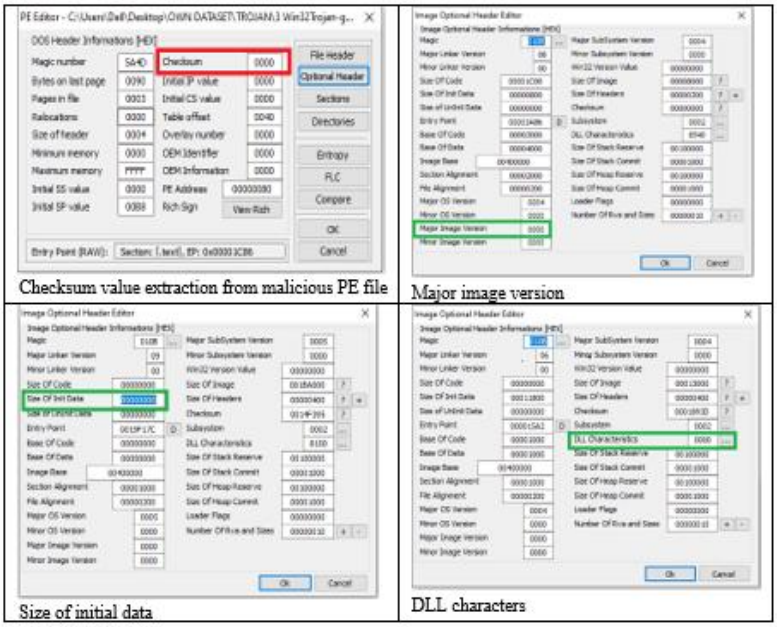

Figure 15. PE malicious header data extraction.

F. Entropy of PE File

File entropy is a measurement of the randomness of the data in a PE file. On a scale of 0 to 8, a score closer to 8 denotes that the data is more ”random”, with 0 denoting that it is less random. Usually, the benign PE file is not packed. If the file is packed, then there will be a possibility of the file contains malicious data. Entropy values for packed files will be extremely high. The high value of entropy may refer to the file being a malicious file. Even if the file is packed then its entropy value lies between 0 to 6.99. If the entropy value crosses the 7 then the file is considered as malicious. The malicious file entropy is shown in figure 16. The figure 16 shows the summary of all sections entropy with the individual PE file sections entropy.

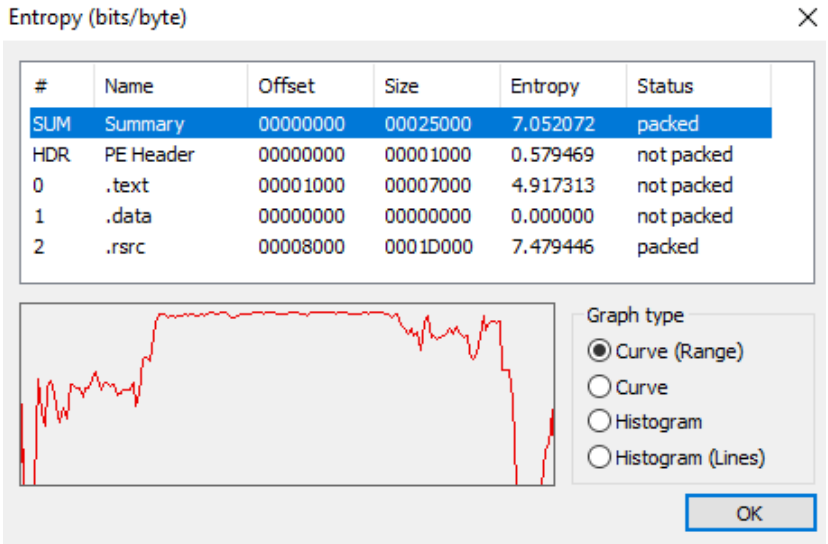

Figure 16. Entropy value of PE file using PE editor.

V. Comparison of Proposed Research with Existing Research

To extract static characteristics from PE harmful files, several researchers [2], [11], [18], [19], [20], [21] worked on them. Examples of feature types include strings, imports, OP code, section information, packed information, and entropy. These properties make it impossible to use machine learning techniques to find a malware file. Table 1 in this research article displays the retrieved properties that could be used in the future to discover malicious PE files using machine learning methods. As a consequence, the proposed study article is more effective than the current findings.

VI. Discussion

In this study, forensic tools and techniques that give additional traits and characteristics were used to identify the PE virus features. The analysis of the malware in this research paper starts with the extraction of dangerous code from the malware sample file using the free IDA forensic tool. The magic bytes, off-set values, and string data characteristics that were extracted from the malicious file using Hex Editor were utilised to identify whether the file is a PE file or not. The Exeinfo PE forensic tool indicated that the malicious file's internal content was concealed using the packer's utility. The Windows PowerShell command-line tool, which produces the results in ASCII and Unicode text format, was used to decode the encrypted file. The created text file is used to recover damaging information, including URLs, registry hives, malicious websites, and social networking links. The malware PE file sections are extracted and unknown sections are identified. More importantly, the malicious PE header features are extracted. In the end, the entropy of the PE file is identified. Table 1 lists the characteristics from this research that were retrieved and employed in forensics tools.

Table 1. Malicious PE Features that were extracted.

SL.NO	Features
1	Source code
2	Hash values
3	Encryption/packer identification
4	Offset values
5	Strings
6	Unicode and ASCII
7	PE sections
8	Registry data
9	Registry hives
10	Checksum
11	Size of initial data
12	Obfuscation data
13	Major image version
14	Entropy

VII. Conclusion

A crucial part of any incident response procedure is malware identification. It is found that this research is more accurate at recognizing PE malware features after taking into account all of the aforementioned techniques and extracted attributes. To identfify the given file is malware or benign, the proposed research all features or some of them were used. This study will help to detect the malicious PE file in malware research studies and the malware incidence response method.

VIII. Future Work

This research will be very beneficial in the future when machine learning is implemented by utilizing the suggested features. The precise family to which the PE malware belongs is identified using the proposed research, and the PE virus's harmful impact on the machine is calculated. The features are also going to be utilized in building the strong and accurate antivirus software.

References

[1] S. Talukder and Z. Talukder, "A survey on malware detection and analysis tools," *Int. J. Netw. Secur. Appl.*, vol. 12, no. 2, pp.
37–57, Mar. 2020.

[2] T. Rezaei and A. Hamze, “An efficient approach for malware detection using PE header specifications,” in *2020 6th International Conference on Web Research (ICWR)*. IEEE, Apr.
2020.
[3] M. T. Kamble and Sridevi, “Feature extraction and analysis of portable executable malicious file,” in *2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA)*, 2022, pp. 1–6.
[4] C. Rathnayaka and A. Jamdagni, “An efficient approach for advanced malware analysis using memory forensic technique,” in *2017 IEEE Trustcom/BigDataSE/ICESS*. IEEE, Aug. 2017.
[5] S. Poudyal, K. D. Gupta, and S. Sen, *PEFile Analysis : A Static Approach To Ransomware Analysis*, 2019.
[6] 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, Apr. 2019.
[7] 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,” in *2019 IEEE 9th International Conference on Advanced Computing (IACC)*. IEEE, Dec. 2019.
[8] F. Manavi and A. Hamzeh, “Static detection of ransomware using LSTM network and PE header,” in *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*.
IEEE, Mar. 2021.
[9] O. Gunes¸ and E. Turk, “Mantıku t-tayr (kus¸ masalları) projesinin o¨grenci g˘or¨ us¸leri do¨ grultusunda de˘ gerlendirilmesi,”˘ *Kalem Uluslar. Egit. Ve Insan Bilim. Derg.*, Jun. 2024.
[10] “PeStudio 9.47,” https://www.techspot.com/downloads/ 6350-pestudio.html, accessed: 2023-3-31.
[11] 0xRick, “A dive into the PE file format - PE file structure part 4: Data directories, section headers and sections,” https: //0xrick.github.io/win-internals/pe5/, Oct. 2021, accessed: 20233-28.
[12] S. Daulaguphu, “A comprehensive guide to PE structure, the layman’s way,” https://tech-zealots.com/malware-analysis/ pe-portable-executable-structure-malware-analysis-part-2/, Aug. 2022, accessed: 2023-3-29.
[13] Wikipedia contributors, “.bss,” https://en.wikipedia.org/w/index. php?title=.bss&oldid=1144135589, Mar. 2023, accessed: NANA-NA.
[14] “PE file structure: Sections,” https://keystrokes2016.wordpress. com/2016/06/03/pe-file-structure-sections/, Jun. 2016, accessed: 2023-3-29.
[15] Wikipedia contributors, “Portable executable,” https: //en.wikipedia.org/w/index.php?title=Portable Executable& oldid=1125702252, Dec. 2022, accessed: NA-NA-NA.
[16] “UPX: The ultimate packer for executables - homepage,” https:
//upx.github.io/, accessed: 2023-3-29.
[17] “Threat hunting with the PE checksum,” https:
//practicalsecurityanalytics.com/pe-checksum/, Oct. 2019, accessed: 2023-4-20.
[18] 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.
[19] S. Han, K. Lee, and S. Lee, “Packed PE file detection for malware forensics,” in *2009 2nd International Conference on Computer Science and its Applications*. IEEE, Dec. 2009.
[20] H. H. Al-Khshali, M. Ilyas, and O. N. Ucan, “Effect of PE file header features on accuracy,” in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Dec. 2020.
[21] 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, Apr. 2019.