Exploring the Malicious Document Threat Landscape: Towards a Systematic Approach to Detection and Analysis

Aakanksha Saha
TU Wien
Vienna, Austria
aakanksha.saha@seclab.wien

Jorge Blasco
Universidad Politécnica de Madrid
Madrid, Spain
jorge.blasco.alis@upm.es

Martina Lindorfer
TU Wien
Vienna, Austria
martina@seclab.wien

***Abstract*—Despite being the most common initial attack vector, document-based malware delivery remains understudied compared to research on malicious executables. This limits our understanding of how attackers leverage document file formats and exploit their functionalities for malicious purposes. In this paper, we perform a measurement study that leverages existing tools and techniques to detect, extract, and analyze malicious Office documents. We collect a substantial dataset of 9,086 malicious samples and reveal a critical gap in the understanding of how attackers utilize these documents. Our in-depth analysis highlights emerging tactics used in both targeted and large-scale cyberattacks while identifying weaknesses in common document analysis methods. Through a combination of analysis techniques, we gain crucial insights valuable for forensic analysts to assess suspicious files, pinpoint infection origins, and ultimately contribute to the development of more robust detection models. We make our dataset and source code available to the academic community to foster further research in this area.**

1. Introduction

Documents are a widely used method to deliver malicious payloads during a cyberattack: In 2016, the Microsoft Defender Security Research Team reported that 98% of Office-targeted attacks utilized malicious macros [43]. This dominance of macro-based threats was further corroborated by a recent ReasonLabs cybersecurity report, which identified them among the top 10 threats detected in 2022 [30]. Moreover, Microsoft's disclosure of 59 vulnerabilities, including zero-day exploits, in Word documents during 2023 highlights the criticality of analyzing these formats [22]. Despite the significant threat posed by malicious documents, our understanding of these threats remains limited [37], [48], [60].

Document macros are embedded sequences of commands within a document, similar to the instruction sets found in executable programs [44]. These macros can be weaponized to download malware, exfiltrate sensitive data, or exploit vulnerabilities in the processing software to achieve unauthorized system access. Furthermore, attackers often leverage social engineering tactics to manipulate users into interacting with malicious documents containing clickable links or attachments, or deceptive prompts encouraging users to enable macros [67].

Prior research has explored different approaches for detecting malicious documents. Yan et al. proposed DitDetector, which leverages bimodal machine learning models to combine visual and textual information for macro malware detection [69]. Cohen et al. presented a Structural Feature Extraction Methodology (SFEM) specifically targeted towards Office Open XML (OOXML) document formats, employing machine learning for malicious document identification [11]. A significant portion of document analysis research focuses on extracting and analyzing macro code. Extraction is typically achieved using tools like oletools [34], followed by training detection models. These are based on techniques like Latent Semantic Indexing (LSI) [48], Natural Language Processing (NLP) using Bag-of-Words and Term Frequency-Inverse Document Frequency (TF-IDF) [47], or identification of specific macro code keywords (e.g., AutoOpen and Shell) [29]. Beyond code analysis, recent work by Casino et al. explores the potential of detecting deceptive information within documents by constructing lightweight signatures from file components (e.g., "enable editing" and "enable content") for malware detection [8]. Ruaro et al. took a more targeted approach, focusing on symbolic execution for automated deobfuscation and analysis of Excel 4.0 macros (XL4) prevalent in Microsoft Excel files [60].

While existing research primarily focused on the binary classification of documents as either "malicious" or "benign," we argue that a comprehensive understanding of the evolving landscape of malicious documents is required for effective defense strategies. This is mainly because of two key factors: (1) *The diverse nature of file formats* (e.g., OLE and OOXML) *and macro types* (e.g., Visual Basic for Applications (VBA) macros [44] and Excel 4.0 macros [53]) presents challenges for extracting file metadata and macro code. This variety allows attackers to develop new variants utilizing different formats, effectively evading signature-based detection [70]. (2) *Attackers actively employ obfuscation techniques to evade analysis* and hinder the effectiveness of existing tools in accurately analyzing these samples [50], [51]. As a result, malware analysts are often limited by the capabilities of available tools. When these tools encounter incompatible file formats or obfuscated code, analysts resort to time-consuming and resource-intensive manual analysis.

The aforementioned challenges highlight the need for a deeper understanding of the malicious document ecosystem. Building upon prior efforts in malicious content detection we conduct a measurement study on a large dataset of recent malware samples. This study focuses on prevalent threats embedded within Microsoft Office

documents (Excel and Word) and Rich Text Format (RTF) files to inform future research efforts in this domain. We leverage state-of-the-art automated feature extraction techniques specifically designed for document-based malware analysis. Through a comprehensive evaluation, we assess our capability to parse complex document file formats, detect malicious indicators, and ultimately, identify gaps in current automated document analysis approaches.

In summary, our main contributions are as follows:

- We collect a dataset of malicious documents covering the period from January 2020 to January 2024 and encompassing 9,086 samples across various file formats (.docx, .xlsx, .doc, .xls, and .rtf).
- We present a methodology for robust file type identification, particularly crucial when dealing with a diverse set of 10+ file formats. This methodology enhances the accuracy of malicious indicator detection by ensuring proper file parsing.
- We identify the most prevalent document-based threats and pinpoint limitations in current analysis methods. We also provide valuable insights to inform the future development of robust analysis tools and defense mechanisms.

Artifacts. We believe that our findings will contribute to the advancement of knowledge in the malicious documents ecosystem by informing the development of more robust analysis mechanisms. To foster further research in this domain, we make our dataset and source code available at https://github.com/SecPriv/malwaredocumentanalysis.

2. Document File Types

Office File Format. Microsoft Office utilizes a variety of file formats for documents, each with its own capabilities. This variety can be seen within Excel file formats alone, ranging from the older binary formats (.xls, .xlsb) to the XML-based formats (.xlsx, .xlsm). Despite these differences, all Excel files share a core structure: a workbook containing one or more spreadsheets.

Microsoft introduced the Office Open XML (OOXML) format in 2006. This XML-based format, standardized as ECMA-376 [16] and later adopted as ISO/IEC 29500 [26] in 2016, has become the de facto standard for representing documents, spreadsheets, and presentations. OOXML leverages zip-compressed archives to store data. These archives contain multiple files and directories, with the Extensible Markup Language (XML) used to describe the actual document content and associated elements like images or stylesheets. OOXML supports a wide range of features, from spreadsheet formulas and form fields to integration with other XML formats like SVG or MathML. Additionally, features like digital signatures, document encryption, and macro support (in various languages like Basic, JavaScript, and Python) are possible within OOXML documents [49].

Prior to OOXML, Microsoft Office used legacy binary file formats, primarily the Compound File Binary (CFB) format, also known as OLE (Object Linking and Embedding), and the Compound Document File (CDF) format. Introduced with Office 97, this format uses file extensions like .doc (Word), .xls (Excel), and .ppt (PowerPoint) [38]. Analogous to a traditional file system, OLE files are composed of storage objects and streams. Storage objects act as containers, potentially holding additional storage objects or streams. Streams, on the other hand, represent the actual data content, such as text, images, or embedded objects within the file. Although legacy, they remain available as an alternative document-saving option for compatibility purposes.

Rich Text Format (RTF). Introduced by Microsoft in the 1980s, RTF (.rtf) provides a method for encoding formatted text and graphics for use across different applications. This format facilitates interoperability, enabling document exchange between Microsoft products and various word-processing software. Consequently, RTF files can be transferred between operating systems without compromising document formatting [18]. The widespread support of RTF by most word processors, text editors, and document viewers allows for easy sharing and distribution. Unlike the previously discussed formats, RTF files rely on unformatted text, control words, groups, backslashes, and delimiters for formatting. The control words, according to RTF specification, begin with a backslash (e.g., `\fonttbl`), with parameters enclosed in curly braces ({...}). The braces can contain multiple control words (forming a group), plain text, or even nested braces for more complex formatting.

2.1. Document-based Threats

Attackers embed macro malware in documents, such as Word, Excel and PowerPoint files, which can download additional payloads from remote servers, extract malicious code directly from the document, or steal data from the victim's machine. A targeted phishing campaign in September 2023 exemplifies this threat [17]: APT34, a suspected Iranian cyberespionage group, used a seemingly legitimate document titled "MyCv.doc" to deploy the Menorah.exe malware. This document contained hidden macros that downloaded and dropped a .NET malware executable. Phishing emails are a common method for attackers to deliver these malicious documents [42]. However, with increased security awareness and Microsoft's default macro-disabling policies, attackers now resort to deceptive social engineering techniques [67]. This involves using misleading images or text within the document to trick users into enabling editing or running macros. The prevalence of document-based threats is further substantiated by Botacin et al. [6], who observed a significant increase in CDFs used in regionalized and targeted malware attacks. Below we discuss the specifics of macros and their utility in different file formats.

Macros. Macros, essentially embedded code within office documents, can create custom functions that automate specific actions when the document is opened. Visual Basic for Applications (VBA) macros are code snippets written in a variant of Visual Basic, specifically designed for scripting within Microsoft Office applications like Word, Excel, and PowerPoint [44]. Listing 1 shows an example of a VBA macro that extracts malicious code from the document's properties, decodes it, and executes it based on the victim's operating system.

Listing 1. Example of a malicious VBA macro payload: The macro runs automatically when the document is opened, extracts and decodes the "Comments" property, writes the content to a temporary file and executes it via the shell in a platform-specific subroutine for Windows or macOS.

```
Attribute VB_Name = "ThisDocument"
Attribute VB_Base = "1Normal.ThisDocument"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = True
Attribute VB_TemplateDerived = True
Attribute VB_Customizable = True

Sub AutoOpen()
    On Error Resume Next
    Dim found_value As String

    For Each prop In ActiveDocument.BuiltInDocumentProperties
        If prop.Name = "Comments" Then
            found_value = Mid(prop.Value, 56)
            orig_val = Base64Decode(found_value)
            #If Mac Then
                ExecuteForOSX(orig_val)
            #Else
                ExecuteForWindows(orig_val)
            #End If
            Exit For
        End If
    Next
End Sub


Sub ExecuteForWindows(decodedString As String)
    Dim tempFilePath As String, fileNum As Integer
    tempFilePath = Environ("TEMP") & "\tempfile.bat"
    fileNum = FreeFile
    Open tempFilePath For Output As #fileNum
    Print #fileNum, decodedString
    Close #fileNum
    Shell "cmd.exe /c " & tempFilePath, vbHide
End Sub


Sub ExecuteForOSX(decodedString As String)
    ...
End Sub
```

VBA macros reside within a VBA project structure, but their location depends on the document type and file format. In Legacy Binary Formats (1997–2003) VBA projects reside within an OLE storage named "Macros" at the root of the OLE file for Word documents, while Excel stores them in storage called "_VBA_PROJECT_CUR." PowerPoint integrates macros directly into the binary presentation structure, and not in a dedicated storage [32].

The introduction of OOXML formats (2007+), such as .docx, .xlsx, and .pptx, changed how VBA macros are handled. These formats cannot store macros directly by default due to security considerations. Only specific files with enabled macros, denoted by the 'm' at the extension end (e.g., .dotm, .docm, .xlsm, .pptm), can contain them. When enabled, macros are stored in a separate binary OLE file named "vbaProject.bin" within the zip archive structure that forms the core of OOXML formats. This file maintains the same VBA project structure as legacy formats for consistency. However, the location of "vbaProject.bin" within the zip archive varies depending on the document type. For example, in Word documents, it is located at "word/vbaProject.bin", while in Excel 2007 and later it is found at "xl/vbaProject.bin". Similarly, PowerPoint stores it at "ppt/vbaProject.bin". While "vbaProject.bin" is the standard name for the VBA macro storage file in OOXML formats, the standard itself allows for some flexibility. Developers can use custom file names for the macro storage, as long as the relationships are correctly defined within the XML files.

Similar to OOXML files, RTF specifications do not allow embedding VBA macros directly within an RTF file [21], however, there is an exception. RTFs can embed OLE objects with potentially malicious content. Specifically, OLE "Package" objects can store any file type, including executables and scripts. If a user double-clicks such an embedded object, the system will launch the file [31]. For instance, security researchers have documented cases where malicious actors included macro-enabled Excel sheets within RTF documents and tricked users into executing payloads [3].

The extraction and analysis of malicious content depends on understanding the file formats and identifying the embedded macro location. Olevba [32] can parse Office file formats and supports both OLE and OpenXML, detecting VBA macros and extracting their source code. Similarly, extracting embedded objects from RTF files requires parsing their nested control words. Tools like rtfobj from oletool [34] and RTFScan from OfficeMalScanner [5], along with antivirus engines, rely on this parsing capability to identify potential threats within RTF files.

Advanced Techniques. VBA macros have been a popular choice for malicious actors [44], but attackers are constantly exploring advanced techniques like XL4 macros [53] and remote template injection vulnerabilities [57], [58]. Microsoft Excel 4.0 introduced XL4 macros in 1992, a tool for automation through dedicated macro worksheets. This concept differed significantly from VBA macros, introduced one year later in Excel 5.0. Unlike VBA macros, XL4 macros cannot reside within regular worksheets. Instead, they are confined to specific Excel 4.0 macro sheets, where each cell holds a single formula defining the macro's action. In newer file formats, XL4 macros are stored in separate XML files [53]. These macros are often hidden within apparently legitimate documents and under several layers of obfuscation making them difficult to analyze [60].

Beyond hiding and obfuscating macros in documents, attackers leverage other file formats to deliver malware. RTF's object-linking capabilities make it an attractive target for attackers who exploit the format's control words for malicious purposes. By manipulating the `\template` control word, attackers can reference a URL on a malicious server instead of a legitimate template file. This bypasses the intended functionality and directly retrieves the malicious payload upon opening the file [57], [68]. Furthermore, RTF's `\object` control words, with associated parameters and data, allow attackers to embed executable files like PE, VBS, and JS directly within the document [62], [70]. These embedded objects can then download and execute code, effectively transforming the RTF file into a downloader and launcher for malicious content. Identifying such linked objects requires examining the RTF document's specific control words, particularly `\template` and `\object`.

3. Automating Malicious Document Analysis

In this work, we analyze malicious documents used in a wide range of attacks, from common threats (e.g., Qakbot and Emotet [10]) to targeted campaigns by nation-state actors (e.g., Gamaredon and Lazarus [42]). We employ a comprehensive approach that involves building an extensive dataset and meticulously evaluating the effectiveness of state-of-the-art methods in extracting malicious content while identifying their limitations.

We automate the analysis of malicious documents by using a combination of features extracted through static analysis. This includes identifying macro functionalities, extracting relevant textual information, and detecting malicious techniques and code patterns using Yara signatures. We focus on widely used binary file formats (.doc, .xls, .ppt), OOXML formats (.docx, .xlsx, .pptx), and RTF (.rtf) documents based on their ubiquity, the prevalence of recent vulnerabilities discovered in them [12], and their frequent utilization in targeted attacks [63].

3.1. Dataset

Understanding and mitigating the threats posed by malicious documents requires analyzing a dataset of diverse and representative samples. We achieve this by leveraging VirusTotal academic API [2], allowing us to collect a large dataset of malicious files focusing on specific file types. Using VirusTotal has become a standard way and is part of the best practices for doing malware research [9]. We start by identifying malicious documents submitted to VirusTotal between January 2020 and January 2024 using the 'tag: documents' filter and focusing on files classified as malicious by at least 20 antivirus engines. This initial search yielded a vast dataset of 97,288 unique files (identified by SHA256 hashes).

To create a more representative sample size, we implemented a two-step process. We used Microsoft Defender's Threat Labelling Engine to analyze file metadata, where Microsoft Defender consistently provided the most reliable labels, as confirmed by a previous study [60]. This helped us identify prominent malicious tags, threat categories, and malware families. This analysis revealed a significant presence of specific malware families, including Emotet (49,119 files) and Laroux (8,002 files). To avoid the overrepresentation of dominant malware families like Emotet, we employed a random sampling approach. We selected samples from the top 15 identified families, ensuring a diverse representation within the final dataset. Table 1 details the distribution of samples across these families. By using random sampling we narrowed down the initial dataset to 9,086 files on which we perform our experiments. However, to facilitate further research and benefit the community, we are releasing the complete dataset consisting of 97,288 SHA256 hashes.

3.2. Malicious File Analysis

The exploration of the malicious document ecosystem remains largely understudied. This is partly due to the variety and complexity of Microsoft's document formats, characterized by intricate specifications [38]. Malware authors leverage these complexities, crafting documents that exploit vulnerabilities in static parsers while maintaining compatibility with parsers within Office applications [62].

To understand the prevalence of different document formats within our dataset, we employed a systematic categorization based on each file's Multipurpose Internet Mail Extensions (MIME) type. A standardized identifier, the MIME type specifies a file's format and key characteristics. Essentially, it acts as a digital fingerprint attached to a file, conveying crucial information to systems about its content. As discussed in Section 2, this information is useful for the proper handling, interpretation, and processing of a file. Table 2 details the over 20 MIME types and file formats identified in our dataset. A significant portion of the files (about 87%) fall into Word or Excel formats, including both OOXML and pre-OOXML versions.

TABLE 1. MALICIOUS DOCUMENTS ACROSS THE TOP 15 FAMILIES.

Malware Family	# of Samples	Percentage
Emotet	1,335	14.69%
Thus	831	9.14%
Laroux	654	7.20%
Obfuse	431	4.74%
EncDoc	423	4.66%
Mailcab	284	3.12%
Donoff	240	2.64%
Sadoca	236	2.60%
Marker	199	2.19%
LionWolf	163	1.79%
Woreflint	120	1.32%
Madeba	104	1.14%
Hancitor	93	1.02%
Xaler	91	1.00%
Leonem	74	0.81%
Total Dataset	9,086	100%

Filetype Identification. Our analysis revealed limitations in current approaches in identifying the correct file MIME type. To investigate the prevalence of file misclassification within our dataset, we employed three distinct tools: Python-magic [24], ExifTool [23], and Magika [20]. Python-magic is a widely used library that leverages libmagic, the ubiquitous library behind the file command, to identify file types using a predefined database of magic numbers and header patterns. ExifTool, though primarily focused on metadata extraction, can also identify file types based on header information in various formats. Finally, Magika, an open-source library developed by Google, utilizes a deep learning model to perform file type identification, including complex or less common formats.

Table 2 shows significant discrepancies among the three MIME identification tools used in our analysis. Notably, Magika classified 521 files (5.73%) as "application/ms.outlook" (indicating Microsoft Outlook messages), while libmagic assigned the generic category "application/CDFv2" to all of them. "Application/CDFv2" typically references Compound Document Format Version 2, however, it serves as a container format and lacks the granularity to differentiate between specific document types like Word, Excel, or PowerPoint. Further, libmagic assigned the "application/msword" label to 264 files (8.38%), while Magika categorized them as "application/vnd.ms-powerpoint", "application/vnd.ms-excel", or even "application/x-msi" (Microsoft Installer) for two files. Interestingly, Magika identified a considerably higher number of MSI files (106, or 1.16%) compared to ExifTool and libmagic, which only identified one. The presence of these MSI files within the dataset warrants further investigation as part of future work. Libmagic also exhibited limitations in identifying RTF files, misclassifying 37 (14.74%) as "text/plain" or "text/octet-stream" (generic binary data). ExifTool presented further inconsistencies. It failed to identify the MIME type for 71 (0.78%) files and assigned two unexpected types: "application/vnd.fpx" (flashpix image) for 1,240 (13.67%) files and "Microsoft

TABLE 2. LIST OF MIME TYPES IDENTIFIED WITHIN OUR DATASET ALONG WITH THE NUMBER OF SAMPLES DETECTED FOR EACH MIME TYPE.

MIME Type	File Type	Magika	Libmagic	ExifTool
application/vnd.ms-excel	Excel Spreadsheet	3,100	2,812	2,602
application/msword	Word Document	2,884	3,148	2,937
application/vnd.openxmlformats-officedocument.wordprocessingml.document	Word OOXML	1,156	1,076	1,630
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	Excel OOXML	841	779	137
application/vnd.ms-outlook	Outlook Message	521	-	-
text/rtf	Rich Text Format	251	214	213
application/vnd.ms-powerpoint	PowerPoint Presentation	188	9	6
application/x-msi	Microsoft Installer	106	1	-
application/x-iso9660-image	ISO 9660 CD-ROM	9	-	-
text/plain	Plain Text Document	6	12	13
application/vnd.openxmlformats-officedocument.presentationml.presentation	PowerPoint OOXML	6	4	2
application/chm	Windows HtmlHelp Data	4	-	-
text/vbscript	Visual Basic Source	3	-	-
inode/x-empty	Empty File	3	3	-
audio/mpeg	mp3 Media File	3	-	-
application/x-java-applet	Java Archive Data (JAR)	3	7	-
application/zip	ZIP Archive	1	52	8
application/x-bytecode.python	Python Compiled Bytecode	1	-	-
application/CDFv2	Compound Document Format Version 2	-	663	-
application/octet-stream	Binary Data	-	147	-
application/encrypted	Encrypted Data	-	100	-
application/x-hwp	Hangul Word Processor Document	-	66	-
image/vnd.fpx	Media Type Flashpix	-	-	1,240
application/vnd.ms-word.template.macroEnabledTemplate	Microsoft Word Macro-enabled Template	-	-	226
unknown	N/A	-	-	71

TABLE 3. PAIRWISE AGREEMENT IN IDENTIFYING MIME TYPES.

	Magika	Libmagic	ExifTool
Magika	1.000000	0.825336	0.698657
Libmagic	0.825336	1.000000	0.742681
ExifTool	0.698657	0.742681	1.000000

word macro-enabled template" for 226 (2.48%) files. This analysis highlights the limitations of current approaches in handling the diverse document formats.

To further assess the degree of inconsistency between different approaches, we calculated the pairwise agreement rates in identifying MIME types for the 9,086 samples within our dataset. Table 3 presents the computed agreement rates between libmagic, ExifTool, and Magika. As our results show, the agreement rates vary across tool pairs. Magika and libmagic exhibit the highest agreement of 82.53%. Conversely, the agreement between Magika and ExifTool is the lowest at 69.86%. These discrepancies suggest potential for malicious actors to disguise file formats to evade detection and show the limitations of current approaches in handling less common file formats.

Our Solution: Majority Voting. Effective extraction of malicious content relies on accurate file type identification. As established in Section 2, distinct file formats possess unique structures and characteristics, and misidentifying the file type can lead to errors and missed malicious indicators. To overcome this challenge, we employ a multi-step voting mechanism to ensure the most accurate file type identification and increase the likelihood of extracting malicious content. Initially, we strive for a unanimous decision among libmagic, ExifTool, and Magika. If all three agree, we adopt their consensus. In the case of disagreement, a majority vote (two out of three tools) determines the file type. However, when the tools disagree or identify generic formats (e.g., CDF2), we use Magika's "best guess" indicator. This functionality utilizes the deep learning model to provide its most likely type based on previous encounters and confidence levels. To demonstrate the efficiency of our approach, we ran our malicious content extractor (described below) without the file type detection. Our approach processed and extracted indicators from 13.34% more files (99.80% vs. 86.46%). This highlights the importance of using a multi-faceted approach for analyzing malicious documents, as dependence on a single approach can introduce limitations.

3.3. Malicious Content Extraction

Prior research has explored various aspects of document-based malware, including analyzing malicious VBA macros [47], [48], or analyzing embedded images and textual information within documents [69]. Additionally, research efforts have addressed the challenge of complex macro code by developing techniques for automated extraction of Indicators of Compromise (IoCs) through deobfuscation [28], [60]. Building upon prior research, our content analysis pipeline utilizes oletools [34], a well-established suite within the malware forensics community for document parsing and macro extraction.

To understand the social engineering tactics embedded within documents, we leverage the Python library textract [41]. This library extracts text content from a wide range of document formats. However, we encounter extraction errors with certain file types, particularly newer OOXML documents containing a large number of pages and complex layouts. For these file types, we employ a dedicated extraction pipeline using the python-docx module [7], ensuring comprehensive text extraction.

Finally, we employ Yara, a rule-based tool that matches text phrases, code snippets, and unique patterns within a file to identify potential threats. We leverage a curated set of Yara rules from reputable sources maintained by active developers, including Inquest Labs [25], Florian Roth [59], and Ditekshen [14].

TABLE 4. FREQUENCY OF EXTRACTED FEATURES.

Stage of Feature Extraction Pipeline	# of Samples
Processed Files	9,068 (99.80%)
Extracted Macros	6,944 (76.42%)
Extracted Text Content	3,414 (37.57%)
Yara Rules Triggered	8,934 (98.32%)

TABLE 5. FREQUENCY OF TRIGGERED YARA RULES.

Triggered Yara Rule [14], [25], [59]	# of Samples
Office_Document_with_VBA_Project	7,860 (87.98%)
Microsoft_Excel_Hidden_Macrosheet	3,234 (36.20%)
Windows_API_Function	2,932 (32.82%)
Microsoft_Excel_with_Macrosheet	1,838 (20.57%)
Suspicious_PowerShell_WebDownload	679 (7.60%)
Powershell_Command_Fileless_Malware	654 (7.32%)
SUSP_Excel4Macro_AutoOpen	490 (5.48%)
Base64_Encoded_URL	368 (4.12%)
Office_AutoOpen_Macro	365 (4.09%)
PowerShell_in_Word_Doc	343 (3.84%)

For streamlined analysis, we perform asynchronous scanning of malicious documents. We use 7zip [1], a widely used archive utility, to extract all storage and streams from the OLE compound files and OOXML zip containers. This process creates a folder and an individual file for each successfully extracted storage and stream. Subsequently, we apply the Yara rules to both the original and extracted content. Our Yara analysis pipeline outputs detailed information for each processed file, including the number of extracted files, file matches, matched strings or byte patterns, and the corresponding Yara rule.

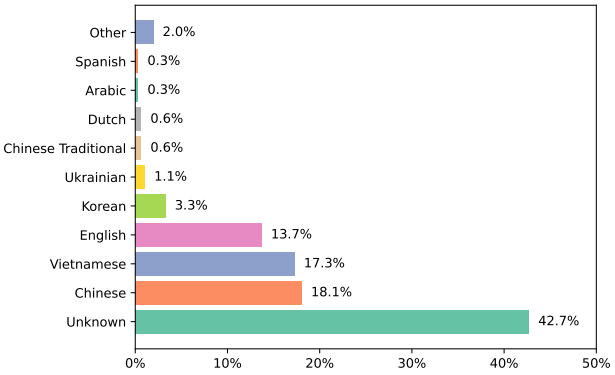
Table 4 details the success rates for extracting various features from our dataset of 9,086 files. We were able to extract macro content from 6,944 samples (76.42%) and textual information from 3,414 samples (37.57%). Finally, 8,934 files (98.44%) triggered at least one Yara detection rule. Moreover, Table 5 details the frequency of the different Yara rules identified in our dataset. "Office Documents with VBA Projects" are the most frequently triggered rule (87.98%), followed by "Microsoft Excel Hidden Macrosheets" detected in 36.20% of samples.

It is worth noting that our processing pipeline encountered errors during the analysis of 18 files (0.2%), resulting in a processed file count of 9,068 files as shown in Table 4. These errors stemmed from two main file types. First, thirteen files were identified as Microsoft Word documents with obfuscated tags in VirusTotal. This obfuscation might render them unprocessable for further analysis. The five other files were RTFs, which we identified to be associated with the Royal Road Hacking Tool used by the Chinese APT group APT29 (also known as Royal Road) [15]. This sophisticated malware exploits previously unknown vulnerabilities to create corrupted RTF documents that trick victims into executing malicious code. Notably, this malware was used in a high-profile attack against the Mongolian Ministry of Foreign Affairs, highlighting its potential for significant disruption [15].

4. Malicious Document Characteristics

Macros. Our macro extraction pipeline was able to extract macros from 6,944 samples (76.42%), meaning 2,124 files had no detectable macros. However, further analysis revealed 708 files (33.33%), where our pipeline identified the presence of macros but failed to extract the clear-text macro code. Supporting this, Table 5 shows the Yara rule for VBA projects in Office documents (macro-enabled documents) triggered in 87.98% of the files analyzed. We hypothesize that these unextracted macro codes could be corrupted or heavily obfuscated, requiring manual analysis or advanced techniques for extraction. Another possibility is that they might be embedded unconventionally and located within non-standard parts of the document.

Figure 1. Distribution of languages within the malicious documents.



To further analyze the threats within macro-enabled documents we investigate the combination of triggered Yara rules. The most prevalent and insightful combination is documents flagged as "Office Document with VBA Project" appearing alongside "Microsoft Excel Hidden Macrosheet" (374 occurrences). This suggests the utilization of *hidden* macrosheets within Excel documents (described in Section 4.1), likely to bypass detection. Another common combination occurring in 344 samples is "Microsoft Excel with Macrosheet" with "Office Document with VBA Project," highlighting the general risk associated with Excel macrosheets. Finally, we observed another common Yara combination of "Office Document with VBA Project" with PowerShell commands and fileless malware (e.g., "Powershell Command Fileless Malware"). This suggests the sophisticated use of macro-enabled documents as a delivery mechanism for fileless malware attacks. We further investigated the signature associated with this technique and identified the use of several embedded Windows API calls. PowerShell, a scripting language, can utilize Windows API functionalities to perform a variety of malicious actions, such as downloading additional malware, manipulating files, or executing commands.

Deceptive Text. We further investigate the textual content of socially engineered documents within our dataset by analyzing the language distribution. Figure 1 shows the frequency of the most common languages, such as Chinese, Vietnamese, and English, likely reflecting international campaigns. Interestingly, a significant portion of documents fell into the 'unknown' category for two key reasons. First, some documents triggered low-confidence language accuracy, requiring further analysis. Second, a manual examination of a sample set revealed documents containing scripts or macro commands embedded within the textual content. Additionally, examining the

relationship between text and macros, we found that 1,468 (42.97%) of the 3,414 files containing text lacked any evidence of macro code, suggesting not all malicious documents with textual content rely on macros for malicious behavior. However, the majority (70.98%) of these macroless files were Excel documents, where further analysis revealed the presence of XL4 macros embedded within the cell content of the spreadsheets (discussed in Section 4.1).

No Indicators. We identified 46 DOCX, 30 XLSX, and 12 OneNote documents with no immediate malicious indicators such as macros or text content. OneNote files (.one) present a unique challenge for analysis. While they represent the native format for Microsoft OneNote, a digital note-taking application, current parsing tools struggle to handle them effectively and Magika misclassified nine OneNote files as ISOs and three as MP3s. This is particularly concerning as recent research by Proofpoint indicates the TA577 cybercrime group is increasingly using OneNote documents for malware delivery, potentially paving the way for similar tactics by others [40]. This group has also been linked with high confidence to a March 2021 Sodinokibi ransomware attack that initially compromised victims via malicious Microsoft Office attachments containing macros that downloaded and executed IcedID malware [35]. While parsing OneNote documents presented challenges, we further identified 28 out of 30 XLSX files that lacked any malicious indicators to be linked with the Bluenoroff cryptocurrency campaign associated with North Korean threat actors [55]. By comparing the file hashes to those identified in public reports, we discovered a repetitive exploitation pattern in these XLSX files. All of them relied on the well-known CVE-2017-0199. The exploit involves fetching malicious content from a URL embedded in the document's metadata and then downloading a remote macro-enabled template for further execution. We discuss this technique in detail in Section 4.2. These findings suggest both the prevalence of Excel files as a delivery method in cyberattacks and the use of unconventional exploitation techniques involving Excel and OneNote files by more advanced attackers.

4.1. Evasion Techniques

When delving deeper into the challenges associated with extracting and analyzing malicious content we focus on Excel files: Microsoft Excel allows hiding sheets within a workbook, although one sheet must remain visible [45]. While hidden sheets can be unhidden through the interface or file manipulation, another state marked as "very hidden" requires hexadecimal editing [45]. Malware authors exploit this feature to embed malicious macros in hidden sheets, making them harder to detect (e.g., XL4 macros) [50], [51]. While both VBA and XL4 macros (discussed in Section 2) can be exploited maliciously, XL4 macros introduce unique complexities for analysis due to their intricate structure and dispersed execution logic [27].

In our dataset, we identify 300 occurrences of "Microsoft Excel Hidden Macrosheet" and "Microsoft Excel with Macrosheet," 270 occurrences of "Microsoft Excel Hidden Macrosheet," "Microsoft Excel with Macrosheet" and, "SUSP Excel4Macro AutoOpen" Yara rule combinations. Both entries highlight the prevalence of malicious macros hidden within Excel documents. The presence of the "SUSP Excel4Macro AutoOpen" rule in the second entry suggests an additional layer of concern. These documents not only contain hidden macros but also have macros specifically designed to run automatically upon opening the file. This increases the risk of immediate execution of malicious code if the user opens the document.

We further identified 708 files where our extraction pipeline detected the presence of macros but failed to extract the clear-text macro code. Interestingly, of the 708 unextracted macros, 625 (88.27%) originated from Excel files containing XL4 macros. This highlights the versatility of XL4 macros and the challenges in extracting the macro code for malicious content analysis. In response to these challenges, Ruaro et al. [60] designed Symbexcel, a symbolic execution engine specifically for analyzing and deobfuscating XL4 macros in Excel 4.0. This technique aims to infer the "correct" values of any environment variables, leading to the deobfuscation of the malicious payload hidden within the macro.

We used Symbexcel to analyze 625 Excel files identified as containing XL4 macros. Symbexcel successfully extracted IoCs from the macro logic within 458 files (73.28%). However, it encountered limitations in processing of 154 files. One of the primary limitations is the lack of complete grammar that more accurately describes the Excel 4.0 Grammar. In several parsing scenarios, the grammar parser failed to evaluate formulas, such as `ShrFmla` due to them deviating from their expected behavior. Additionally, certain non-standard function calls like `_xlfn.CONCAT` instead of `CONCAT()` also contributed to parsing errors. Parsing XL4 formulas correctly is crucial for analyzing these malicious documents. While it might initially appear straightforward, the syntactical features of Excel formulas are quite complex. These findings suggest the need for a more precise formula grammar, ideally one that completely matches the one implemented in Excel, to effectively handle complex Excel 4.0 malware.

4.2. Other Advanced Techniques

RTF exploits. Rich Text Format (RTF) files offer great versatility in document formatting but present significant challenges for robust security analysis. RTF heavily relies on control words to define document presentation (see Section 2). These control words, along with their associated parameters and data, can introduce vulnerabilities when parsing errors occur [62]. Attackers exploit these errors to embed malicious resources within the RTF file. Historical vulnerabilities such as CVE-2010-3333 and CVE-2014-1761 demonstrate the potential dangers of flawed RTF parsing implementations [70].

Our analysis confirmed the complexities involved in detecting threats within RTF documents. Apart from the ability of five malicious RTF files to evade parsing altogether (see Section 3.3), we were unable to extract malicious indicators within a substantial portion (60.55%) of the RTF files. For these files, we encountered "malformed OLE object" errors during analysis. These errors prevented the extraction of OLE objects, which could potentially contain hidden malicious content.

Our analysis also identified a subset of RTF files (4.71%) that caused errors due to malformed headers.

According to Microsoft's specifications, a valid RTF document should begin with `{rtf1}` (RTF version 1.x). However, manual inspection revealed deviations in these files, where the header started with `{rt0}` or simply `{rt}`. While Microsoft Word can handle these variations, many analysis tools misinterpret them as plain text and fail to process them as RTF documents. Interestingly, all the files with malformed headers were linked to a Remcos RAT (Remote Access Trojan) malware campaign. Remcos, also known as Remote Control and Surveillance Software, is a sophisticated tool that grants attackers full control over compromised machines. In this campaign, the attackers used an RTF document with a malformed header and an Equation object to spread a variant of Remcos [71].

Despite these limitations, our Yara rule-based detection provided valuable insights. We identified 24 unique rules triggered across 256 RTF files. Table 6 details the broad classification of these rules and the corresponding number of detections for each category. The most prevalent threat among the RTF files is exploitation through specific CVEs, particularly CVE-2017-1182, CVE-2017-8759, and CVE-2018-0802 [12]. CVE-2017-1182 exploits a buffer overflow vulnerability within the Microsoft Equation Editor allowing attackers to execute malicious code on opening a specially crafted RTF document. CVE-2017-8759 targets a SOAP WSDL parser code injection vulnerability within Microsoft Office RTF documents allowing attackers to inject arbitrary code during parsing. Finally, CVE-2018-0802 represents a more general memory corruption vulnerability within Microsoft Office applications.

Beyond the exploitation of known vulnerabilities, our analysis revealed two additional frequently occurring indicators. The first involved a combination of embedded malicious objects within the RTF files and deviations from the standard RTF specifications. The second indicator focused on anti-analysis techniques through RTF header manipulation, header obfuscation, and embedding obfuscated OLE object headers.

External Relations and Template Injection. Standard detection tools often focus on malicious code embedded directly within documents. However, Microsoft Office allows documents to reference various resources, including external or remote templates. Attackers exploit this functionality by modifying the document's properties and utilizing *external relations*. These modifications point the document to a malicious template hosted on a remote server, such as a URL or a GitHub repository. Once the compromised document is opened, the malicious code from the external template is loaded and executed. Recent threat reports detail the external relations exploit where attackers crafted DOCX files to trigger macro execution from a remote .dotm template [19], [58]. To assess the prevalence of this technique within our dataset, we examined document relationships within the .rels folder, particularly the `settings.xml.rels` file. We identified 224 (2.47%) samples exhibiting signs of utilizing external relations to potentially download remote malicious content. Furthermore, Yara rules successfully flagged a majority of these samples, triggering indicators like "OLE RemoteTemplate" or "XML WebRelFrame RemoteTemplate." However, three files (less than 1.5%) bypassed detection. These outliers included two XLSX files using hyperlinks to malicious URLs and one PPTX file containing a link to a malicious OLE object. Notably, all three files had text content, suggesting a potential blend of social engineering and malicious links. Interestingly, these three samples were linked to targeted APT attacks, with one associated with a Chinese group known for targeting the Tibetan community [56].

TABLE 6. CLASSIFICATION OF THREATS AMONG RTF DOCUMENTS.

Threat Category	Count
Exploitation of CVEs (2017-1182, 2017-8759, 2018-0802)	264
Malicious Embedded Objects	253
RTF Header Manipulation	166
RoyalRoad Exploits	25

While remote template injection poses a significant threat within OOXML formats, attackers have also exploited similar vulnerabilities in RTF documents. A recent report by Proofpoint details a technique known as RTF template injection [57]. RTF files store formatting instructions as plain text, and attackers manipulate the *template property* to redirect the file to execute a malicious script. Proofpoint observed a rise in this technique, primarily employed by APT groups linked to India and China. The simplicity and effectiveness of RTF template injection suggest its potential for wider adoption by various cyber criminals. The trickle-down effect in cyberattacks further amplifies this concern [61]. As seen with the Royal Road tool, initially used by APT groups, the techniques become more widespread over time [15].

Dynamic Data Exchange (DDE). DDE is a protocol originally designed for data sharing between Microsoft Office applications. DDE was partially superseded by Object Linking and Embedding (OLE) and is currently maintained in Windows systems for backward compatibility [66]. However, attackers exploit DDE to execute malicious commands, including downloading additional payloads. This technique works in both OLE and OOXML file formats. While newer Office versions alert users about DDE commands within documents, attackers have adapted their phishing tactics to bypass these warnings. This method is commonly used by threat actors like APT28 and FIN7 [54]. One method of bypassing warnings involves manipulating the DDE syntax to craft obfuscated prompts. Attackers can achieve this by directly invoking PowerShell through modified parameters, resulting in prompts that appear less suspicious to users. This technique increases the likelihood of user interaction and subsequent compromise [13]. Basic DDE detection can be achieved through string scanning, which involves manually searching the text content of a file for keywords like `DDEAUTO` or `DDE`, although it can be time-consuming and can miss obfuscated instances. Another approach is using Yara rules to identify keywords and specific features associated with DDE exploitation. Our malicious content extraction pipeline, combining Yara and msodde [33] from oletools, detected 240 (2.64%) files using DDE techniques.

5. Threats to Validity

Our study offers valuable insights into the use of malicious documents, however, we acknowledge limitations in the dataset that might affect the generalizability

of our findings. Our focus on malware delivered solely via malicious Office documents detected by VirusTotal introduces a potential bias. Limiting the scope to a specific attack vector excludes other methods employed by attackers, such as drive-by downloads, malicious PDFs, and weaponized email messages. This limitation restricts our understanding of the complete threat landscape. Moreover, VirusTotal relies on various antivirus engines and threat intelligence feeds to identify malware. While a valuable source, some malicious samples might evade detection by some engines, leading to the underrepresentation of certain types of malware in the dataset. This can skew the results towards malware types that are more easily identified by VirusTotal.

Despite these limitations, our findings can still be considered a valuable lower bound for the overall threat landscape. The high prevalence of malicious documents within our dataset suggests its significance as an attack vector. Furthermore, our insights can inform the development of mitigation strategies specifically for document-based threats.

6. Recommendations and Future Directions

Automating malicious document analysis remains critical for both post-mortem incident response and proactive threat prevention. In post-mortem analysis, understanding malicious actions performed is essential, while identifying IoCs helps track future attacks. However, current solutions face several limitations:

File Identification. Analyzing malicious Microsoft Office files is challenging due to the complexity and variety of file formats, each with its own potential for exploitation. Unfortunately, current methods often misidentify files, especially those that are incompatible or deliberately disguised. This leads to missed threat indicators and necessitates time-consuming manual analysis. However, our findings in Section 3.2 demonstrate that using a combination of multiple tools can substantially improve the identification process and reduce processing errors.

Macros and Obfuscation. Macro malware in documents continues to remain a prevalent threat, as detailed in Section 4, where 76.42% files in our dataset contained macros and the majority of them are Excel and Word documents. However, Excel files pose the most significant challenge. Their layered evasion and obfuscation techniques make content analysis difficult. Existing methodologies often struggle with heavily obfuscated macro code and incomplete grammar, as discussed in Section 4.1. To effectively combat complex Excel 4.0 malware, developing a comprehensive Excel 4.0 formula grammar and parser is essential. Here, Large Language Models (LLMs) offer a promising solution. LLMs trained on large datasets of real-world Excel formulas, encompassing both standard and non-standard functions, could potentially learn to recognize and interpret functions beyond those explicitly defined in the current grammar. Furthermore, as Excel evolves with new features and functions, the LLM's ability to learn continuously from new data would enable it to adapt and handle these changes automatically, reducing the need for constant grammar updates.

Unconventional Attack Vectors. RTF files have become a prominent attack vector. These techniques include embedding malicious resources directly within the document, exploiting remote template injection vulnerabilities, or leveraging parsing errors. As shown in Section 4.2, current static analysis approaches fail to extract indicators from a substantial portion of 60.55% files. Notably, in our analysis, we identified instances of sophisticated APT actors successfully using RTFs as attack vectors. Furthermore, OneNote file formats present additional parsing challenges. While Yara rules, with their predefined patterns and known threat signatures, offer some defense against known threats, they struggle with entirely novel or obfuscated malware. This creates a reactive approach, hindering our ability to proactively understand the evolving threat landscape and adapt to new file types and attack methods. The development of advanced static analysis tools that can effectively detect malicious document structures, reliably handle different file formats (RTFs, OneNote), and extract embedded malicious content, would be a step towards proactively aiding in threat detection and analysis.

Future Directions. As an immediate improvement to the current static analysis methods we aim to enhance RTF file analysis techniques by focusing on two key areas: (1) identification and extraction of control words commonly used for embedding or obfuscating objects (e.g., `\objdata`, `\objemb`, `\template`), and (2) anomaly detection through analysis of control word order, combinations, and nesting patterns within the RTF structure to pinpoint unusual behavior. Additionally, we will investigate embedded OLE objects within identified control word groups, searching for non-standard characters and byte sequences that might indicate the presence of potentially malicious code or obfuscation. We further plan to investigate the social engineering tactics employed within the documents. Here, our initial focus is on incorporating an image extraction component to capture deceptive images. We will leverage OCR tools capable of processing images across various document formats, to effectively analyze the content of embedded images. Furthermore, we plan to explore the translation of non-English content to English for a more comprehensive analysis of the types and variations of prompts used to lure users into opening or clicking on malicious documents. Finally, we will explore methods for identifying, parsing, and extracting critical file metadata and potentially malicious content from Microsoft OneNote documents.

7. Related Work

Malware analysis and automated malware detection techniques have long been a well-established field. While significant research has focused on executable files, malicious document files remain a relatively understudied area. Within the document space, extensive research exists on malicious PDF detection [36], [39], [64], [65], [72]. However, these approaches often rely on format-specific features and do not generalize to other file formats like Microsoft Office documents due to structural differences.

Several studies have explored document analysis through file structure examination. Otsubo et al. [52] investigated deviations from standard file formats in documents containing executables. Cohen et al. [11] extended

this concept to XML-based Office documents, employing machine learning on extracted structural features for malicious document detection.

Other studies have explored VBA macro analysis for malicious document detection. Bearden et al. [4] classified macros using K-Nearest Neighbors on features extracted from p-code opcodes (translated VBA code) with TF-IDF weighting. Mimura et al. [46], [48] investigated raw VBA code with Doc2vec models and LSI for malicious macro detection. Kim et al. [28] proposed a machine learning approach for detecting obfuscated VBA macros, categorizing obfuscation techniques based on prior research. More recently, Casino et al. [8] proposed perceptual hashing of document images by extracting key document components for lightweight signature creation. Yan et al. proposed DitDetector [69], a bimodal learning approach using visual and textual information from document previews for macro malware detection.

In a more targeted line of research, Ruaro et al. [60] proposed Symbexcel, a symbolic execution approach for automated deobfuscation and analysis of Excel 4.0 macros (XL4), a growing attack vector within documents. Complementing this line of research, Blonde et al. [37] performed a measurement study on targeted attack documents of 3,815 samples identifies attacker focus. The authors identified several attacks targeting specific regions and ethnicities, highlighting the trend of socially engineered malware. They also found a focus on exploiting known vulnerabilities, indicating attackers prioritize readily available attack vectors.

Extending on prior research, we conducted a measurement study of malicious document samples from the past four years (2020-2023). We focused on Microsoft document formats to identify both the common techniques employed by attackers and the limitations of existing approaches in detecting these threats.

8. Conclusion

Non-binary files, particularly Microsoft Office documents, are a prevalent initial infection vector. The widespread use of Office suites, coupled with their inherent complexity and persistent vulnerabilities, creates a prime target for attackers. Thus, understanding these malicious vectors is as crucial as analyzing executables. Our analysis revealed significant limitations in current document analysis approaches. In 17.47% of the analyzed files, the malicious file type could not be definitively identified, leading to a 13.34% error rate in processing. We attribute this primarily to the obfuscated file formats and the inability of toolchains to handle emerging file types. Furthermore, current approaches struggle to extract malicious content from obfuscated Excel macros (24.64%) and reliably parse and extract malicious indicators from RTF file formats (60.55%). These issues suggest that malicious documents remain a prominent attack vector. Achieving a level of maturity in document analysis comparable to executable file analysis necessitates robust frameworks for the automated processing of diverse file formats. Parsing tools capable of handling a wider range of formats and reliably extracting malicious components for forensic analysis are essential. With this study, we aim to inform and guide further research efforts in the malicious document ecosystem.

Acknowledgements

We thank the anonymous reviewers for their valuable insights for improving the paper. We would also like to thank Godwin Attigah for their assistance with data collection, Alyssa Scheer for their help with the analysis of language detection in text data, and Alexander Lazarev for their contributions in the initial phases of the project.

This material is based on research supported by the SecInt Doctoral College at TU Wien and SBA Research (SBA-K1), a COMET Center within the COMET – Competence Centers for Excellent Technologies Programme and funded by BMK, BMAW, and the federal state of Vienna. The COMET Programme is managed by FFG. This work has also been partially supported by the Recovery, Transformation, and Resilience Plan funded by the European Union (Next Generation).

References

[1] 7-Zip. https://www.7-zip.org/, 2024.

[2] VirusTotal. https://www.virustotal.com/, 2024.

[3] Ionut Arghire. Malicious RTF Persistently Asks Users to Enable Macros. https://web.archive.org/web/20240516132100/https://www.securityweek.com/malicious-rtf-persistently-asks-users-enable-macros/, 2018.

[4] Ruth Bearden and Dan Chai-Tien Lo. Automated Microsoft Office Macro Malware Detection Using Machine Learning. In *Proc. of the IEEE International Conference on Big Data (Big Data)*, 2017.

[5] Frank Boldewin. OfficeMalScanner. https://github.com/fboldewin/reconstructer.org, 2019.

[6] Marcus Botacin, Hojjat Aghakhani, Stefano Ortolani, Christopher Kruegel, Giovanni Vigna, Daniela Oliveira, Paulo Lício De Geus, and André Grégio. One Size Does Not Fit All: A Longitudinal Analysis of Brazilian Financial Malware. *ACM Transactions on Privacy and Security (TOPS)*, 2021.

[7] Steve Canny. python-docx (v1.1.0). https://pypi.org/project/python-docx/, 2023.

[8] Fran Casino, Nikolaos Totosis, Theodoros Apostolopoulos, Nikolaos Lykousas, and Constantinos Patsakis. Analysis and Correlation of Visual Evidence in Campaigns of Malicious Office Documents. *Digital Threats: Research and Practice (DTRAP)*, 2023.

[9] Fabrício Ceschin, Marcus Botacin, Albert Bifet, Bernhard Pfahringer, Luiz S. Oliveira, Heitor Murilo Gomes, and André Grégio. Machine Learning (In) Security: A Stream of Problems. *Digital Threats: Research and Practice (DTRAP)*, 2024.

[10] CISA. Emotet Malware. https://www.cisa.gov/news-events/cybersecurity-advisories/aa20-280a, 2020.

[11] Aviad Cohen, Nir Nissim, Lior Rokach, and Yuval Elovici. SFEM: Structural Feature Extraction Methodology for the Detection of Malicious Office Documents Using Machine Learning Methods. *Expert Systems with Applications*, 2016.

[12] CVEdetails. Microsoft Word Security Vulnerabilities. https://www.cvedetails.com/vulnerability-list/vendor_id-26/product_id-529/Microsoft-Word.html, 2024.

[13] Mike Czumak. Abusing Microsoft Office DDE. https://web.archive.org/web/20240515161640/https://www.securitysift.com/abusing-microsoft-office-dde/, 2017.

[14] Ditekshen. Detection Yara. https://github.com/ditekshen/detection/tree/master/yara, 2024.

[15] Docguard. Royal Road Is Still in Use! https://web.archive.org/web/20240515155837/https://www.docguard.io/royal-road-malware-rtf/, 2023.

[16] ECMA. ECMA-376 – Office Open XML File Formats. https://ecma-international.org/publications-and-standards/standards/ecma-376/, 2021.

[17] Mohamed Fahmy and Mahmoud Zohdy. APT 34 Deploys Phishing Attack With New Malware. https://web.archive.org/web/20240517103343/https://www.trendmicro.com/en_us/research/23/i/apt34-deploys-phishing-attack-with-new-malware.html, 2023.

[18] File Format Guide. Word Processing File Formats - RTF. https://docs.fileformat.com/word-processing/rtf/, 2023.

[19] Nicole Fishbein. How to Analyze Malicious Microsoft Office Files. https://intezer.com/blog/malware-analysis/analyze-malicious-microsoft-office-files/, 2023.

[20] Yanick Fratantonio, Elie Bursztein, Luca Invernizzi, Marina Zhang, Giancarlo Metitieri, Thomas Kurt, Francois Galilee, Alexandre Petit-Bianco, and Ange Albertini. Magika Content-type Scanner. https://github.com/google/magika, 2023.

[21] Lenovo Glossary. RTF Files. https://web.archive.org/web/20240516081339/https://www.lenovo.com/us/en/glossary/rtf/, 2024.

[22] Jonathan Greig. CISA Warns of Attacks Using Microsoft Word, Adobe Bugs. https://web.archive.org/web/20240519175623/https://therecord.media/microsoft-adobe-bugs-cisa-kev-list, 2023.

[23] Phil Harvey. Exiftool (v12.4). https://exiftool.org/, 2022.

[24] Adam Hupp. python-magic (v0.4.27). https://pypi.org/project/python-magic/, 2022.

[25] InQuest. yara-rules-vt. https://github.com/InQuest/yara-rules-vt, 2024.

[26] ISO. ISO/IEC 29500-1:2016 - Office Open XML File Formats. https://www.iso.org/standard/71691.html, 2016.

[27] Jamie. zloader and XLM 4.0: Making Evasion Great Again. https://web.archive.org/web/20240515190020/https://clickallthethings.wordpress.com/2020/05/13/zloader-and-xlm-4-0-making-evasion-great-again/, 2020.

[28] Sangwoo Kim, Seokmyung Hong, Jaesang Oh, and Heejo Lee. Obfuscated VBA Macro Detection Using Machine Learning. In *Proc. of the 48th International Conference on Dependable Systems and Networks (DSN)*, 2018.

[29] Vasilios Koutsokostas, Nikolaos Lykousas, Theodoros Apostolopoulos, Gabriele Orazi, Amrita Ghosal, Fran Casino, Mauro Conti, and Constantinos Patsakis. Invoice# 31415 Attached: Automated Analysis of Malicious Microsoft Office Document. *Computers & Security*, 2022.

[30] Reason Labs. A Consumer Cybersecurity Trends Report. https://reasonlabs.com/research/consumer-cybersecurity-trends-report-2023, 2023.

[31] Philippe Lagadec. Anti-Analysis Tricks in Weaponized RTF. http://decalage.info/rtf_tricks, 2016.

[32] Philippe Lagadec. Tools to Extract VBA Macro Source Code from MS Office Documents. https://www.decalage.info/en/vba_tools, 2017.

[33] Philippe Lagadec. msodde. https://github.com/decalage2/oletools/blob/master/oletools/msodde.py, 2019.

[34] Philippe Lagadec. oletools (v0.60.1). https://github.com/decalage2/oletools, 2022.

[35] Selena Larson, Daniel Blackford, and Garrett G. The First Step: Initial Access Leads to Ransomware. https://www.proofpoint.com/us/blog/threat-insight/first-step-initial-access-leads-ransomware, 2021.

[36] Pavel Laskov and Nedim Šrndić. Static Detection of Malicious JavaScript-bearing PDF Documents. In *Proc. of the 27th Annual Computer Security Applications Conference (ACSAC)*, 2011.

[37] Stevens Le Blond, Cédric Gilbert, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and David R. Choffnes. A Broad View of the Ecosystem of Socially Engineered Exploit Documents. In *Proc. of the Network & Distributed System Security Symposium (NDSS)*, 2017.

[38] Library of Congress. Microsoft Office Word 97-2003 Binary File Format (.doc). https://www.loc.gov/preservation/digital/formats/fdd/fdd000509.shtml, 2023.

[39] Daiping Liu, Haining Wang, and Angelos Stavrou. Detecting Malicious Javascript in PDF through Document Instrumentation. In *Proc. of the 44th International Conference on Dependable Systems and Networks (DSN)*, 2014.

[40] Tommy Madjar, Corsin Camichel, Joe Wise, Selena Larson, and Chris Talib. OneNote Documents Increasingly Used to Deliver Malware. https://www.proofpoint.com/us/blog/threat-insight/onenote-documents-increasingly-used-to-deliver-malware, 2023.

[41] Dean Malmgren. textract (v1.6.4). https://textract.readthedocs.io/en/stable/, 2017.

[42] Mandiant. Advanced Persistent Threats (APTs). https://www.mandiant.com/resources/insights/apt-groups, 2023.

[43] Microsoft. New Feature in Office 2016 Can Block Macros and Help Prevent Infection. https://www.microsoft.com/en-us/security/blog/2016/03/22/new-feature-in-office-2016-can-block-macros-and-help-prevent-infection/, 2016.

[44] Microsoft. Office VBA Reference. https://learn.microsoft.com/en-us/office/vba/api/overview/#vba-programming-in-office, 2021.

[45] Microsoft. Hide Sheets and Use the xlVeryHidden Constant in a Macro. https://learn.microsoft.com/en-us/office/troubleshoot/excel/hide-sheet-and-use-xlveryhidden, 2022.

[46] Mamoru Mimura. Using Fake Text Vectors to Improve the Sensitivity of Minority Class for Macro Malware Detection. *Journal of Information Security and Applications*, 2020.

[47] Mamoru Mimura and Hiroya Miura. Detecting Unseen Malicious VBA Macros with NLP Techniques. *Journal of Information Processing*, 2019.

[48] Mamoru Mimura and Taro Ohminami. Towards Efficient Detection of Malicious VBA Macros with LSI. In *Proc. of the 14th International Workshop on Security (IWSEC)*, 2019.

[49] Jens Müller, Fabian Ising, Christian Mainka, Vladislav Mladenov, Sebastian Schinzel, and Jörg Schwenk. Office Document Security and Privacy. In *Proc. of the 14th USENIX Workshop on Offensive Technologies (WOOT)*, 2020.

[50] Amirreza Niakanlahiji and Pedram Amini. Extracting "Sneaky" Excel XLM Macros. https://inquest.net/blog/extracting-sneaky-excel-xlm-macros/, 2019.

[51] Amirreza Niakanlahiji and Pedram Amini. Getting Sneakier: Hidden Sheets, Data Connections, and XLM Macros. https://inquest.net/blog/getting-sneakier-hidden-sheets-data-connections-and-xlm-macros/, 2020.

[52] Yuhei Otsubo, Mamoru Mimura, and Hidehiko Tanaka. O-Checker: Detection of Malicious Documents Through Deviation from File Format Specifications. *BlackHat USA*, 2016.

[53] Outflank. Old School: Evil Excel 4.0 Macros (XLM). https://web.archive.org/web/20240515191958/https://www.outflank.nl/blog/2018/10/06/old-school-evil-excel-4-0-macros-xlm/, 2018.

[54] Pierluigi Paganini. Russia-Linked APT 28 Group Observed using DDE Attack to Deliver Malware. https://web.archive.org/web/20240511190941/https://securityaffairs.com/65318/hacking/dde-attack-apt28.html, 2017.

[55] Seongsu Park and Vitaly Kamluk. The BlueNoroff Cryptocurrency Hunt is Still on. https://web.archive.org/web/20240519163632/https://securelist.com/the-bluenoroff-cryptocurrency-hunt-is-still-on/105488/, 2022.

[56] Michael Raggi. Chinese APT TA413 Resumes Targeting of Tibet Following COVID-19 Themed Economic Espionage Campaign Delivering Sepulcher Malware Targeting Europe. https://www.proofpoint.com/us/blog/threat-insight/chinese-apt-ta413-resumes-targeting-tibet-following-covid-19-themed-economic, 2020.

[57] Michael Raggi. Injection Is the New Black: Novel RTF Template Inject Technique Poised for Widespread Adoption Beyond APT Actors. https://www.proofpoint.com/uk/blog/threat-insight/injection-new-black-novel-rtf-template-inject-technique-poised-widespread, 2021.

[58] RedxorBlue. Executing Macros from a DOCX with Remote Template Injection. https://web.archive.org/web/20240515180936/https://blog.redxorblue.com/2018/07/executing-macros-from-docx-with-remote.html, 2018.

[59] Florian Roth. Signature-Base. https://github.com/Neo23x0/signature-base/tree/master/yara, 2021.

[60] Nicola Ruaro, Fabio Pagani, Stefano Ortolani, Christopher Kruegel, and Giovanni Vigna. SYMBEXCEL: Automated Analysis and Understanding of Malicious Excel 4.0 Macros. In *Proc of the 43rd IEEE Symposium on Security and Privacy (S&P)*, 2022.

[61] Alex Scroxton. IT Leaders Fear 'Trickle-Down' of Nation-State Cyber Attacks. https://web.archive.org/web/20240515181415/https://www.computerweekly.com/news/252505571/IT-leaders-fear-trickle-down-of-nation-state-cyber-attacks, 2021.

[62] Chintan Shah. An Inside Look into Microsoft Rich Text Format and OLE Exploits. https://web.archive.org/web/20240517151249/https://www.mcafee.com/blogs/other-blogs/mcafee-labs/an-inside-look-into-microsoft-rich-text-format-and-ole-exploits/, 2020.

[63] Chintan Shah. The Tale of Two Exploits - Breaking Down CVE-2023-36884 and the Infection Chain. https://www.trellix.com/about/newsroom/stories/research/breaking-down-cve-2023-36884-and-the-infection-chain/, 2023.

[64] Charles Smutz and Angelos Stavrou. Malicious PDF Detection Using Metadata and Structural Features. In *Proc. of the 28th Annual Computer Security Applications Conference (ACSAC)*, 2012.

[65] Nedim Šrndic and Pavel Laskov. Detection of Malicious PDF Files Based on Hierarchical Document Structure. In *Proc. of the 20th Network & Distributed System Security Symposium (NDSS)*, 2013.

[66] Etienne Stalmans and Saif El-Sherei. Macro-less Code Exec in MS Word. https://sensepost.com/blog/2017/macro-less-code-exec-in-msword/, 2017.

[67] TrendLabs APT Research Team. Spear-Phishing Email: Most Favored APT Attack Bait. https://documents.trendmicro.com/assets/wp/wp-spear-phishing-email-most-favored-apt-attack-bait.pdf, 2012.

[68] Umut Tosun. How to Analyze RTF Template Injection Attacks. https://www.letsdefend.io/blog/how-to-analyze-rtf-template-injection-attacks, 2022.

[69] Jia Yan, Ming Wan, Xiangkun Jia, Lingyun Ying, Purui Su, and Zhanyi Wang. DitDetector: Bimodal Learning based on Deceptive Image and Text for Macro Malware Detection. In *Proc. of the 38th Annual Computer Security Applications Conference (ACSAC)*, 2022.

[70] Junfeng Yang. How RTF Malware Evades Static Signature-Based Detection. https://www.mandiant.com/resources/blog/how-rtf-malware-evad, 2024.

[71] Xiaopeng Zhang. New Remcos RAT Variant is Spreading by Exploiting CVE-2017-1188. https://web.archive.org/web/20240516131624/https://www.fortinet.com/blog/threat-research/new-remcos-rat-variant-is-spreading-by-exploiting-cve-2017-11882, 2018.

[72] Xin Zhou and Jianmin Pang. Expdf: Exploits Detection System Based on Machine-Learning. *International Journal of Computational Intelligence Systems*, 2019.