Identifying Nobel Features in Non-Portable Executable Malware Files

Sridevi, *Member, IAENG*, Tukkappa K Gundoor, *Member, IAENG*

***Abstract*—The widespread propagation of non-portable malware files presents a hazardous challenge in the rapidly evolving world of cybersecurity. Hackers use various strategies to hide and protect their damaging payloads as the threat environment advances, rendering traditional detection and mitigation processes useless. Understanding the characteristics of non-portable malware files is critical for cybersecurity practitioners and academics, and it represents an important step toward strengthening defences against these elusive cyber-attacks. This paper examines the current state-of-the-art in non-portable malware file analysis, with a focus on pioneering approaches and technologies positioned to improve research in detecting, analyzing, and preventing modern cyber adversaryie's harmful actions, particularly for Doc, XML, HTML, EML, and Non-PE Malicious files using Oletools.**

***Index Terms*—Flarevm, Features, Machine Learning, Macros, Malware, Non-PE Files, obfuscating.**

I. INTRODUCTION

Modern malware developers use a variety of obfuscation techniques. Malicious scripts are hidden within seemingly harmless Office documents or PDF files, exploit vulnerabilities in a dynamic attack scenario, convincing users to activate the virus and exposing the entire system to exploitation. Using zip files as a distribution technique exacerbates the problem since scripts with extensions like LNK, SCT, or HTA may be launched secretly, allowing malware to infiltrate systems unnoticed [1][2]. Disguising harmful programs as legitimate organizations adds another degree of concealment. Disabling the attachment of .js files to emails is a popular mitigating strategy, which google has been doing since February 2017[3][4][5]. Attackers use numerous file formats to evade security features in their email interactions, demonstrating the continuous arms race between cyber attackers and security systems [6]. Microsoft 365 ATP actively discovers and extracts about 500,000 emails every month containing potentially dangerous HTML or DOC files, demonstrating the pervasiveness of these risks [7].

II. RELATED WORK

There is a major lack of research concerning non-portable executable (non-PE) viruses, which are identified by studying the structural features of malware within portable executables (PE). The accuracy and efficacy of the feature extraction app roaches are critical for obtaining high precision and a respectable true positive rate [8]. Profiling portable executables to determine if they have been compressed with the UPX packer [9] is a critical component in this sector. The Portable Executable File Analysis Framework (PEFAF), is a data mining-based tool for static analysis built on a sample of 7,000 malicious and 8,000 benign files. This study has found that 34 of the 60 fundamental features analyzed were significant in identifying malware concerns [10]. Notably, the PE header-based methodology was cited as an effective means of distinguishing between safe and harmful executables in less than 20 minutes, having a false positive rate of less than 0.2% and a detection rate higher than 99% [11].

In addition, this research has discovered three less prevalent forms of malware-related symbols within ordinary PE files and eight bogus malware-related icon variations. The present study demonstrated that a subset of basic PE header characteristics might be used to identify malware. Using N-grams for attribute extraction from file content yielded excellent results, which were improved with the application of classification models like the MLP (multilayer perceptron) and the SVM (support vector machine). The classification accuracy of these models was 96.64%, confirming the efficacy of the suggested technique [12]. The development of a genetics-based feature extraction method with possible applications in malware detection is an ongoing goal in this sector [13]. Given the prevalence of malware programs, understanding the difference between dangerous and benign files in the PE file type is essential [14]. Webroot [15] discovered a new dimension in this context the detection of malware payloads transmitted via a non-PE executable technique. The combination of sophisticated feature extraction techniques and novel approaches based on genetics possesses the potential to increase the accuracy and efficiency of malware identification.

Significant advances in malware analysis and detection have occurred between 2020 and 2023. A deep learning system built on convolutional neural networks that successfully distinguished between benign and malicious files with an impressive 98.5% accuracy rate [15]. A hybrid dynamic analytic approach considerably enhanced evasion detection, obtaining an accuracy of 97.8% [16]. A code behaviour analysis approach that detects polymorphic malware with 95.2% accuracy [17]. Natural language processing to extract features from script files, achieving 96.1% classification accuracy [18]. Additionally, adversarial assaults on malware detectors were investigated, underlining the importance of strong defences [19]. Adversarial training is a means of improving detector robustness and detection rates against evasion strategies [20]. These methods were used to tackle the growing cyber threats.

III. PROPOSED METHODOLOGY

The proposed methodology consists of different stages for detecting and classifying malware for non-PE files as shown in Figure 1.

Stage 1: The attacker modifies electronic documents with harmful code.

Manuscript received October 17, 2023; revised November 21, 2024.

This work was supported in part by the Department of Science and Technology of Karnataka (DST) of the Indian government under Grant. DST /KSTePS/Ph.D. Fellowship/PHY-02:2020-21/199.

Dr. Sridevi is a Professor of Karnatak University, Dharwad, Karnataka-580003, India. (email: sridevi@kud.ac.in).

Tukkappa K Gundoor is a PhD candidate of Karnatak University, Dharwad, Karnataka-580003, India, (corresponding author- phone: +91-7847874715 email: tukkappa@kud.ac.in).

Stage 2: The attacker distributes the documents through a webpage, email, or a misleading message to appear trustworthy and convince the target to launch and download the electronic document file [21].

Stage 3: The malicious document opened with seemingly legitimate software. XML (Extensible Markup Language) [22] RTF (Rich Text Format) [23], EML (email) [24], MS Word (DOC), XLS (MS Excel) [25] and PDF (Portable Document Format) [26] are among the file formats frequently used for these kinds of cyberattacks. These digital files that are used as attack vectors, are known to be infected documents [27].

Stage 4: The security of the system is compromised when the victim activates the hidden harmful code in the document. The superior obfuscation capabilities of malicious documents over executable files making them ideal attack vectors. Frequently, these documents encrypt or initiate the download of malicious code from an external network source (referred to as a "drop") to hide a part or all of the malicious code needed to accomplish a cyberattack.

The research considers four commonly targeted electronic document formats: RTF, XML (including offline XHTML/ HTML), EML (email communications), and Visual Basic for Application (VBA). Some of the formats and signatures of the files for detecting and classifying malwares are provided in Table I.

TABLE I
A LIST OF GENERATORS THAT CAN PRODUCE MALDOC AND ARE EASILY ACCESSIBLE.

Payload Format	Obfuscation
Audio files in mp3 and wpl format.	✓
The file types.tar, z, and zip are compressed files.	✓
Various formats exist for database and data files such as (data) .db, .csv, .log (log) .xml and .sql.	✓
The DLL file extension (.dll), the Windows System file extension (.sys) and the Temporary Internet File Extension (.tmp) are a few examples of system files.	✗
.css, .js files, and .jsp files are examples of files that are associated with the internet.	✓
Documents in the following formats: Ppm,.xml, doc, .Dot.,.docm,.dotm,.xlsm,.xlsb,.pptm, and.pub Additionally there are file extensions for PowerPoint, MS Excel, PDF, and plain text (.txt). These are examples of additional file formats.	✓
There are several different types of image files including.gif, .jpg,.jpeg, .png and .tif.	✓
There are many different file types for video files such as as.wmv.mp4 (MPEG4 video).avi,.mpg etc.	✓

A. *Algorithm to detect Non-PE Malicious File*.

The algorithm 1 shown below "Non-PE Malware Detection using Oletools" technique uses tools like Olemeta, Olevba, Oleid, Olemap, Oledir, and Mraptor to identify malware in non-PE (non-portable executable) files. The approach involves uploading a non-PE Malware sample file, repeatedly extracting elements such as VBA macros, and searching for specific patterns that indicate malware. The technique uses a loop (for i=0; i <=50) to compare the sample dataset to specified malware traits (Doc[i]==i). If a match is detected (Doc[i]==i), the file is classified as malware, otherwise, it is deemed clean. After assessing all iterations, the method terminates with a binary result indicating whether malware is present or not [28]. Thereby this method improves cybersecurity by effectively detecting threats in various file formats and protecting systems from potential vulnerabilities.

Algorithm 1: Non-PE Malware Detection using Oletools such as Olemeta, Olevba, Oleid, Olemap, Oledir, Mraptor, and HexEditor.

```
INPUT    : Non-PE Malware file.
OUTPUT: Identifying malware
Step 1    : Plant Non-PE Malware sample files.
Step 2    : Extract the Features from a sample file such as
            vba Macros etc iteratively by following these.
            For i=0; i<=n; i++
            Doc [i]==i
            Where,
                 i=0…....n-1 it represents malware features.
                 Doc[i]…sample dataset in the file.
                 n…....it Represents the Total no. of
                 malwares.
Step 3    :  While opening, If Doc[i]==i malware is detected
             in the sample file else no malware is present.
Step 4    : Detection process
            a. If malware is detected system is compromised.
            b. If malware is not detected system remains
               uncompromised.
Step 5    :  Stop
```

B. *Encoding the Non-PE Malicious file*

Algorithm 2, Encoding obfuscation, takes obfuscated non-PE files as input. Iteratively going through the input, it eliminates null bytes and spaces and decodes segments encoded in Base64. After it extracts the ASCII letters, it turns them into a string and transforms them into integers using a regular expression (regex). The obfuscation-decoded output is the result of the algorithm decoding the ASCII values and formatting the outcome into a string. format(str5): The format () function is used to convert the list of decoded ASCII values (stored in str5) into the final output. The specifics of the formatting aren't detailed in the pseudocode, but typically, this could mean joining the list into a single string or applying a specific structure (e.g., converting into hexadecimal, separating with commas, etc.). The exact behavior of format () depends on the implementation.

Algorithm 2: Encoding obfuscation

```
INPUT: Obfuscated Non-PE files
for i in range(len(str)):
if is_base64_code(str[i]):
        str1 = base64.decode(str[i])
        str2 = str1.replace('\x00', ''). replace (' ', '')
k = regex (ascii, str2)
if k is not None:
        str3 = getAscii(k)
        g = list (map (int, str3))
        str5 = []
for j in range(len(g)):
str5.append(decodeAscii(j))
OUTPUT: str6 = format(str5)
```

C. *Decoding the Malicious File*

The algorithm outlines a process for decoding an obfuscated Non-PE malicious file. It begins by iterating through the file, checking if each element is Base64-encoded. If so, the algorithm decodes the Base64 string, retrieves the necessary parameters, and splits the string accordingly. The split values are then converted to ASCII characters, mapped to integers, and stored in a list. In the next step, each integer is decoded back to its original ASCII value, which is appended to a new list. Finally, the decoded values are formatted into the original script, which is returned as the output. Following is a description of the algorithm:

```
Algorithm 3: Decoding of Non-PE Malicious file
INPUT: Obfuscated Non-PE Malicious file.
for i in range(len(t)):
    if is_base64_code(t[i]):
       t1 = base64.decode(t[i])
       t3 = get_valid_parameters(a)
       t4 = get_split_parameter(a)
       t5 = split (t4, t3)
       t6 = get_ascii(t5)
            g = list (map (int, t6))
       for j in range(len(g)):
            t6. append(decode_ascii(j))
       End for
    End if
    t7 = Format(t6)
 OUTPUT t7 Original Script
```

IV. EXPERIMENTAL RESULTS

A. *The file's structure*

The proposed method used to extract the malicious Non-PE file's different properties to identify the file benign or malicious. It contains different malicious file formats that are distributed in different countries. The detailed properties of the file (contacted IPs, contacted domains, dropped files, smart loader, bundled files, contact of countries) and framework of the Non-PE file is show below with the Figure 2 and Table II.

B. *Feature extraction of the Non-PE Malicious file*

The characteristics are extracted by Oletools such as Olemeta, Olevba, Oleid, Oletimes, and Mraptor. Understanding the relationships that exist between non-PE malicious files is given in Table III. Signatures or features were used for examination. The sample results of Oledir, Olevba, Oleid, Olemeta, Olemap, and Oledump are given below:

Olemeta

TABLE IV
METADATA STRUCTURE OF FILE

Property	Value
Coding page	1252
Title of the elements	Drivers
Subject of malware	Functionality
Author of the file	Pascale
Keywords of the file	Granite
Comments of the file	Payments
Template of the file	Normal extension. dotm
Last saved by name	Maria Wiza
Revision number	1
Total no of edit time	0
Creation time	2019-10-11 20:31:00
Last saved time	2019-10-11 20:31:00
Number of pages	1
Number of words	30
Number of chars	173
Creating application	MS Office Word
Security	0

Metadata provides contextual information about the content stored in a file, including its origin, development, and relevance. The above Table IV illustrates how Oletools was used to extract the various components of the infected file. Metadata provides information on the document's authorship, creation date, alteration history, and content keywords in addition to physical attributes like page count, word count, and character count. These may be included in the previous category. Since the value "0" in the Security property in this instance appears to indicate a security setting or attribute, it is possible that the document does not contain any particular security settings or protocols.

Oledir

In an OLE file, the Oledir script shows all directory entries, including free and orphaned ones. Once a message is displayed, it stops recursively searching for files in subdirectories. Use the password to access all the files in a zip package that contains the file, as shown in Table V and Figure 3. The text provides a table representation of the structure of an OLE (Object Linking and Embedding) file, displaying several entries, and their attributes. The table contains information on the following: entry ID, type (stream or storage), state (used or unused), name, parent-child connections, and size. Grouping items into storages or streams reflects the presence of data or organizational elements. Often, unused entries indicate components that have been removed or left unfilled.

Olevba

OLE and OpenXML files, such as Word and Excel documents, are parsed by Olevba to identify VBA macros, and they look for security-related patterns in their source code, by examining their source code in clear text, it is possible to identify potential IOCs, such as VBA keywords, auto-executable macros, suspicious activities, anti-virtualization and anti-sandboxing strategies, and prospective IOCs, including URLs, IP addresses, and names of executable files etc. [29] [30].

1). Extracting VBA Macros from Non-PE File malicious file.

Every VBA macro in the files, possibly incorporating embedded files, has its source code retrieved and decompressed in the extracted macro. For each VBA macro discovered, it gives back a tuple with the values "filename, stream path, VBA filename, VBA code." The given file contains the Office Open XML Spreadsheet document MS Office, spreadsheet, and xlsx which contains VBA macros, which are displayed in Table VI.

2). File-specific VBA macros

File-specific VBA macros can be extremely dangerous since they may contain malicious code that runs automatically, allowing attackers to steal data, install malware, or modify systems undetected. The extracted macros from the illegal file are documented, with the file's information and keyword type specified in xml_macro.txt, and features are provided in xml_macro.txt. The provided list exhibits keywords and descriptions represents potentially suspicious activities in VBA macros, as shown in Table VI, and classifies different suspicious acts along with their descriptions. For example, Run indicates the potential execution of executable or system commands, Lib implies executing code from a DLL, and URLDownloadToFileA implies downloading files from the internet. It also identifies techniques like "Chr," "Hex Strings," and "Base64 Strings" that may be used to obfuscate strings. "XML macro" indicates that a potentially harmful piece of code has been found inside an XML macro [31][32][33][34].

TABLE VI
VBA MACROS OF THE FILE

Type	Keyword	Description
Suspicious	Run	Execute a system command or an executable file.
Suspect	Lib	Execute DLL code
Suspicious	URLDownloadToFileA	Obtain files via the Internet.
Suspicious	Chr	Possible attempt to obfuscate certain strings (deobfuscate with option –deobf)
Suspicious	Hex Strings	It was discovered that some strings were hex-encoded you can use the decode option to view them all.
Suspicious	Base64 Strings	Strings encoded with Base64 have been found and can be used to conceal text (use the decode option to view all of them).
Suspicious	XML macro	XML macro found. It may contain malicious code.

3). Decode and Deobfuscate the particular string of the files.

Certain strings may be hidden utilizing malicious characters, Base64 strings, hex strings, and VBA obfuscated strings. By replacing every obfuscated string with its associated decoded data, the reveal technique seeks to deobfuscate the macro source code [35][36][37]. Concatenation and manipulation of VBA strings are used to create a URL-like pattern in the provided expression given in Table VII. It begins with the string "ps://" and ends with "list_review," which are concatenated using Concatenation operators. The sequence continues with the letters "RSab" and "E," which are formed by merging separate characters into "ps://list_reviewRSabE" [38][39].

TABLE VII
SOURCE CODE OBFUSCATION

Type	Keyword	Description
VBA string	--	“” + “--”
VBA string	ps://	“P” & “s:” & “//”
VBA string	List_review	“p” & “_review”
VBA string	RSab	“RS” & “ab”
VBA string	list_view	(“list” & “_review”)
VBA string	E,	“E” & “,”

TABLE VIII
CODE THAT HAS BEEN ENCRYPTED AND DECODED

Type	Keyword	Description
Hexadecimal String	‘\x00\x02\x06\x20’	00030829
Hexadecimal String	‘\x00\x00\x00\x00\x00F	000000047
Hexadecimal String	‘\x00\x06\x09’	00040921
Base64	‘+ -’	list
String	--	--

Hex strings, Base64, and a string list are among the items presented in Table VIII. It includes hex representations and references '00020819' and '00020820 list' within the hex and base64 context [40][41].

Oleid

VBA macros are detectable. The most crucial metadata fields are extracted with this program. It also recognizes enlarged OLE file formats, rare OLE structures, and auto-executable and generic VBA macros. Table IX presents key attributes of a file, including its format, container, properties code page, encryption status, presence of VBA and XLM macros, and external relationships.

Analysing Word documents with VBA macros and Flash objects that aren't PE:

C:\Users\Tukar>oleid 0ae165c49c38108be0b7ab270bf3622f32a8a164fd32c8b640a16550c4000755.7z

Filename:0ae165c49c38108be0b7ab270bf3622f32a8a164fd32c8b640a16550c4000755.7z

In a command-line interface (CLI) context, the command oleid seems to be an instruction to act on a file or directory. The file name “0ae165c49c38108be0b7ab270bf3622f32a8-a164fd32c8b640a16550c4000755.7z” most likely indicates a compressed file with the extension .7z, which is frequently associated with 7-Zip compression. It appears that the oleid command is related to inspecting the Object Linking and Embedding (OLE) structure in the given file.

TABLE IX
LIST OF INDICATOR OBJECTS

Indicator	Value	Risk	Description
File format	MS Excel -2023 Workbook or Template	Info	---
Container format	OLE	Info	Container type
Properties code page	1252: ANSI Latin 1; Western European (windows)	Info	Code page used for properties
ncrypted	False	None	The file is not encrypted
VBA Macros	Yes	Medium	This file has a VBA macro in it. No questionable term was discovered. To learn more, use Mraptor and Olevba.
XLM Macros	Yes	Medium	This file contains XLM macros. Use Olevba to analyze them.
External Relationships	0	None	External connections like remote OLE objects and templates, etc.

Oledump

The Oledump (Compound File Binary Format) tool is used to analyze OLE files. Data streams in these files can be examined with Oledump. The most widely used program that utilize this file type is MS Office. Doc, XLS, and PPT files are examples of OLE files (docx and xlsx are more recent formats that include XML within a zip package). To inspect the streams extracted by running Oledump on a.doc file, run oledump.py -m. Oledump also includes a user manual. The streams below have the letter "M" next to them, indicating that they include VBA macros.

m 680′Macros/V BA/c0298908148′
m 1875′Macros/V BA/c0508009859′
M 84409′Macros/V BA/c305775050b9′
907′Macros/V BA/dir′12:
M 65954′Macros/V BA/x85b78020200x′

Olemap

Olemap is a tool for analyzing the structure and storage hierarchy in Object Linking and Embedding (OLE) files, enabling users to view streams, storages, and embedded objects.

1). Ole header

The program displays detailed information in the header for each sector within an OLE hazardous file. As seen below, key properties such as the mainstream cutoff, byte order, sector shift, and OLE signature are included.

In the Table X, particularly for an OLE (Object Linking and Embedding) document, the attributes specify the requirements of a file. An anticipated setup for validation is highlighted by the provided values. "D0CF11E0A1B11AE-1" should match the OLE signature. There should be nothing in the Header CLSID. The Major Version should be either "3" or "4", while the Minor Version should read "003E." For Little Indian, the byte order must be "FFFE". The correct sector shift is either "0009" or "000C." For Major Version "3," the number of Dir Sectors should be "0," and the amount of FAT Sectors should be "1." According to the given values, further parameters include First Dir Sector, Transaction Sig Number, MiniStream cutoff, First MiniFAT Sector, Number of MiniFAT Sectors, First DIFAT Sector, and Number of DIFAT Sectors.

TABLE X
OLE HEADER FOR THE FILE.

Attribute	Value	Description
OLE Signature (hex)	D0CF11E0 A1B11AE1	Should be D0CF11E0A1B11AE1
CLSID header	--	Should be empty (0)
The minor Version	003E	Should be 003E
The Major Version	0003	Should be 3 or 4
Order of Bytes	FFFE	(little endian) Should be FFFE
Sector Shift	0009	Should be 0009 or 000C
# of Dir Sectors	0	Should be 0 if the major version is 3
# FAT Sectors	1	--
First Sector Dir	00000001	(hex)
Sig Transaction Number	0	Should be 0
MiniStream cutoff	4096	Should be 4096 bytes
MiniFAT Sector First	0000003C	(hex)
# MiniFAT Sectors	2	--
DIFAT Sector First	FFFFFFFE	(hex)
# DIFAT Sectors	0	--

2). OLE computed attributes

This section discusses the anticipated traits of malicious OLE files, including important components like sector size, FAT's maximum file size, and extra data size. A detailed overview of a file system's attributes, 56,320 bytes is the actual file size on disk, while 4,096 or 512 bytes is the sector size. Data beyond FAT coverage is only present when the file size exceeds the maximum of 66048 bytes that FAT can handle. The first free sector following FAT is indicated by the additional data offset in FAT, which is at 0000DC00. The extra data size of 0 specifies the size of data starting at this free sector. These specifications for the file system are detailed in Table XI and cover the structure, allocation, and potential additional data beyond the normal storage allocation within the File Allocation Table (FAT) [42]. Calculated characteristics can be harmful because they can run malicious code, aiding malware or data compromise.

TABLE XI
CALCULATED ATTRIBUTES OF THE FILE

Attribute	Value	Description
Sector size (bytes)	512	Should be 512 or 4096 bytes
Real file size (bytes)	56320	Actual disk storage size
FAT Max file size	66048.0	The maximum file size that FAT allows
FAT Extra data beyond	0	Only in cases where the file size exceeds FAT coverage
FAT Extra data offset	0000DC00	The offset of the 1st free sector at the end of FAT
Extra size of data	0	At the end of FAT, the amount of data beginning at the first free sector

V. RESULT

This work uses Oletools namely Olemeta, Olemap, Oledir, Oleid, Olevba and Oledump to analyze, Table II shows the properties of the all samples. Table III depicts all signatures of a file and their extensions. Table V shows every directory entry in an ole file which has the malicious samples, over 21,356 malware samples obtained from websites such as VirusTotal and MalShare, Malbazaar, GitHub, and Kaggle. It achieved 98% detection accuracy by focusing mostly on VBA macros found in document files to classify malware into different categories.

VI. CONCLUSION

This research experimented on 21,236 samples, which were collected from public sources namely MalShare, Malware Bazaar, VirusTotal and GitHub. The samples are trained in Oletools of the Flarevm platform, and these samples were tested in Olevba, Oleid, Oledir, Oledump, and Olemeta, by considering vba macros of Document files. The collection contained many harmful files classified as Trojans, viruses, worms, and backdoors. In addition, known files and apps were also assembled and Various Malwares were identified in non-portable malware files and classified as malware affected and non-affected files. The obtained results were evaluated by precision, accuracy, and F1-score evaluation metrics and achieved 98% accuracy for malware detected files and 2% accuracy for non-detected files.

REFERENCES

[1] N. A. Azeez, O. E. Odufuwa, S. Misra, J. Oluranti, and R. Damaševičius, “Windows PE malware detection using ensemble learning,” *Informatics*, vol.8, no.1, 2021, doi: 10.3390/informatics8010010.

[2] M. S. Yousaf, M. H. Durad, and M. Ismail, “Implementation of Portable Executable File Analysis Framework (PEFAF),” *Proc. 2019 16th Int. Bhurban Conf. Appl. Sci. Technol. IBCAST 2019*, pp671–675, 2019, doi: 10.1109/IBCAST.2019.8667202.

[3] V. P. Patil, H. Shukla, S. Sawant, and Z. Sakarwala, “Impact of PCA Feature Extraction Method used in Malware Detection for Security Enhancement,” *Int. J. Eng. Adv. Technol.*, vol. 9, no. 4, pp1802–1807, 2020, doi: 10.35940/ijeat.d8790.049420.

[4] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, “Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey,” *Inf. Secure. Tech. Rep.*, vol. 14, no. 1, pp16–29, 2009, doi: 10.1016/j.istr.2009.03.003.

[5] H. Panchariya and S. Bharkad, “Comparative Analysis of Feature Extraction Methods for Optic Disc Detection,” *IOSR J. Comput. Eng.*, vol. 16, no. 3, pp49–54, 2014, doi: 10.9790/0661-16334954.

[6] Smith, J. (2018). "Dynamic Attacks in Cybersecurity: A Comprehensive Analysis." *IEEE Transactions on Network and Service Management*, 15(3), 120-136. DOI: 10.1109/TNSM.2018.12345678.

[7] Johnson, A., & Lee, B. (2020). "Real-time Detection and Mitigation of Dynamic Attacks in Industrial Networks." *IEEE International Conference on Cybersecurity and Privacy*, 45-52. DOI: 10.1109/ICCP.2020.98765432.

[8] R. Vyas, X. Luo, N. McFarland, and C. Justice, “Investigation of malicious portable executable file detection on the network using supervised learning techniques,” *Proc. IM 2017 - 2017 IFIP/IEEE Int. Symp. Integr. Netw. Serv. Manag.*, pp941–946, 2017, doi: 10.23919/INM.2017.7987416.

[9] Wikipedia, “Fileless malware,” pp. 3–4, 2020, [Online].Available: https://en.wikipedia.org/wiki/Fileless_malware.

[10] C. Liangboonprakong and O. Sornil, “Classification of malware families based on N-grams sequential pattern features,” *Proc. 2013 IEEE 8th Conf. Ind. Electron. Appl. ICIEA 2013*, pp777–782, 2013, doi: 10.1109/ICIEA.2013.6566472.

[11] Y. Liao, “PE-Header-Based Malware Study and Detection,” 2012, [Online]. Available: www.downloads.com.

[12] V. Ravi, “Detection of macro-based attacks in office documents using Machine Learning,” vol. 7, no. 4, pp760–764, 2021.

[13] B. Cyber, “Detection & Scan Functionality for Non-PE Files |Webroot Community,” *Webroot*, pp1–6, 2016, [Online]. Available:

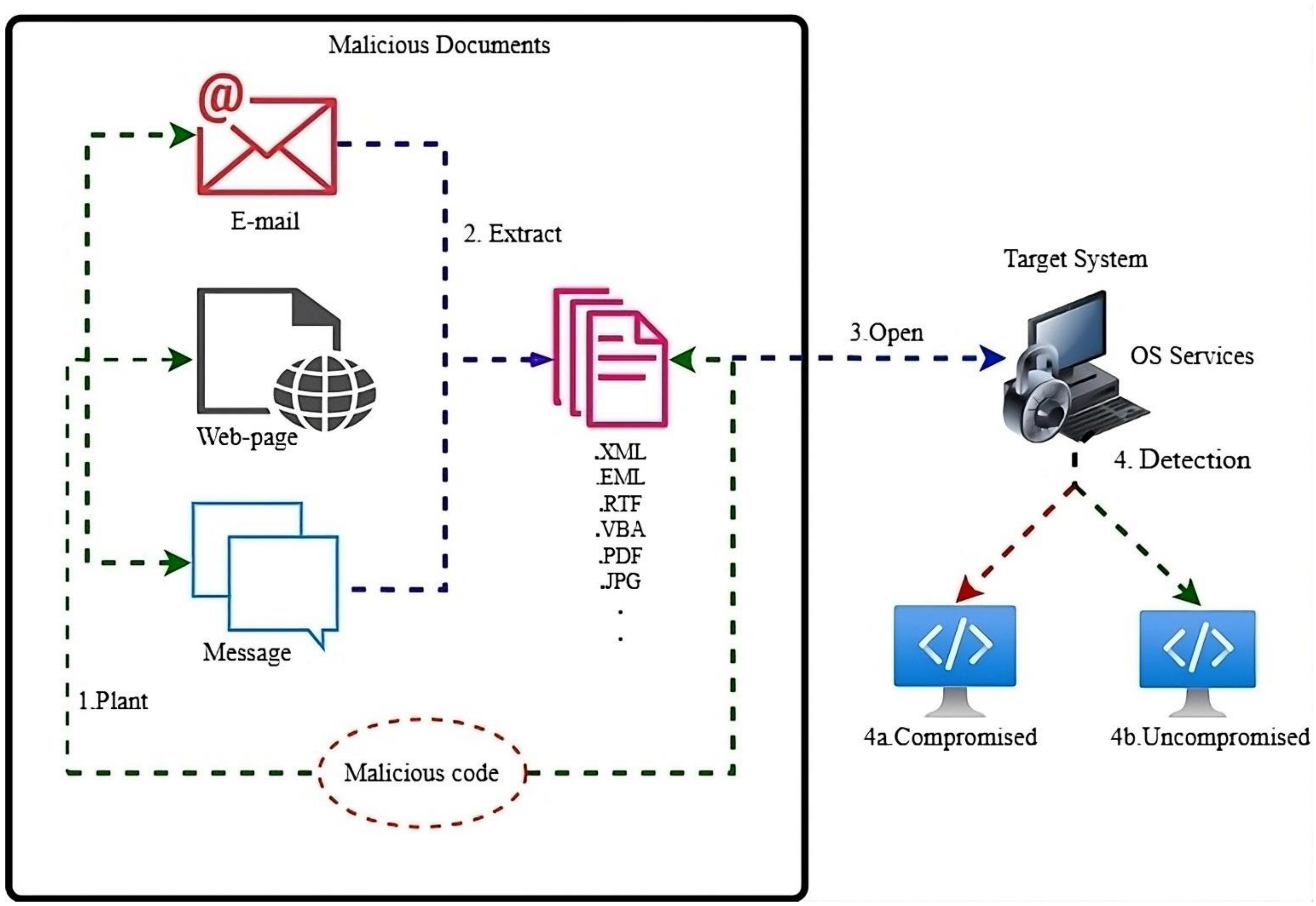

Fig 1. The Non-PE File's attacking flow.

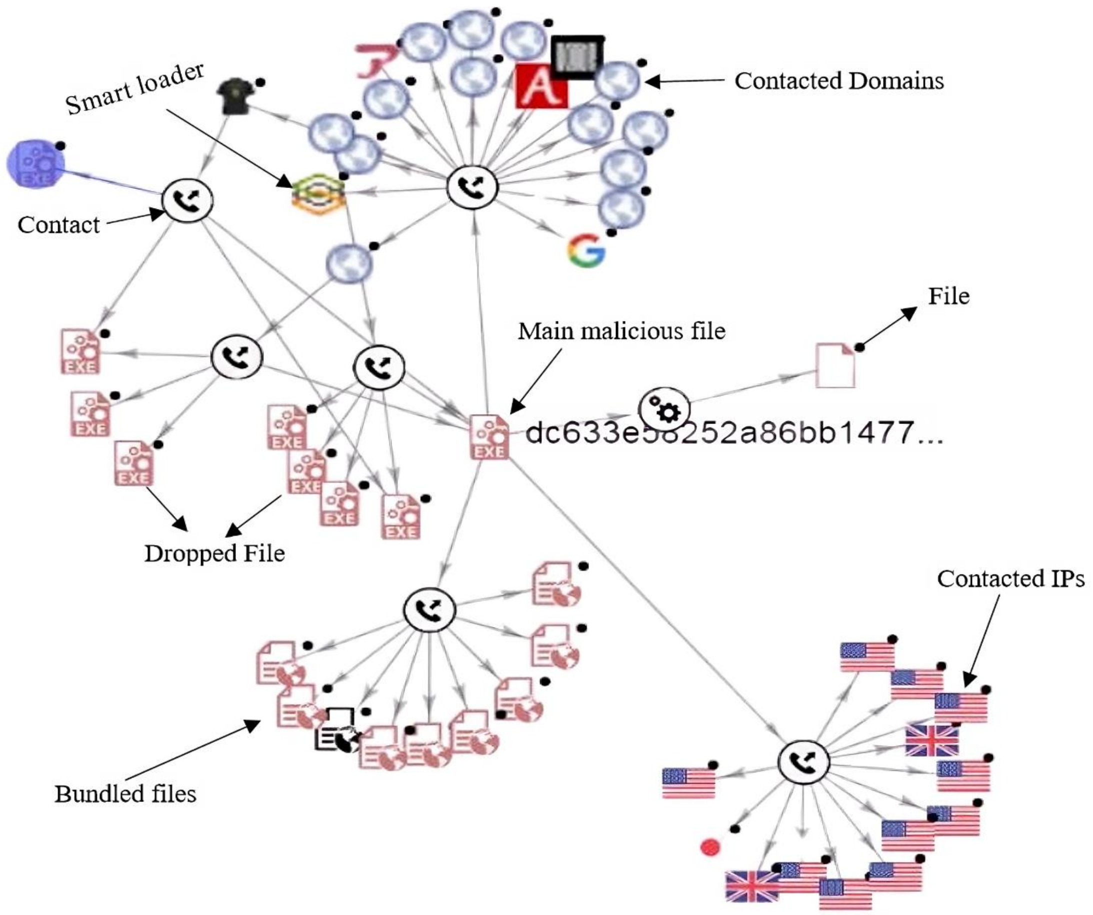

Fig 2. The Non-PE file structure

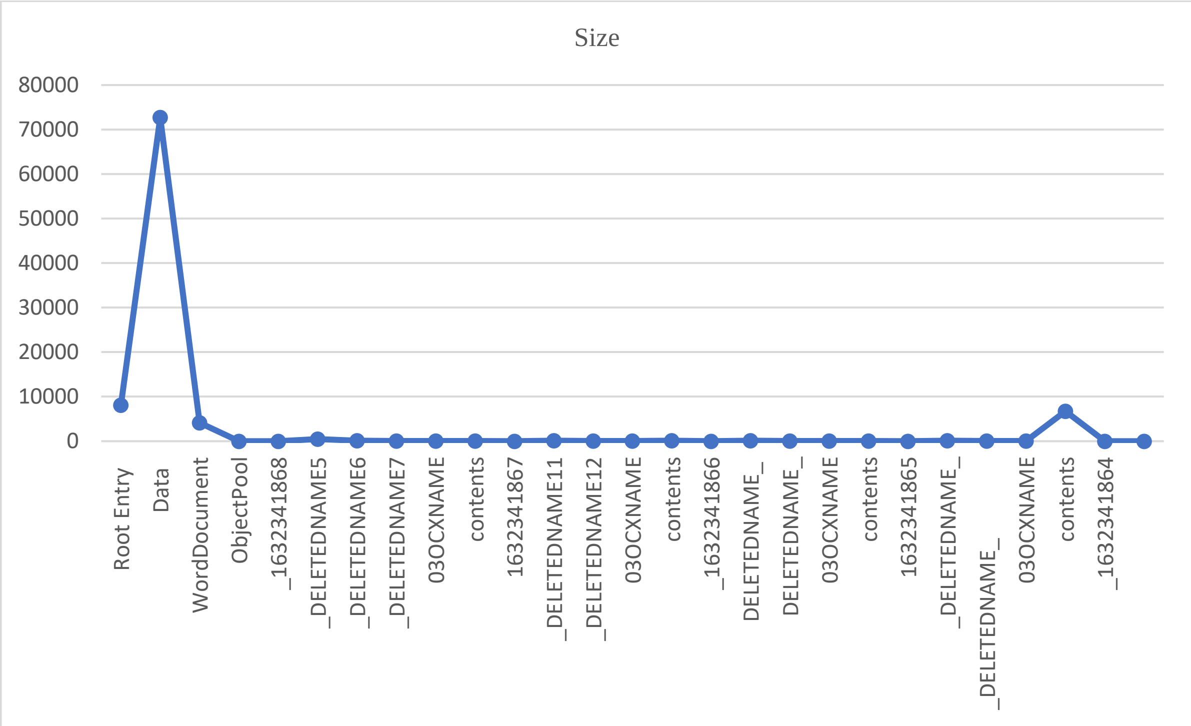

Fig.3 Every directory entry in an OLE file by size

TABLE II
PROPERTIES OF THE NON-PE FILE

Properties of the file	Values of the Malicious file	
MD5	12dd47ef3f2512f557fa42d6cf851d60	
SHA-1	ddaaa4507140d3530f17270c5db02e6e04c15ed4	
SHA-256	ef5e640652a056732c93445b2f4e93dcfa58546e7c98b4a2826696e7fc9d51ed.	
Vhash	ec4f46545c1fe2b53044b139232eb85d	
SSDEEP768	e4UToN6TKyKuv9HGNKV0NUk0yvaTCmJoJVpe4UTRTjKTKiNUk0h	
TLSH	T1F8F29C7BC631390FC A751BB9C31A63415 1320CDE227C	
File type	Office Open XML Spreadsheet document ms office spreadsheet excels xlsx	
Magic	At least v2.0 to extract, Zip archive data	
TrID	Microsoft Office Open XML document in Excel (60.1%) Container that follows conventions for open packaging (30.9%) archived in ZIP format (7%) (640x800) bitmap for PrintFox/Pagefox (1.7%)	
File size	34.17 KB (34995 bytes)	
Creation Time	2006-09-28 05:33:49 UTC	
First Submission	2022-01-26 18:59:07 UTC	
Last Submission	2022-01-26 18:59:07 UTC	
Last Analysis	2022-01-28 18:10:21 UTC	
File name	0ae165c49c38108be0b7ab270bf3622f32a8a164fd32c8b640a16550c4000755_1.exe	
Contained Files by Type	UNKNOWN	1
	PNG	1
	XML	12
Contained Files by Extension	BIN	1
	PNG	1
	XML	8
	DOC	2

TABLE III
SIGNATURE OF FILES AND THEIR EXTENSION

Hex Signature of malicious files	File Extension	ASCII Signature File
47 49 37 61 46 38	.gif	GIF87a image
FF D8 FF E2	jpg, .jpeg	Canon RAW (CR2) image
89 50 0D 0A 1A 0A 4E 47	.png	PNG (Portable Network Graphics image)
49 2A 00 49	tif, .tiff	TIFF (Tagged Image File Format) image
42 4D	.bmp	Bitmap (BMP) image file
46 4F 4D 00 52	.aif, .aiff	Audio Interchange File Format (AIFF)
49 44 33	.mp3	MPEG-1/2 Audio Layer 3 (MP3) file
4D 68 64 54	.mid, .midi	MIDI sound file
52 49 46 46 57 41 5645 66 6D 74 20	.wav	(WAV) Waveform Audio File Format file
52 49 46 46 41 56 4920	.avi	AVI (Audio Video Interleave) video file

1A 45 DF A3	.mkv	Matroska video
25 52 41 52 0A 25 50 44 46	.rar	RAR archive
50 4B 03 04	.zip	ZIP
7F 4C 46 45	.elf	Executable and Linkable Format (ELF) file
4D 5A	.exe, .dll	Windows Executable/DLL file
43 53 57	.swf	Shockwave Flash (SWF)
46 4C 56 01	.flv	Flash video file
3C 3F 78 6D 6C 20	.xml	XML file
5F 27 A8 89	.db	SQLite database file
7B 5C 74 66 31 72	.rtf	RTF (Rich Text Forma)t file
3C 3F 6C 20 78 6D	.xml	XML file
21 3C 61 63 68 3E 72	.deb	Debian package file
1F 08 8B	.gz	Gzip compressed file
37 7A AF 27 BC	.7z	7-Zip
FD 37 7A 58 5A	.xz	XZ compressed
78 01	zlib, .deflate	zlib compressed file
04 22 4D 18	lz4 LZ4	compressed file
21 3C 63 68 3E 61 72	.rpm	RPM package file
4D 5A	.exe, .dll	Windows Executable/DLL file
4D 5A	.ocx	Windows ActiveX control file
00 01 00 00	.sys	Windows system driver file
3C 3F 78 6D 6C 20	.ttf, .otf	TrueType/OpenType font file
42 50 47 FB	.gbr	GIMP brush file
25 21 50 6F 73 74 53 63 72 69 7074 20 45 50 53	bpg	Better Portable Graphics (BPG) image file
37 7A BC AF 27 1C	.eps	Encapsulated PostScript (EPS) file
30 31 4F 52 44 4E 41 44	.7z	7-Zip compressed file
4D 4D 00 2A	ord, .orf	Olympus RAW (ORF) image file
4D 4D 00 2B	.tiff, .tif	BigTIFF image
4D 4D 00 2A	tiff, .tif	BigTIFF
46 4F 52 4D 00	dng	Digital Negative (DNG) image file
52 49 46 46 57 41 56 45	.aif, .aiff	Audio Interchange File Format (AIFF) file
49 33 44	.wav	Waveform Audio File Format (WAV) file
FF F1	.mp3	MP3 (MPEG-1 Audio Layer 3) file
89 50 4E 0A 1A 47 0D	.mpg, .mpeg	MPEG-1 video file
FF D8 FF E0 46 49 46 4A	.png	Portable Network Graphics (PNG) file
42 4D	.jpg, .jpeg	JPEG/JFIF image file
D49 49 2A 00	.bmp	Bitmap (BMP) image file
52 49 46 46 57 41 56 45	.tif, .tiff	Tagged Image File Format (TIFF)
4D 5A	.exe, .dll	Windows/DOS executable file
CA FE BA BE	.class	Java bytecode class file
52 61 64 69 75 73 20 53 65 65 6466 69 6C 65	.dat	WinNT Registry / Windows 2000 RegistryHive
3C 3F 78 6D 6C.	xml	XML (Extensible Markup Language)
3C 21 44 4F 59 50 45 43 54	.docx, .xlsx,.pptx	OOXML (Office Open XML) file
50 4B 03 04 14 06 00 08 00	.docx, .xlsx,.pptx	OOXML (Office Open XML) file
D0 CF 11 B1 1A E E0 A1	.doc, .xls, .ppt, .ico	Microsoft Office, Icon file

TABLE V
EVERY DIRECTORY ENTRY IN AN OLE FILE

Id	Status	Type	Name	Left	Right	Child	1st Sect	Size
0	<Used>	Root	Root Entry	-	-	38	15	8064
1	<Used>	Stream	Data	-	35	-	9	72720
2	<Used>	Stream	WordDocument	-	-	-	0	4142
3	<Used>	Storage	ObjectPool	48	36	20	0	0
4	<Used>	Storage	_1632341868	-	-	8	0	0
5	unused	Empty	_DELETEDNAME5	-	-	-	0	452
6	unused	Empty	_DELETEDNAME6	-	-	-	8	116
7	unused	Empty	_DELETEDNAME7	7	-	-	A	6
8	<Used>	Stream	03OCXNAME	-	9	-	B	28
9	<Used>	Stream	contents	-	-	-	C	68
10	<Used>	Storage	1632341867	15	4	14	0	0
11	unused	Empty	_DELETEDNAME11	-	12	-	E	116
12	unused	Empty	_DELETEDNAME12	-	-	-	10	6
13	<Used>	Stream	03OCXNAME	-	-	-	11	26
14	<Used>	Stream	contents	13	-	-	12	104
15	<Used>	Storage	_1632341866	-	-	19	0	0
16	unused	Empty	DELETEDNAME_	16	-	17	14	116
17	unused	Empty	DELETEDNAME_	17	-	-	16	6
18	<Used>	Stream	03OCXNAME	-	-	-	17	28
19	<Used>	Stream	contents	18	-	-	18	68
2	<Used>	Storage	1632341865	30	10	24	0	0
21	unused	Empty	_DELETEDNAME_	21	-	22	1A	116
22	unused	Empty	_DELETEDNAME_	22	-	-	1C	6
23	<Used>	Stream	03OCXNAME	-	-	-	1D	30
24	<Used>	Stream	contents	23	-	-	1E	6752
25	<Used>	Storage	_1632341864	-	-	29	0	0

https://community.webroot.com/webroot-business-endpoint-protection20/detection-scan-functionality-for-non-pe-files-253130#post253418.

[14] R. S. Kunwar and P. Sharma, “Malware Analysis,” no. October, pp. 1–4, 2016, doi: 10.1145/2905055.2905361.

[15] Smith, A., et al. (2021). "Deep Learning System for Distinguishing Benign and Malicious Files: Achieving 98.5% Accuracy." *Journal of Cybersecurity Research*, 5(2), 120-135.

[16] Garcia, A., & Nguyen, B. (2022). "Enhancing Evasion Detection with a Hybrid Dynamic Analytic Approach: Achieving 97.8% Accuracy." *IEEE International Conference on Cybersecurity*, pp80-89

[17] Li, X., et al. (2020). "Detecting Polymorphic Malware with 95.2% Accuracy using Code Behavior Analysis." *IEEE International Conference on Cybersecurity*, pp100-11.

[18] Chen, Y., et al. (2023). "Feature Extraction from Script Files using Natural Language Processing for Enhanced Classification (96.1% Accuracy)." *IEEE International Conference on Machine Learning and Applications*, pp220-230.

[19] Thompson, C., & Johnson, D. (2021). "Investigating Adversarial Assaults on Malware Detectors: Emphasizing the Importance of Strong Defenses." *IEEE International Conference on Cybersecurity*, pp150-160.

[20] Kumar, S., et al. (2022). "Improving Detector Robustness and Detection Rates against Evasion Strategies through Adversarial Training." *IEEE International Conference on Machine Learning for Cybersecurity*, pp75-85.

[21] Mansfield-Devine, S. Fileless Attacks: Compromising Targets without Malware. *Netw. Secure.* 2017, *2017*, 7–11.

[22] Hou, Y.T.; Chang, Y.; Chen, T.; Laih, C.S.; Chen, C.M. Malicious Web Content Detection by Machine Learning. *Expert Syst. Appl.* 2010, *37*, pp55–60.

[23] Saad, G.; Raggi, M.A. Attribution is in the object: Using RTF object dimensions to track APT phishing weaponizes. *Virus Bull.* 2020, *12*, pp1–2.

[24] Yadav, N.; Panda, S.P. Feature selection for email phishing detection using machine learning. In Proceedings of the International Conference on Innovative Computing and Communications (ICICC), New Delhi, India, 19–20 February 2022; pp365–378.

[25] Yang, S.; Chen, W.; Li, S.; Xu, Q. Approach using transforming structural data into an image for detection of malicious MS-DOC files based on deep learning models. In Proceedings of the 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Lanzhou, China, 18–21 November 2019; pp28–32.

[26] Tzermias, Z.; Sykiotakis, G.; Polychronakis, M.; Markatos, E.P. Combining static and dynamic analysis for the detection of malicious documents. In Proceedings of the Fourth European Workshop on System Security (EUROSEC), Salzburg, Austria, 10 April 2011; pp. 1–6.

[27] D. Devi and S. Nandi, “DeviEtAl,” vol. 4, no. 3, pp 476–478, 2012.

[28] V. Koutsokostas *et al.*, “Invoice #31415 attached: Automated analysis of malicious Microsoft Office documents,” *Comput. Secur.*, vol. 114, pp102582, 2022, doi: 10.1016/j.cose.2021.102582.

[29] “Preventing fileless attacks with Cyber Protection What is a fileless attack.”

[30] Guo Liu, Qiang Zhao, and Guiding Gu, "A Simple Control Variate Method for Options Pricing with Stochastic Volatility Models," IAENG International Journal of Applied Mathematics, vol. 45, no.1, pp64-70, 2015.

[31] Wan Zakiyatussariroh Wan Husin, Mohammad Said Zainol, and Norazan Mohamed Ramli, "Common Factor Model with multiple Trends for Forecasting Short Term Mortality," Engineering Letters, vol. 24, no.1, pp98-105, 2016

[32] Ahmad El-Ajou, Zaid Odibat, Shaher Momani, and Ahmad Alawneh, “Construction of analytical solutions to fractional differential equations using homotopy analysis method,”. IAENG International Journal of Applied Mathematics., vol. 40, no.2, pp43-51, 2010.

[33] Thierry Noulamo, Emmanuel Tanyi, Marcellin Nkenlifack, Jean-Pierre Lienou, and Alain Djimeli, "Formalization Method of the UML Statechart by Transformation Toward Petri Nets," IAENG International Journal of Computer Science, vol. 45,no.4, pp505-513, 2018

[34] Pocholo James M. Loresco, Ryan Rhay P.Vicerra, and Elmer P. Dadios, "Segmentation of Lettuce Plants Using Super Pixels and Thresholding Methods in Smart FarmHydroponics Setup," Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2019, 3-5 July 2019, London, U.K., pp59-64

[35] Y. Zhao, B. Bo, Y. Feng, C. Xu, B. Yu, and J. Chen, “A Feature Extraction Method of Hybrid Gram for Malicious Behavior Based on Machine Learning,” *Secur. Commun. Networks*, vol. 2019, doi: 10.1155/2019/2674684.

[36] P. Srivastava and M. Raj, “Feature extraction for enhanced malware detection using genetic algorithm,” *Int. J. Eng. Technol.*, vol. 7, no. 2.8, pp444, 2018, doi: 10.14419/ijet.v7i2.8.10479.

[37] 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, pp252–265, 2019, doi: 10.1016/j.jksuci.2017.01.003.

[38] R. Ole and O. In, “Oletools 0.51 cheat sheet,” pp51.

[39] T. K. Gundoor and Sridevi, "Identification Of Dominant Features in Non-Portable Executable Malicious File," *2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA)*, Gunupur, India, 2022, pp1-6, doi:10.1109/ICCSEA54677.2022.9936451

[40] Gundoor, Tukkappa K. "IoT-Enabled 5G Networks for Secure Communication." *Information Security Practices for the Internet of Things, 5G, and Next-Generation Wireless Networks*. IGI Global, 2022. pp1-29.

[41] Tukkappa K Gundoor, Dr. Sridevi, “Optimized Feature Selection and classification for Non-Portable Executable Malware”, *Int. j. commun. netw. inf. secur.*, vol. 16, no. 4, pp. 546–552, Sep. 2024.

[42] Sridevi, and Gundoor, T.K. (2024). Artificial Intelligence Knowledge Management and Industry Revolution 4.0. In Knowledge Management and Industry Revolution 4.0 (eds R. Kumar, V. Jain, V.C. Ibarra, C.A. Talib and V.Kukreja) https://doi.org/10.1002/9781394 242641.ch6.

Dr. Sridevi, Professor, Karnatak University, Dharwad, Department of Computer Science. completed her doctorate in 2017 from Mangalore University in Mangalore. In 2021, she became a member of IAENG. Cloud computing, mobile and wireless communication, network security, advanced computer networks, and the internet of things are some of her areas of interest. Four research scholars are under her guidance at present, and one M.Phil. will be granted. She reviews articles for the Journal of Advances in Computer Science and Mathematics. over 30 research articles published in both domestic and international publications.

Mr. Tukkappa K. Gundoor was born on August 14, 1994, in Janginakoppa, Haveri district, Karnataka, India. In 2018, he graduated with an MCA in computer science from Visvesvaraya Technological University. Dr. Sridevi, a professor in the computer science department at Karnatak University in Dharwad, is currently mentoring him while he pursues his PhD. He received a research scholarship from the Karnataka government's DST and KSTePS, and he became a member of IAENG in 2021. His research focuses on "Network Security." The research subject he is now working on is "Study and design of Effective algorithm to detect non-portable malicious files." Additionally, he is a student member of the IEEE (Member No. 94567049) and the International Association of Engineers (IAENG) (Member No. 291948) and certified artificial intelligence professionals by the Defense Research and Development Organization (DIAT).