Computers & Security 114 (2022) 102582

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

TC 11 Briefing Papers

Invoice #31415 attached: Automated analysis of malicious Microsoft Office documents

Vasilios Koutsokostas[a], Nikolaos Lykousas[a], Theodoros Apostolopoulos[a], Gabriele Orazi[c], Amrita Ghosal[d], Fran Casino[b,e], Mauro Conti[c,f], Constantinos Patsakis[a,b,*]

[a] Department of Informatics, University of Piraeus, 80 Karaoli & Dimitriou str., 18534 Piraeus, Greece
[b] Information Management Systems Institute, Athena Research Center Artemidos 6, Marousi 15125, Greece
[c] University of Padua, Padua, Italy
[d] University of Limerick, Ireland
[e] Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Tarragona, Spain
[f] Faculty of Electrical Engineering, Mathematics and Computer Science Delft University of Technology, NL

ARTICLE INFO

Article history:
Received 12 August 2021
Revised 17 October 2021
Accepted 17 December 2021
Available online 21 December 2021

Keywords:
Malware
Office documents
Macro malware
Powershell
LOLBAS

ABSTRACT

Microsoft Office may be by far the most widely used suite for processing documents, spreadsheets, and presentations. Due to its popularity, it is continuously utilised to carry out malicious campaigns. Threat actors, exploiting the platform's dynamic features, use it to launch their attacks and penetrate millions of hosts in their campaigns.

This work explores the modern landscape of malicious Microsoft Office documents, exposing the means that malware authors use. We leverage a taxonomy of the tools used to weaponise Microsoft Office documents and explore the modus operandi of malicious actors. Moreover, we generated and publicly shared a specially crafted dataset, which relies on incorporating benign and malicious documents containing many dynamic features such as VBA macros and DDE. The latter is crucial for a fair and realistic analysis, an open issue in the current state of the art. This allows us to draw safe conclusions on the malicious features and behaviour. More precisely, we extract the necessary features with an automated analysis pipeline to efficiently and accurately classify a document as benign or malicious using machine learning with an F_1 score above 0.98, outperforming the current state of the art detection algorithms.

1. Introduction

Microsoft Office is by far the most widely used office suite. The documents that are produced from the suite are by no means static and contain many dynamic elements to make them aesthetically more pleasing wile offering advanced functionalities and interaction. The dynamic nature of the elements is further augmented by the use of a programming language VBA (Visual Basic for Applications). However, the many benefits that the support from such a language can provide come at a great cost. The reason is that an adversary may armour the documents to launch an attack. In fact, malicious office documents are widely used by malicious entities to spread malware worldwide, usually through emails. As reported by various sources (Avira, 2020; ESET, 2020), MS Office documents are the second most widely used file format used by malware for Windows and continuously used in malspam campaigns (Crowdstrike, 2020; Patsakis and Chrysanthou, 2020).

The main reason that these files are used so often in such campaigns is that they are continuously exchanged by users. Therefore, users will more likely download and open an MS Office file that they received, even from an unknown sender, than, e.g., an executable or an "exotic" file format. In this regard, an MS Office file is used by an adversary to set foot on the victim's machine and then proceed with the actual infection of the host. While there are many specific checks by AVs to prevent the infection from such files, on top of several specially crafted controls from MS that allow users to selectively grant execution permissions, users still fall victims of these attacks. Notably, one of the most notorious malware worldwide that was recently taken down Malwarebytes, Emotet, was using armoured MS Office documents to infect the

* Corresponding author.
E-mail addresses: nlykousas@unipi.gr (T. Apostolopoulos), francasino@unipi.gr (F. Casino), conti@math.unipd.it (M. Conti), kpatsak@unipi.gr (C. Patsakis).

https://doi.org/10.1016/j.cose.2021.102582

hosts, causing huge losses to its victims (Cybersecurity and Infrastructure Security Agency CISA, 2020; Department of Justice, United States, 2021).

1.1. Motivation and contribution

The goal of this work is to perform a thorough analysis of malicious MS Office documents to date and establish a baseline understanding of their modus operandi through an automated approach. To this end, we established an automated pipeline that includes a set of steps that analyse the documents both statically and dynamically, and extracts a set of features that can be used to facilitate the classification of documents from benign to malicious.

The contribution of this work is manifold. The bulk of the work in malicious MS Office documents is focusing on the features that can be used for detecting malicious documents, but not on the actual reasons why these exist. In this context we actually reveal what a malicious document does. Contrary to simply stating that, e.g., hex strings are used, through extended experimentation we show that despite their potential capabilities, these malicious MS Office documents are actually droppers; they download and execute a malicious payload. In this context, our paper is the first to quantify the problem showcasing which are the means to execute the dropped payload. To the best of our knowledge, this is the first paper in the academic literature using deobfuscation to show this while most authors are simply considering the obfuscation as an indication of maliciousness. The latter means that they detect the symptom but not its cause. Additionally, beyond simply reporting the used LOLBAS/LOLBIN, we show the diversity of these choices which indicates that some EDR and EPP solutions, despite knowing this attack vector are not blocking them efficiently. The latter justifies the high impact of malware campaigns that use this approach.

A very important contribution is also the dataset that we are using. As we later discuss in the manuscript, biased datasets which for instance do not include MS Office documents with VBA code may easily exhibit outstanding results, but in practice, perform rather poorly. On the contrary, our dataset is crafted to prevent such biases. The latter is used to identify features that are not used in the literature, e.g., VBA stomping, DDE etc. which enable us to detect malware samples that exploit, e.g., new MS Office vulnerabilities, unknown at the time that the article was originally submitted. The latter clearly illustrates the efficacy and potential of the proposed method. Finally, the selected features and machine learning approach clearly outperform existing work in the field. Due to the characteristics of this dataset, for the reputability of our results and to advance the research in the field the dataset is provided in Zenodo (Koutsokostas et al., 2021).

In what follows, we first provide an overview of the related work regarding malware and office documents. To this end, we analyse their structure, the methods which are used to armour them, some countermeasures and detection methods, and some open-source tools that are be used to weaponise MS Office files. Then, we discuss our methodology in terms of data collection, processing and analysis. Afterwards, in Section 3 we provide an overview of our dataset and some exploratory analysis of its content. Section 5 analyses our findings and Section 6 discusses the methods that were identified from each part of our methodology, as well as their efficiency and drawbacks. Finally, the manuscript concludes summarising our findings and contributions.

2. Related work

In this section we provide the readers with the necessary background information and overview of the related work that will be used in our work. Therefore, we first discuss the structure of documents that are supported by MS Office. Then, we present the basic methods that are used by adversaries to armour an MS Office file and the methods that are used to detect such documents. Finally, we present the most well-known open-source tools that are used to either create malicious MS Office documents or to enrich their capabilities.

2.1. Files supported by MS Office and their structure

MS Office supports several document formats; nevertheless, the core of most of the documents is XML. These formats were introduced around two decades ago in Office XP to support an XML-based format for Excel and continued in 2002 with another XML format for Word. These formats were integrated and became the default formats for MS Office 2003. However, ever since MS Office 2007, MS has adopted the *Office Open XML* format, also known as OpenXML or OOXML format, which allows interoperability with other similar suites and processing software. The Office Open XML format is based on XML and has been adopted by ECMA International as ECMA-376 (Ecma International, 2006) and became an international standard (ISO/IEC 29500) (International Organization for Standardization, 2016). Evidently, for compatibility and interoperability, MS Office supports all the previous formats.

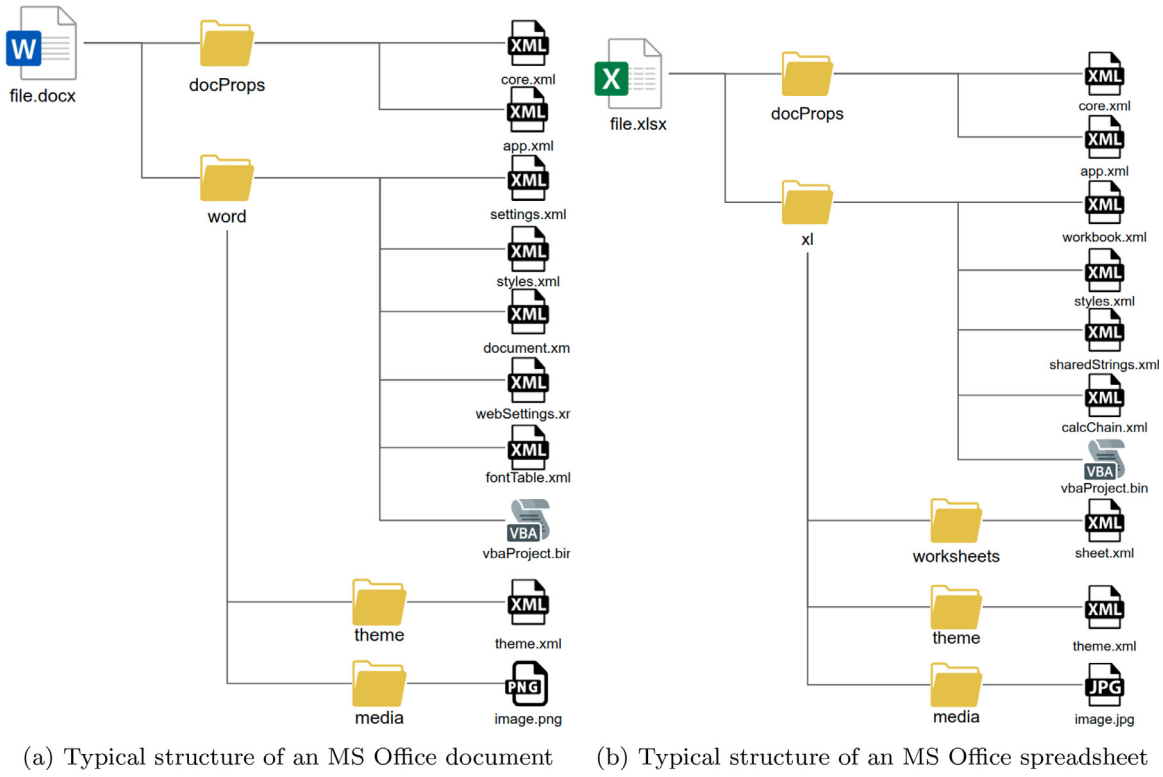
The files that are following the OOXML format are packages containing several specific XML files along with multimedia files and scripts, that are compressed in the form of ZIP file. The key element of the package is the `[Content_Types].xml` file that is stored in the root of the package. This file acts as the index of the package and lists all content types of the parts that the package contains. Two key folders are `_rels`, which stores the relationships between the other parts and resources outside of the package, and `docProps` that contains the core properties of the OOXML file. Then, depending on the file, one may find a `word` or `xl` folder that contains the content of the file. In these folders one may typically find the `vbaProject.bin` file; an OLE Compound Document (Microsoft, 2020a) in binary format, which contains the VBA code for the macros that the OOXML may have. The typical structure of an OOXML document and spreadsheet file is illustrated in the left and right side of Fig. 1, respectively.

Beyond the above, MS Office supports other formats, with the most significant from the security perspective being the Compound File Binary Format (CFBF) which are structured storage files (Microsoft, 2018b), XLSB files (Microsoft, 2021) for MS Excel, which have binary format and are significantly faster than traditional XLS/XLSX files.

2.2. Malicious MS Office files and their detection

It is not the plethora of different file formats that MS Office supports that creates issues, but the fact that these formats are designed to support *dynamic* documents. This dynamism is enabled through various components and modules that use many programming parts. The most obvious part is the support for VBA, which allows someone to write arbitrary code and execute it whenever deemed necessary, even automatically when the file is opened or closed. What is important is that the VBA code is not isolated within the MS Office document context, but it can interact with the filesystem, exchange data over the Internet, and execute shell commands. Evidently, the above expose users to critical risks and have been widely used in the scope of cyberattacks for many years. Beyond VBA, one may perform the same tasks using XLM macros (Microsoft, 2020b) and the Dynamic Data Exchange (DDE) protocol (Microsoft, 2018a) or use pure XML documents (Mendrez, 2015).

A typical attack using an MS Office document is illustrated in Fig. 2. Once the user opens a malicious file with MS Of-



(a) Typical structure of an MS Office document (b) Typical structure of an MS Office spreadsheet

Fig. 1. Typical structures of a document and a spreadsheet OOXML files (Casino et al., 2021b).

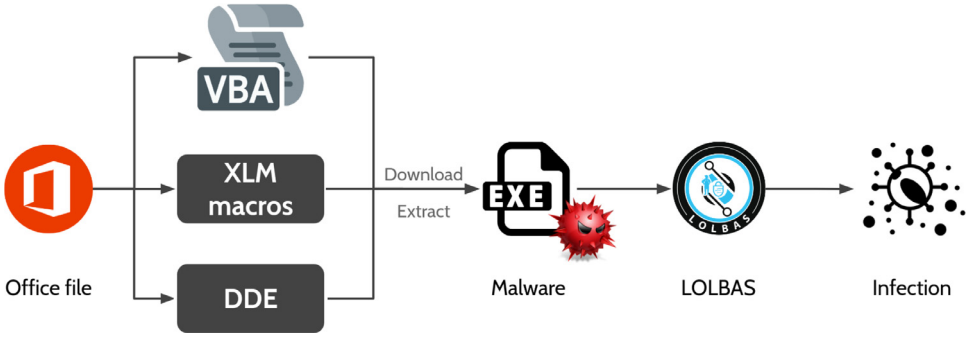


Fig. 2. Overview of an attack through malicious MS Office documents.

fice, the adversary will use a trigger to launch a VBA script (using, e.g., `Workbook_Open()`, `AutoOpen()`, and `AutoClose()` functions) or automatically evaluate dynamic values of the document that are linked with DDE (using e.g., Equation Editor or cells equations) or XLM macros in associated with cells. While the code could perform several tasks, in most campaigns, e.g., Emotet (Patsakis and Chrysanthou, 2020), the code will either download a payload from the Internet or extract it from the document itself. However, this introduces an issue for the adversary in the latest Windows versions as the downloaded file would not be signed by a trusted authority. Hence, the User Account Control (UAC) of Windows would request explicit user consent to allow the execution of the file. Since the user would most likely reject such request, malicious documents follow a different approach. Instead of executing the malicious payload directly, they resort to a "proxy" trusted by the operating system, e.g., PowerShell, Explorer. There are several binaries and scripts which are preinstalled in Windows or downloaded by MS and are either signed or whitelisted by the operating system and enable additional and exploitable functionalities to be performed. Due to their signature, they bypass the UAC of Windows, allowing among others the execution of arbitrary code, code compilation, as well as downloading/upload files, process dumping and collection of credentials, without requesting any user interaction. These binaries and scripts are known as *Living Off The Land Binaries and Scripts (and also Libraries)* or LOLBAS (Campbell et al., 2020) and are widely used by red teams and malware. As a result, the malicious documents utilise LOLBAS to execute the malicious payload and infect the host in a seamless way.

To prevent their detection and analysis, MS Office documents have their macros obfuscated using junk code, various encodings (such as base64, hex, octal), break strings into smaller ones, results of functions, or abuse MS Office related functions. Inherent encryption features are also used to prevent the analysis of their code. In these cases, the adversary would have either appended the decryption password with the phishing email to the victim or abuse an old Excel bug (Lopera, 2020; Mendrez, 2020; Zhang, 2020). Finally, it is worth noticing that MS Office has a compiled version of the VBA code for robustness and performance, called *p-code*. This feature is being abused lately to execute malicious code. More precisely, in this attack, called *VBA stomping* Vesselin Bontchev, (Harold Ogden and Kirk Sayre, 2018), the adversary removes the VBA code from the document; however, MS Office executes the payload from the stored p-code. Therefore, the malicious payload of the VBA code is hidden from the AVs.

To prevent VBA code execution, the latest versions of MS Office require the user to authorise the execution through a button that displays "*Enable content*". However, to detect malicious MS office documents most researchers have devoted their efforts to using natural language processing methods to extract several features of the VBA code, both in terms of the code and in terms of used functions Kim et al. (2018); Mimura (2019); Mimura and Ohminami (2019). The approach is the detection of a symptom and not the infection, that is the presence of obfuscated code in VBA macros. Therefore, the scope is to detect the imbalances in the representation of characters in the VBA code measuring the entropy, length of words, total characters in code and comments etc. In terms of code, some of them tend to enumerate the native functions used and group them in terms of content, such as text, math, conversion, etc. and use a machine learning classifier to determine its efficacy.

Similarly, researchers use n-grams (Bearden and Lo, 2017) entropy and other byte-level statistics over data stream fragments (Rudd et al., 2018) to detect obfuscation and therefore characterise files as malicious. Finally, (Nissim et al., 2016) have also exploited the more perplex structure of the folder and files in the OOXML structure to classify documents as malicious.

Despite the fact that Microsoft has disabled the automatic execution of the embedded code, demanding the explicit user consent of the user to do so, as common practice has shown, this has not solved the problem. Indeed, due to the complexity of MS Office documents there are several bypasses that occasionally appear and may render this measure useless. Well-known examples include the use of DDE (e.g., in the Hancitor malspam campaign), XLM macros (e.g., used in Quakbot's malspam campaign), or recently in the ZLoader campaign with the use of another MS Office file that is downloaded[1]. Evidently, most the above methods cannot be combated through MS Office as these security checks fall beyond its scope. Therefore, Microsoft has integrated many such security checks in Defender for Endpoint[2].

For more on malicious MS Office documents, the interested reader may refer to Müller et al. (2020); Singh et al. (2020).

2.3. Open-source VBscript and Office malicious generators

Recent analyses (Ben Koehl & Joe Hannon, 2020; Jazi and Segura, 2020; Team, 2018) show that open-source tools, which weaponise MS Office documents created by researchers to facilitate red-teaming adversary simulations, are used by APT groups in phishing campaigns. In the following paragraphs, we provide a brief analysis of the most powerful open-source tools that can assist in the pipeline of generating malicious documents as some of them focus on a single aspect, e.g., obfuscation. We provide an overview of these tools in Table 1.

macro_pack Nasi is one of the most powerful tools in malicious document creation process and supports a plethora of evasion techniques and format outputs. Its execution methods include WMI, Wscript, COM objects, XLM macro, Task Scheduler, InvokeVerb, CreateProcess and Run PE. It supports a diversity of output formats ranging from Office formats to VBScript formats such as VBS, HTA, WSF, SCT and XSL. It also serves a great number of evasion techniques including AV Bypass, VB and Command line Obfuscation, Self decode in memory, Run in Excel memory, Anti-Malware Scan Interface (AMSI) bypass, Social Engineering tricks, Anti sandbox, run an executable in memory, ASR bypass, UAC bypass, and XLM Injection.

EvilClippy Hegt is a tool that can be used to manipulate existing malicious Office documents to evade static analysis by antivirus engines and VBA macro analysis tools. It accomplishes that by implementing a number of techniques such as the hiding of VBA macros from the GUI editor, implementing VBA stomping, randomising module names in the directory stream or locking the VBA Project.

WePWNise Yiu can generate VBA code that adds a level of intelligence to identify weaknesses and dynamically deliver its payload, bypassing Software Restriction Policies (SRPs) and Enhanced Mitigation Experience Toolkit (EMET) protected binaries. To achieve this, it enumerates registry settings to identify unprotected binaries that are safe to inject the payload via WINAPI calls in VBA.

CACTUSTORCH (Jazi and Segura, 2020) is a shellcode execution framework that utilises the DotNetToJScript technique to execute shellcode in Windows scripting formats like VBScript and Javascript. DotNetToJScript is a method to reflectively load a.NET assembly using native Windows scripting languages.

SharpShooter (Team, 2018) is a payload generation framework. Among its many capabilities, it allows the creation of payloads in various formats such as HTA, JS, VBS, WSF. It can also create Excel 4.0 SLK Macro enabled documents and execute arbitrary C# code from a VB script. For example, it can create a VBS file that executes Mimikatz. It can also incorporate AMSI bypass and anti-sandbox analysis among its many techniques.

EXCELntDonut Security is a tool that can output XLM macros from a C# source file and also apply sandbox checks and obfuscation.

Macrome Weber is a tool that can also build XLM Macro (Excel 4.0 macros) documents from shellcode input as well as apply a level of obfuscation.

LuckyStrike Lang is another powerful malicious document generator that can embed standard shell commands, custom PowerShell scripts, or even executable files (.exe) as payloads. Moreover, it uses Invoke-Obfuscation to obfuscate the payloads. It infects the document in many different ways, including Wscript.Shell to fire the command in a hidden window, cell embedding of base64 encoded or encrypted PowerShell scripts. Thereafter, it can execute them in a fileless way without touching the disk, embedding a base64 encoded binary file into cells which it then saves as a text file to disk and uses certutil to decode and execute it or insert the payload into the metadata of the malicious file in the Subject field. It can also make use of Invoke-ReflectivePEInjection to execute a b64 encoded PE that is dropped on disk or inject it into another process.

3. Methodology

To perform an accurate analysis for state of the art in malicious office documents, we need to establish a rigorous methodology for the tasks that have to be performed and assess the expected outcomes from each step. In general, our methodology consists of three phases, each of which has specific tasks that are performed. An outline of our methodology used for the analysis in this work is illustrated in Fig. 3. The methodology consists of three major steps that are analysed in the following paragraphs.

Data Collection In this step, we collected all available data from public sources. We would like to highlight the public nature of the document sources here as the use of private sources implies several methodological issues. The most obvious one is the legal aspect as organisations are quite reluctant to share documents. Even if the documents are malicious, since they might be products of a spear-phishing attack, e.g., intercepted internal documents might have been used, they do not share them. However, by finding them in public sites and/or public malware repositories, practically the recipients have given their consent to use them, alleviating any le-

[1] https://securityaffairs.co/wordpress/119902/hacking/malspam-new-evasion-technique-macro.html

[2] https://www.microsoft.com/security/blog/2021/03/03/xlm-amsi-new-runtime-defense-against-excel-4-0-macro-malware/

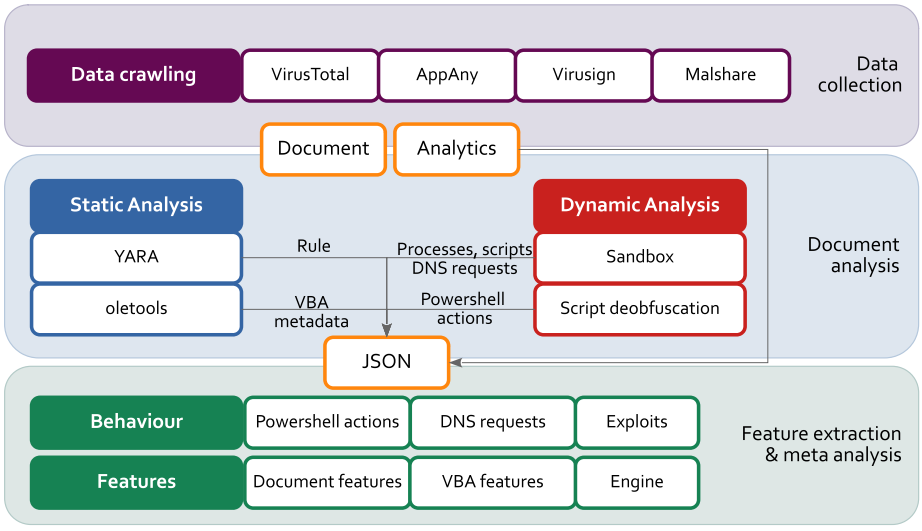


Fig. 3. An overview of the methodology used in our analysis.

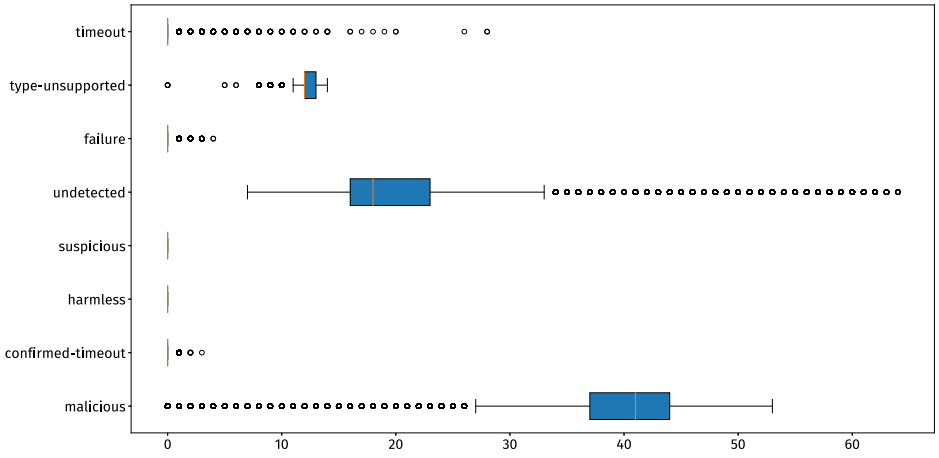


Fig. 4. Distribution of the different verdict outcomes collected from the VT scan analysis.

gal issues for the collection and processing. Moreover, the public sources are more unbiased as contrary to private sources they contain samples from more campaigns. In the case of benign samples, we randomly crawled different official public sites, more precisely a total of 726 governmental and 1010 educational sites belonging to disparate countries and institutions, with an automated pipeline that queried Google for MS Office docs with `xls`, `xlsx`, `doc` and `docx` extensions in such sites. The retrieved files were inspected to determine whether they included macros or not, and the latter were discarded. Thereafter, these files were submitted to Triage and VirusTotal to confirm that they are not malicious. After these operations, a total of 2736 benign files were selected for our dataset. Clearly, the aforementioned benign files introduce a bias against our methodology as typical files would not meet these requirements, nonetheless, it is the best approach to stress the efficacy of our approach. Therefore, our database is created to exemplify the worst case scenario, in which we try to distinguish malicious files from benign ones, in both cases using macros and DDE. Note that if a file does not contain macros (hidden VBA and DDE are detected by our features, as later described in Section 5), the file can be classified as benign without the need to compute further features, thus easing the classification task.

In the case of malicious files, we used three well-known repositories, namely AppAny, Virusign and Malshare to collect our samples. Moreover, we used VirusTotal to collect some further information about our sample. More precisely, we collected information about the detection rate of various antivirus products on these documents as well as additional information regarding the DNS calls that these samples made.

Document analysis In the second phase, we analysed our documents statically and dynamically. In our static analysis we extracted all possible metadata using oletools Lagadec and ExifTool Harvey, and then extracted the VBA code with oletools. In our dynamic analysis, we executed all samples in a sandbox environment collecting all PowerShell actions, all processes that were opened, and all DNS requests that were performed. Furthermore, for each of the collected PowerShell scripts, we tried to perform automatic deobfuscation.

Feature extraction & meta analysis All the aforementioned collected information was stored in a JSON format to facilitate processing for the final phase. In this final phase, we performed a thorough analysis of the results, trying to extract several behavioural and static features that our sample exhibits. For instance, we wanted to determine which are the most common used methods and actions that are used in VBA and PowerShell from these documents. Moreover, we wanted to determine whether the documents are using specific exploits. Finally, we explored the relations that these samples had regarding the DNS calls they performed.

Table 1
List of open-source generators assisting maldoc creation.

Tool	Payload Format	Obfuscation	Evasion
macro_pack	Microsoft Office (Word, Excel, PowerPoint), MS Project, MS Visio, MS Access, VBS, HTA, SCT, WSF, XSL LNK, SLK, SCF, CHM, Visual Studio Project INF, IQY	✓	✓
EvilClippy	-	✗	✓
WePWNise	VBA	✗	✓
CACTUSTORCH	VBS, VBA, JS, JSE, WSF, HTA, VBE	✓	✗
SharpShooter	HTA, JS, VBS, VBA, WSF, SLK	✓	✓
EXCELntDonut	XLM	✓	✓
Macrome	XLS	✓	✗
LuckyStrike	XLS, DOC	✓	✓

4. Dataset exploration

In addition to the 2736 benign samples collected by following the methodology reported in Section 3, our dataset consists of 15571 malicious samples which date from 2006 to 2020 according to the recorded dates that were collected through VirusTotal, AppAny, Virusign and Malshare. According to their corresponding original names, the sample consists of 13518 Word documents, 1996 Excel spreadsheets and 13 Powerpoint presentations. In VirusTotal, 15571 had been scanned with 14264 of these files being reported as malicious by at least one antivirus (AV). Note that none of the officially shared files was reported to be malicious. However, the reports exhibited great variation in terms of how many antiviruses detected them and which at each time. It should be noted though, that these results refer to the final scan and not the first one, as in average each document had been scanned 9.39 times after users submissions. Practically, this means that several AVs persistently failed to flag them as malicious. As illustrated in Figure 4 there are a lot of variations in the reported outcomes of the AVs. Nonetheless, these results may be attributed to shared signatures or different approaches, e.g., run-time behaviour. In general, since a user may submit a file to VirusTotal for scanning, it is evident that she will receive many reports with different outcomes according to the accuracy of the reported verdict of all AVs at the time of scanning.

In terms of metadata, due to the various formats, we present the corresponding statistics in Table 2. It is clear that while there are many documents where metadata and content have not been taken care of, e.g., they do not have any content inside, have a low amount of edits, etc., in most cases the malware authors have taken measures to show some relevant actions and make them more realistic. From the samples, 15,021 had embedded VBA code that was extracted with oletools. The different statistics and features collected from these files depicted in Table 3. In this regard, we report the size of the VBA code, when the VBA code is executed, how many times the shell command was used, how many samples used Base64 strings, or how many tried to write data to a file. Moreover, in Fig. 5 we present a fragment of the graph that illustrates the interconnection of samples and the domains they tried to connect to. Notably, as illustrated in Fig. 6, most of the identified domains are considered harmless by VirusTotal. The latter can be attributed to the connection with several truly benign domains to collect data, credentials etc., or even temporarily compromised domains where malicious executables may have been injected. For instance `pastebin` is not a malicious service but is often used to dump credentials while Google services, since they are whitelisted they are often exploited or used to exfiltrate sensitive user information.

In addition to the previous analysis, we collected the system calls performed by Living Off The Land Binaries from the PowerShell scripts contained in the dataset files in Table 4, and the PSDecode[3] (a deobfuscation tool for PowerShell) analysis in Table 5. As it can be observed from both tables, `CMD.EXE` appeared a high number of times as the actual task was to download a file from the Internet and execute the file. Note that the use of a LOLBAS implies that no alert would be issued to the user.

5. Experiments

Given the samples collected by using the methodology presented in Section 3, we computed a set of binary features which are described in Table 6. We assess the power of our proposed features to differentiate between malicious and benign samples (i.e., binary classification). We selected a set of machine learning methods to leverage a binary classification task. More concretely, we used Random Forest, a non-parametric ensemble classifier, XGBoost; which implements gradient boosted decision trees, a Support Vector Classifier (SVC), and a Multi-layer Perceptron (MLP) classifier.

The hyperparameters of each model were tuned with a grid search to maximise classification performance in the task of distinguishing between benign and malicious samples. Table 7 summarises the configuration parameters that achieved the highest performance. In the case of both Random Forest and XGBoost, we observe that the maximum number of features used for the classification task is far below the total number of features computed for each file (i.e., we computed 40 different features, see Table 6), which will be discussed later in this section. In the case of the SVC model, we used a radial basis function (rbf) kernel. In all experiments, we employed standard 10-fold cross-validation and repeated such experiment three times to get a roughly unbiased estimate of the performance of predictive models we trained.

All our experiments were performed on a system equipped with an NVIDIA TITAN Xp PG611-c00 to speed-up the computations, while we utilised the implementations of the `scikit-learn`[4] library. We evaluate the performance of the trained classifiers using the standard classifications metrics of precision, recall, accuracy and F_1 score. The outcomes achieved by each model are summarised in Table 8. As it can be observed, the results obtained considering the standard classification metrics for all the classifiers are close to 100%, with Random Forest and MLP exhibiting the best performance compared to the other two models. Moreover, the low values of standard deviation for all the classifiers indicate the robustness of the experimental outcomes.

According to Tables 7 and 8, we observed that the best outcomes were achieved when using only a subset of the features of our system. Therefore, we studied the relevance of the features in the Random Forest, the XGBoost and the SVC models, which are depicted in Fig. 7. As it can be observed, a common subset of features has significant relevance in all the three models. Particularly, we noticed that autopen, shell, creatobject, base64 strings and document_open are the most relevant features in all the three models.

6. Discussion

In this section, we put forward a discussion highlighting the significance of this work and provide a comparative assessment of the current state of the art.

[3] https://github.com/R3MRUM/PSDecode
[4] https://scikit-learn.org

Table 2
Statistics and metadata from the files of our database.

Word	Values			
Property	Min	Max	Average	σ
Author len (not null)	2	52	10.26	5.34
Title len (not null)	1	83	9.24	7.21
Pages	0	525	3.12	15.34
Characters	0	3058756	7114.45	71958.98
Words	0	301,411	1101.81	8579.76
Paragraphs	0	6948	31.27	205.75
Size (bytes)	4608	15996114	260752.19	731835.41
Revisions	0	6933	22.22	155.75
Edit-Creation	0	9435312000	1747626.29	121637315.79
Reported Applications	12	-	-	-
Reported Templates	102	-	-	-

Table 3
VBA statistics collected from our dataset. The symbol * refers to the minimum, average, and maximum size values.

Statistic	Count	Statistic	Count
Size*	0/ 7246.51 / 3,561,235	Xor	599
Hex Strings	12,397	Windows	715
Base64 Strings	10,809	Output	701
Shell	7351	Print	685
Chr	6341	CallByName	666
CreateObject	5908	StrReverse	647
autoopen	3843	Auto_Open	622
AutoOpen	3505	Kill	530
Run	2859	CreateTextFile	495
VBA Stomping	2811	open	489
Document_open	2749	SaveToFile	415
ShowWindow	2612	FileCopy	414
Call	2542	showwindow	410
Open	2346	system	404
ChrW	1791	System	395
Document_Open	1673	Autoopen	394
Write	1371	Shell.Application	375
Environ	1299	Document_Close	342
Lib	1265	run	335
Workbook_Open	1068	call	331
ChrB	960	ShellExecute	315
Binary	920	shell	302
Put	849	write	301
WScript.Shell	833	ActiveWorkbook.SaveAs	299
vbHide	777	output	298

Table 4
Living Off The Land Binaries statistics from PowerShell scripts.

Process	Count	Process	Count
CMD.EXE	20,886	EVENTVWR.EXE	46
SC.EXE	1810	MSBUILD.EXE	40
WSCRIPT.EXE	1702	EXPAND.EXE	39
MSIEXEC.EXE	1232	MAKECAB.EXE	37
WINWORD.EXE	1230	VBC.EXE	36
RUNDLL32.EXE	1076	NETSH.EXE	36
REG.EXE	1019	REGSVCS.EXE	34
EXCEL.EXE	688	POWERPNT.EXE	23
FINDSTR.EXE	541	INSTALLUTIL.EXE	21
SCHTASKS.EXE	487	ESENTUTL.EXE	18
CSC.EXE	432	MSDT.EXE	13
WMIC.EXE	427	RUNONCE.EXE	10
MSHTA.EXE	416	PRESENTATIONHOST.EXE	9
CSCRIPT.EXE	380	UPDATE.EXE	9
CERTUTIL.EXE	379	MSCONFIG.EXE	8
FORFILES.EXE	226	FTP.EXE	5
REGSVR32.EXE	190	HH.EXE	4
MMC.EXE	174	WSRESET.EXE	3
CMSTP.EXE	141	APPVLP.EXE	2
VERCLSID.EXE	102	IE4UINIT.EXE	1
BITSADMIN.EXE	93	MSXSL.EXE	1
REGEDIT.EXE	65	PCWRUN.EXE	1
REGASM.EXE	61	WAB.EXE	1
CONTROL.EXE	55	RASAUTOU.EXE	1

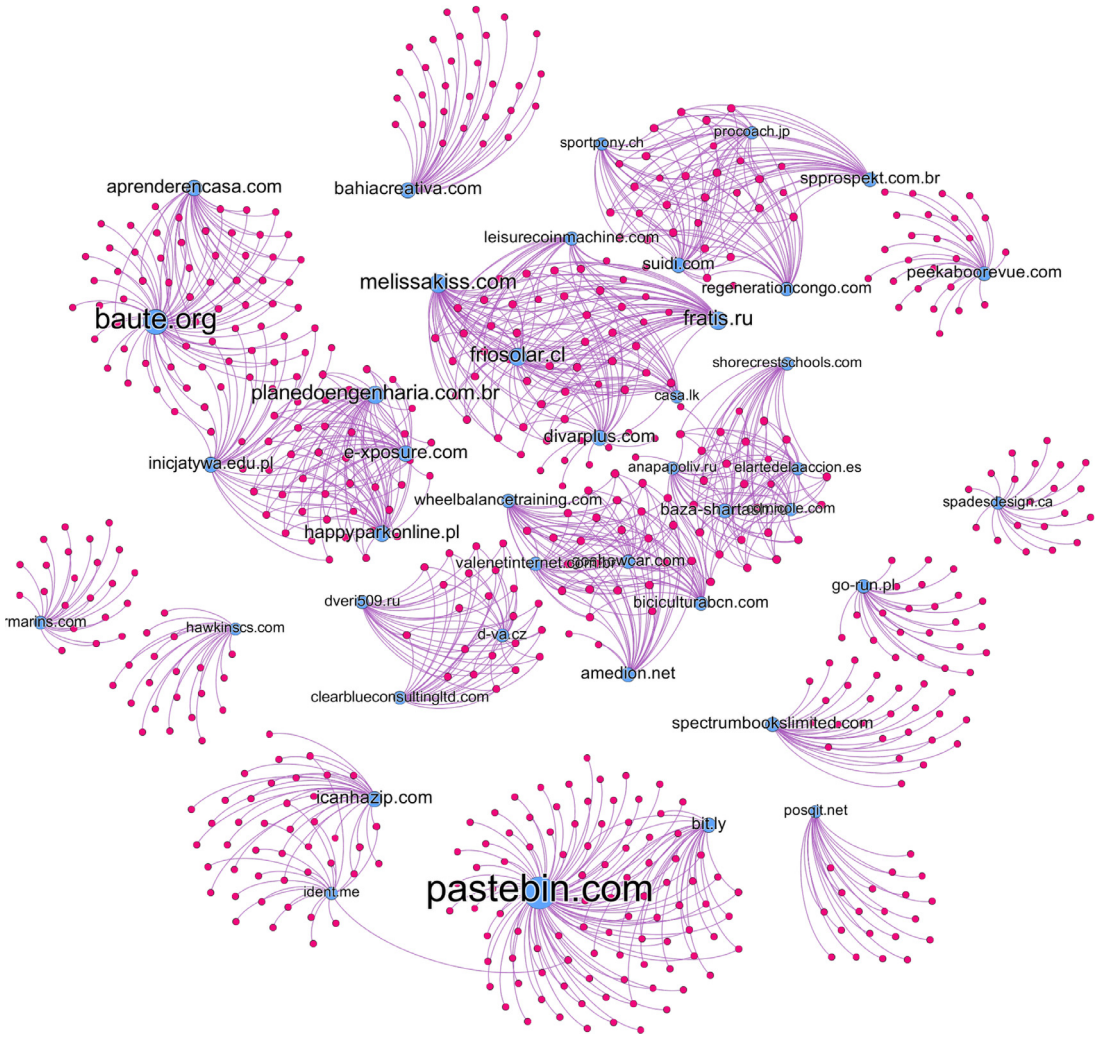


Fig. 5. Connections between domains used in the samples.

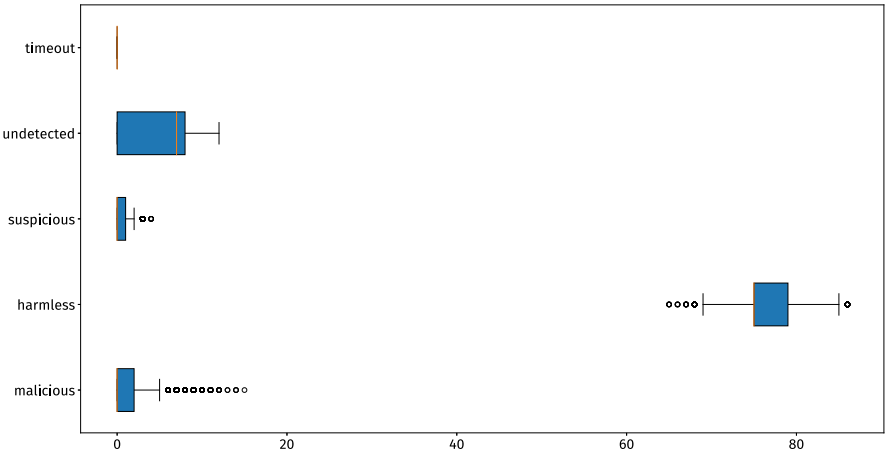


Fig. 6. Statistics of the domains scanned with VT.

6.1. Robustness of the proposed methodology

In this work, at first, we performed a detailed analysis of malicious MS Office documents to understand their functionality using automated analysis. The documents were also analysed both statically and dynamically, for extracting feature sets that would prove beneficial for classifying the documents as malware or benign. To be more specific, we used machine learning techniques that provide an F_1-score that outperforms the current state of the art. To enhance the performance in real world scenario, the samples we chose for performing the experiments are specific benign files that have macros and VBA code. Such dataset is used to identify features that are not used in the literature, e.g., VBA stomping, DDE which enable us to detect malware samples that exploit, e.g., new MS Office vulnerabilities, unknown at the time that the article was originally submitted. Moreover, as stated in Section 3 if a

Table 5
Outcomes of the PSDecode Analysis.

Action	Count
System.Net.WebClient.DownloadFile	23,668
Get-Item.length	22,398
Invoke-Item	1763
System.Net.WebClient.DownloadString	107

file does not contain macros (hidden VBA and DDE are detected by our features, see Table 6), the file can be classified as benign without the need to compute further features. One may argue that the proposed features are derived from static analysis and therefore an adversary may bypass them. While the claim is valid, there are practical limitations in the dynamic setting. Contrary to binaries, which constitute the bulk of malware, MS Office documents are constantly exchanged among colleagues among and across organisations which require immediate processing. The sere amount of all these documents and the rate that they are exchanged leaves little if any time for dynamic analysis. Moreover, their dynamic analysis requires a dedicated host which will open them and monitor their execution.

We argue that an optimal way to handle this volume and velocity; as actually done in practice, is the two layer approach. First, we have the static analysis, where our approach fits ideally, to prune the documents that are detected as malicious. Then, endpoint security mechanisms such as EPPs and EDRs, which are monitoring the calls of all applications for suspicious patterns may intervene. They can detect many anomalous patterns, e.g. Word opening a LOLBAS/LOLBIN, using machine learning, and depending on their capabilities even block it. However, since specially crafted attacks may bypass them (Karantzas and Patsakis, 2021), the sieving phase from the static analysis may significantly improve the robustness of an information system against cyber attacks.

6.2. Comparison with the state of the art

In what follows, we perform a descriptive comparison with the state of the art in terms of dataset, features, classification methods, and reported detection accuracy. A summary of the qualitative aspects of each work can be found in Table 9.

The main idea behind the work presented in Nissim et al. (2016) is to extract and analyse the structural paths contained in docx files. Such paths are processed and converted to features used by an SVM classifier enhanced with Active Learning. The dataset collected for their experiments contains 16,811 docx samples curated using VirusTotal, 16,484 of them benign, with less than 0.5% of such benign files containing macros. The latter implies that the underlying dataset is extremely unbalanced. The authors used several machine learning classifiers, namely J48, RF, LogitBoost, Logistic Regression and SVM, and reported recall values of 93.48% in the best case with SVM. Moreover, they also obtained accuracy values of 99.6%, yet such value is hard to interpret since only 1.9% of the dataset are malicious files despite the

Table 6
Features used in our approach and their corresponding description.

Notation	Description
`base64 strings`	Indicates whether the VBA code contains base64 strings.
`kill`	The VBA code contains the kill command to delete files.
`call`	VBA uses the `call` statement to transfer control to another procedure.
`callbyname`	VBA uses the `callbyname` function to manipulate an object.
`shellexecute`	VBA uses `shellexecute` from DLL to execute a command.
`chrw`	VBA uses `chrw` to handle Unicode characters
`shell.application`	VBA creates a shell object using `SHELL32.dll`.
`createobject`	VBA uses `createobject` to create an ActiveX object.
`activeworkbook.saveas`	Saves changes to the workbook in a different file.
`xor`	The VBA code XORs values.
`vba stomping`	Indicates whether the sample was using VBA stomping.
`binary`	The VBA may read or write a binary file.
`strreverse`	The VBA code uses `strreverse` to manipulate strings.
`chr`	VBA uses `chr` to map an integer to ASCII.
`lib`	VBA declares some DLLs to load.
`system`	Run an executable file or a system command on a Mac.
`wscript.shell`	VBA uses `wscript.shell` to execute a shell command.
`document_open`	VBA is automatically launched when the document opens.
`auto_open`	VBA is automatically launched using `auto_open`.
`showwindow`	VBA uses `showwindow` function from DLL to manipulate windows.
`workbook_open`	Execute VBA code automatically once the user opens the workbook.
`print`	VBA code uses write to write a file.
`filecopy`	VBA uses `filecopy` to copy files.
`virtual`	May detect virtualization environment.
`autoopen`	VBA code has macros named `AutoOpen` to automatically execute them.
`open`	VBA uses `open` to manipulate a file.
`shell`	VBA uses `shell` to execute a shell command.
`windows`	VBA code uses windows to enumerate windows.
`write`	VBA code uses `write` function to write a file.
`document_close`	VBA code is executed once the document closes.
`run`	VBA code uses `run` function to run a macro or call a function.
`output`	VBA code uses `output` function to write a file.
`vbhide`	VBA code uses the `vbhide` parameter to hide execution window.
`chrb`	VBA code uses `chrb` string function.
`executeexcel4macro`	VBA executes a Microsoft Excel 4.0 macro function.
`savetofile`	VBA uses `savetofile` to write data to a file.
`environ`	VBA uses `environ` to collect OS environment variables.
`createtextfile`	VBA uses `createtextfile` to create a text file.
`hex strings`	Indicates whether the VBA code contains hex strings.
`dde`	Indicates whether the sample contains DDE.

Table 7
Best configuration parameters of each model.

Model	Best configuration
Random Forest	n_estimators=200, max_depth=20, max_features=10
XGBoost	learning_rate=0.02, max_depth=5, subsample=0.6, max_features=10
Support Vector Classifier	kernel='rbf'
Multi-layer Perceptron	hidden_layer_sizes=(11,11,11), max_iter=500

Table 8
Average outcomes and their corresponding standard deviation σ.

Model	Precision		Recall		Accuracy		F_1-score	
	Average	σ	Average	σ	Average	σ	Average	σ
Random Forest	0.993	0.002	0.976	0.003	0.975	0.003	0.985	0.001
XGBoost	0.989	0.003	0.968	0.003	0.965	0.004	0.979	0.002
Support Vector Classifier	0.992	0.002	0.976	0.003	0.974	0.003	0.984	0.001
Multi-layer Perceptron	0.993	0.002	0.976	0.003	0.975	0.003	0.985	0.002

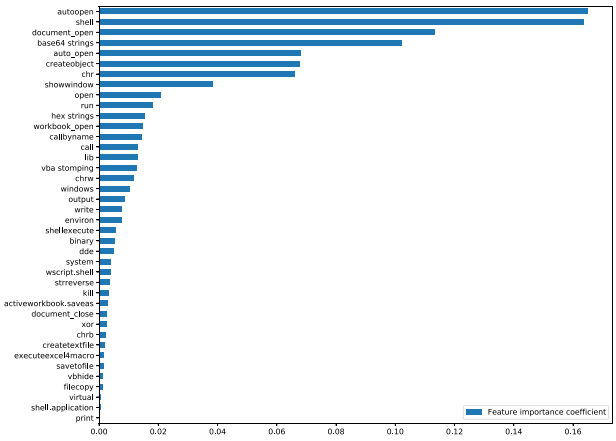
Table 9
A descriptive comparison of our work with the most relevant state-of-the-art approaches.

Reference	Dataset	Features	Method(s)	Outcomes
Nissim et al. (2016)	16,811 docx samples (98.1% of them benign, only 1.9% malicious) collected from VirusTotal, Contagio and Ben-Gurion Uni	Path analysis features	J48, RF, LogitBoost, Logistic Regression and SVM	The best method was SVM and obtained a recall of 93.48% and an accuracy of 99.6%
Kim et al. (2018)	Random collection of MS office files, curated according to VirusTotal analysis. A total of 2537 files and 4212 macros (sometimes more than one per file), from which 877 were obfuscated.	A set of 15 lexicographical and function call features	SVM, RF, MLP, LDA, BNB	The different machine learning approaches report accuracies around 90% in the task of identifying obfuscated macros. MLP was the most prominent with a 92% F_2-score
Mimura (2019)	7145 samples including macros collected from VirtusTotal	Different language processing-related features, including SCDV, LSI, Doc2vec, Bag-of-words	SVM	The best F_1-score reported is 93%
Mimura and Ohminami (2019)	As in Mimura (2019)	LSI	SVM	F_1-score reported near to 95%
Our approach	Benign samples with macros collected from official sites and malicious samples collected from VirusTotal AppAny, Virusign and Malshare, for a total of 2736 benign and 15,571 malicious	Lexicographical, VBA statistics, and function call analysis (LOLBAS, PowerShell, PSDecode)	SVC, RF, XGBoost, MLP	Best F_1-score above 98% and recall above 97.5% with RF. In the case of F_2-score, we obtain a 98% with RF

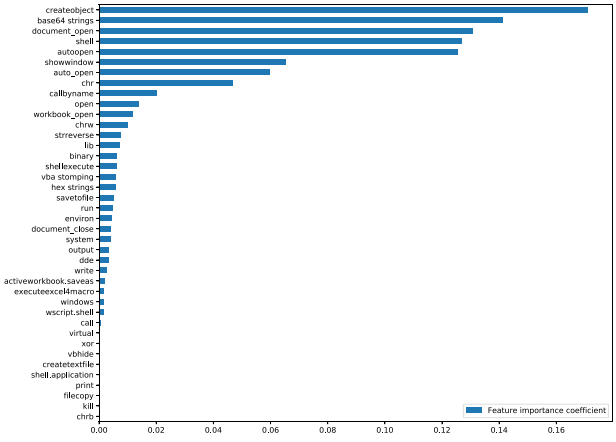
efforts to balance the measurements. Unfortunately, the authors do not report the F_1-score to provide a more fair measurement and comparison with other studies. However, it should be noted that the above structure is not relevant for many of the malicious documents that are used now. Many of the tools that are used nowadays (see Section 2) do not generate such suspicious paths. Moreover, weaponisation with DDE or VBA stomping may not even generate any new path, deprecating such an approach.

In Kim et al. (2018), the authors review the obfuscation detection techniques applied on real-world VBA macros and propose a novel obfuscated macro code detection method by using five machine learning classifiers, namely Support Vector Machines (SVM), Random Forest (RF), MultiLayer Perceptron (MLP), Linear Discriminant Analysis (LDA), and Bernoulli Naive Bayes (BNB). Authors collected a total of 2537 samples, which were curated according to the analysis of VirusTotal. For training the classifiers, their proposed method uses 15 static features. Their evaluation results with selected feature sets show that SVM, RF, and MLP classifiers have an edge among the five classifiers. Notably, RF recorded a precision of 98.2% while MLP recorded a recall of 91.5%. However, both LDA and BNB classifiers were found to be inefficient for detecting obfuscated VBA macros. Also, the authors computed the F_2 score (thus, giving more weight to recall), and reported a 92% score when using the MLP classifier.

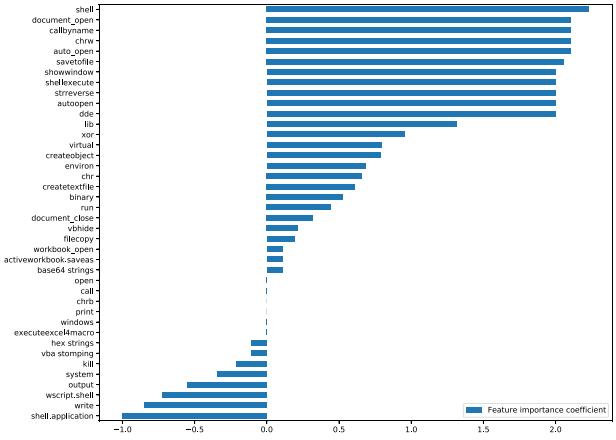
In Mimura and Ohminami (2019), the authors use LSI (Latent Semantic Indexing) for building an efficient language model to achieve better accuracy and efficiency. The detection method used by the authors is the first for detecting new malicious VBA macros with LSI. The words are extracted from the source code and converted into feature vectors using natural language processing techniques. The method used by the authors trains a classifier with benign and malicious VBA macros and detects new malicious VBA



(a) Feature relevance - Random Forest



(b) Feature relevance - XGBoost



(c) Feature relevance - Support Vector Classifier

Fig. 7. Feature relevance coefficient for each model.

macros. The samples were obtained from VirusTotal and they are processed to create a set of time series from them. The authors report their best F_1-score that was achieved to be 95

The work in Mimura (2019) focuses on the use of the feature construction algorithm Sparse Composite document vector (SCDV) and its performance is compared with other language models such as Bag-of-Words, LSI, and Doc2vec. For the detection of malicious VBA macros, the authors proposed a method that extracts words from the source code, and represent such VBA macros by using a language model that can be feed to the classifiers. The performance of such approach was measured using results from the 5-fold cross-validation, highlighting the accuracy of the Doc2vec model over the rest, with an F_1-score of 93%.

It is clear that our method outperforms (Kim et al., 2018; Mimura, 2019; Mimura and Ohminami, 2019) in all relevant metrics. Regarding (Nissim et al., 2016), while the reported accuracy is slightly better, the dataset is heavily biased, as also stated by the authors. Moreover, while our proposed method achieves higher recall, the method of Nissim et al. cannot be applied in many malware families as the structural paths in many samples do not follow the pattern that the authors exploit in their work any more. In summary, taking into consideration the previously discussed literature, our proposed method achieves better results considering the standard classification metrics. Moreover, our approach leverages an automated pipeline for document analysis that considers a rich dataset containing benign documents with VBA and a description of the feature relevance according to each classification method. The use of small and unrepresentative datasets leads to several biases and other issues that can easily lead towards wrong analysis and misleading conclusions. Therefore, we publicly shared our dataset. While this facilitates the reproducibility of our results, we also allow fellow researchers to use a significantly richer baseline dataset to ease comparative experiments, fostering the progress of the state of the art.

7. Conclusions

Despite the various methods that have been introduced over time to deter users from opening malicious MS Office documents, millions of devices are compromised this way, illustrating that AVs must timely detect these documents and prevent users from opening them. Our research has demonstrated that a common modus operandi traverses most of these documents. Notably, while these documents could perform far more tasks, they practically act only as droppers, downloading other executables to do the actual infection of the host.

Using only static features extracted from the documents of our dataset, we showcase that one may classify documents with exceptional performance, even in the case where benign documents have more rich and dynamic features. While in our feature analysis, we observe that some features may negatively impact the outcome, since they can be used, and have been used, by threat actors, we opt for their inclusion. In fact, their inclusion does not impact our method's prevalence over the current state of the art.

In future work, we plan to analyse methods to detect specific tools and further analyse their evasion methods. Moreover, we aim to integrate traffic monitoring analysis combined with data from relevant sources (Casino et al., 2021a; Morato et al., 2018) to leverage the identification of ongoing campaigns in different contexts. Finally, we will explore methods to cluster MS Office documents based on campaigns as well as other methods to weaponise such documents.

Declaration of Competing Interest

The authors declare that they have no financial conflicts of interest.

CRediT authorship contribution statement

Vasilios Koutsokostas: Software, Validation, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Nikolaos Lykousas:** Software, Validation, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Theodoros Apostolopoulos:** Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft,

Writing – review & editing. **Gabriele Orazi:** Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Amrita Ghosal:** Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing. **Fran Casino:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Project administration. **Mauro Conti:** Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition. **Constantinos Patsakis:** Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition.

Acknowledgements

This work was supported by the European Commission under the Horizon 2020 Programme (H2020), as part of the projects CyberSec4Europe (https://www.cybersec4europe.eu) (Grant Agreement no. 830929) and *LOCARD* (https://locard.eu) (Grant Agreement no. 832735). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. F. Casino was supported by the Beatriu de Pinós programme of the Government of Catalonia (Grant No. 2020 BP 00035).

The content of this article does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors.

References

Avira, 2020. Malware threat report: Q3 2020 statistics and trends. https://www.avira.com/en/blog/malware-threat-report-q3-2020-statistics-and-trends.

Bearden, R., Lo, D.C.-T., 2017. Automated microsoft office macro malware detection using machine learning. In: 2017 IEEE International Conference on Big Data (Big Data). IEEE, pp. 4448–4452.

Ben Koehl & Joe Hannon, M. T. I. C., 2020. https://www.microsoft.com/security/blog/2020/09/24/gadolinium-detecting-empires-cloud/.

Campbell, C., Graeber, M., Goh, P., Bayne, J., 2020. Living off the land binaries and scripts. https://lolbas-project.github.io/.

Casino, F., Lykousas, N., Homoliak, I., Patsakis, C., Hernandez-Castro, J., 2021. Intercepting hail hydra: real-time detection of algorithmically generated domains. Journal of Network and Computer Applications 190, 103135.

Casino, F., Totosis, N., Apostolopoulos, T., Lykousas, N., Patsakis, C., 2021. Analysis and correlation of visual evidence in campaigns of malicious office documents. arXiv preprint arXiv:2103.16143.

Crowdstrike, 2020. Malspam in the time of covid-19. https://www.crowdstrike.com/blog/covid19-and-malspam/.

Cybersecurity and Infrastructure Security Agency CISA, 2020. Alert (ta18-201a). https://us-cert.cisa.gov/ncas/alerts/TA18-201A.

Department of Justice, United States, 2021. Emotet botnet disrupted in international cyber operation. https://www.justice.gov/opa/pr/emotet-botnet-disrupted-international-cyber-operation.

Ecma International, 2006. Office open xml file formats. https://www.ecma-international.org/publications/standards/Ecma-376.htm.

ESET, 2020. Eset threat report q3 2020. https://www.eset.com/fileadmin/ESET/CZ/Threat-report/ESET_Threat_Report_Q32020.pdf.

Harold Ogden, Kirk Sayre, C.R., 2018. Vba stomping: Advanced malicious document techniques. DerbyCon 2018.

Harvey, P.,. Exiftool. https://exiftool.org/.

Hegt, S.,. Evilclippy. https://github.com/outflanknl/EvilClippy.

International Organization for Standardization, 2016. Information technology - document description and processing languages - office open xml file formats - part 1: Fundamentals and markup language reference. https://www.iso.org/standard/71691.html.

Jazi, H., Segura, J., 2020. https://blog.malwarebytes.com/malwarebytes-news/2020/10/kraken-attack-abuses-wer-service/.

Karantzas, G., Patsakis, C., 2021. An empirical assessment of endpoint detection and response systems against advanced persistent threats attack vectors. Journal of Cybersecurity and Privacy 1 (3), 387–421.

Kim, S., Hong, S., Oh, J., Lee, H., 2018. Obfuscated vba macro detection using machine learning. In: 2018 48th annual ieee/ifip international conference on dependable systems and networks (dsn). IEEE, pp. 490–501.

Koutsokostas, V., Lykousas, N., Orazi, G., Apostolopoulos, T., Ghosal, A., Casino, F., Conti, M., Patsakis, C., 2021. Malicious ms office documents dataset. https://doi.org/10.5281/zenodo.4559436. 10.5281/zenodo.4559436

Lagadec, P.,. Oletools. https://github.com/decalage2/oletools.

Lang, J.,. Luckystrike. https://github.com/curi0usJack/luckystrike.

Lopera, D., 2020. Excel malspam: Password protected - not!https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/excel-malspam-password-protected-not/.

Malwarebytes,. The emotet malware. https://www.malwarebytes.com/emotet/.

Mendrez, R., 2015. Attackers concealing malicious macros in xml files. https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/attackers-concealing-malicious-macros-in-xml-files/.

Mendrez, R., 2020. Monster lurking in hidden excel worksheet. https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/monster-lurking-in-hidden-excel-worksheet/.

Microsoft, 2018a. About dynamic data exchange. https://docs.microsoft.com/en-us/windows/win32/dataxchg/about-dynamic-data-exchange.

Microsoft, 2018b. Compound files. https://docs.microsoft.com/en-gb/windows/win32/stg/compound-files.

Microsoft, 2020a. Compound file binary file format. https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cfb/53989ce4-7b05-4f8d-829b-d08d6148375b.

Microsoft, 2020b. Working with excel 4.0 macros. https://support.microsoft.com/en-us/office/working-with-excel-4-0-macros-ba8924d4-e157-4bb2-8d76-2c07ff02e0b8.

Microsoft, 2021. Excel (.xlsb) binary file format. https://docs.microsoft.com/en-us/openspecs/office_file_formats/ms-xlsb/acc8aa92-1f02-4167-99f5-84f9f676b95a.

Mimura, M., 2019. Using sparse composite document vectors to classify vba macros. In: International Conference on Network and System Security. Springer, pp. 714–720.

Mimura, M., Ohminami, T., 2019. Towards efficient detection of malicious vba macros with lsi. In: International Workshop on Security. Springer, pp. 168–185.

Morato, D., Berrueta, E., Magaña, E., Izal, M., 2018. Ransomware early detection by the analysis of file sharing traffic. Journal of Network and Computer Applications 124, 14–32.

Müller, J., Ising, F., Mainka, C., Mladenov, V., Schinzel, S., Schwenk, J., 2020. Office document security and privacy. 14th USENIX Workshop on Offensive Technologies (WOOT 20). USENIX Association https://www.usenix.org/conference/woot20/presentation/muller.

Nasi, E.,. macro_pack. https://github.com/sevagas/macro_pack.

Nissim, N., Cohen, A., Elovici, Y., 2016. Aldocx: detection of unknown malicious microsoft office documents using designated active learning methods based on new structural feature extraction methodology. IEEE Trans. Inf. Forensics Secur. 12 (3), 631–646.

Patsakis, C., Chrysanthou, A., 2020. Analysing the fall 2020 emotet campaign. CoRR abs/2011.06479.

Rudd, E.M., Harang, R., Saxe, J., 2018. Meade: Towards a malicious email attachment detection engine. In: 2018 IEEE International Symposium on Technologies for Homeland Security (HST). IEEE, pp. 1–7.

Security, F.,. Excelntdonut. https://github.com/FortyNorthSecurity/EXCELntDonut.

Singh, P., Tapaswi, S., Gupta, S., 2020. Malware detection in pdf and office documents: asurvey. Information Security Journal: A Global Perspective 29 (3), 134–153.

Team, M. D. R., 2018. https://www.microsoft.com/security/blog/2018/09/27/out-of-sight-but-not-invisible-defeating-fileless-malware-with-behavior-monitoring-amsi-and-next-gen-av/.

Vesselin Bontchev, P. L.,. pcodedmp. https://github.com/bontchev/pcodedmp.

Weber, M.,. Macrome. https://github.com/michaelweber/Macrome.

Yiu, V.,. Fsecurelabs. https://github.com/FSecureLABS/wePWNise.

Zhang, J., 2020. Velvetsweatshop: Default passwords can still make a difference. https://blogs.vmware.com/networkvirtualization/2020/11/velvetsweatshop-when-default-passwords-can-still-make-a-difference.html.

Vasilios Koutsokostas is an undergraduate student at the Department of Computer Science of University of Piraeus. He is a cybersecurity enthousiast with a special research focus on web security, exploit writing, and cybercrime detection.

Nikolaos Lykousas is a Ph.D. student at the University of Piraeus studying deviant behavior in modern Social Networks. He received his Master's degree in Intelligent interactive Systems from Universitat Pompeu Fabra in 2017, where he was awarded with an academic excellence scholarship for his achievements, after receiving his B.S. degree from the Department of Informatics at the University of Piraeus, in 2016. Since his undergraduate studies, he has participated as a research engineer in several EC funded projects and has gained considerable experience in the fields of Big Data Analytics, Cybersecurity, Digital Privacy and Cloud computing.

Theodoros Apostolopoulos holds a B.Sc. in Mathematics from the University of Athens, Greece and an M.Sc. in Advanced Computing Systems from University of Piraeus. He is currently a Ph.D. candidate at the University of Piraeus working in the fields of security, malware detection, and cybercrime.

Gabriele Orazi obtained a B.Sc. in Computer Science at the University of Bologna, Italy and an M.Sc. in Cyber Security with a double degree program from both University of Trento, Italy and University of Turku, Finland. He is currently a Ph.D. candidate at the University of Padua, where he is working as a Predoctoral researcher in the Department of Mathematics. His interests are in the security and privacy field, specifically focused in malware detection, penetration testing and web security.

Amrita Ghosal is currently a Marie Skodowska-Curie fellow in the Department of Electronic and Computer Engineering at the University of Limerick, Ireland. She obtained her Ph.D. degree in Computer Science and Engineering from Indian Institute of Engineering Science and Technology, India in 2015. After her Ph.D., she was Postdoctoral researcher in the Department of Mathematics at the University of Padua, Italy. Her research interests are in the areas of security and privacy for mobile and wireless networks. Particularly, she is interested in detection, prevention and mitigation of different DoS style attacks for smart grid, v2x, connected vehicle, cyber-physical systems, and IoT. In these areas, she has published more than 35 papers in high quality journals and refereed conference proceedings. She also has co-authored a number of book chapters.

Fran Casino is a postdoctoral researcher in Athena Research Center and in the Department of Informatics at Piraeus University (Piraeus, Greece). He obtained his B.Sc. degree in Computer Science in 2010 and his M.Sc. degree in Computer Security and Intelligent Systems in 2013, both from Rovira i Virgili University in Tarragona, Catalonia, Spain. He received a PhD in Computer Science from the Rovira i Virgili University in 2017 with honours (A cum laude) as well as the best dissertation award. He was visiting researcher in ISCTE-IUL (Lisbon-2016). He has participated in several European-, Spanishand Catalan-funded research projects and he has authored more than 50 publications in peer-reviewed international conferences and journals. His research focuses on pattern recognition, and data management applied to different fields such as privacy and security protection, recommender systems, smart health, supply chain, and blockchain.

Mauro Conti is Full Professor at the University of Padua, Italy. He is also affiliated with TU Delft and University of Washington, Seattle. He obtained his Ph.D. from Sapienza University of Rome, Italy, in 2009. After his Ph.D., he was a Post-Doc Researcher at Vrije Universiteit Amsterdam, The Netherlands. In 2011 he joined as Assistant Professor the University of Padua, where he became Associate Professor in 2015, and Full Professor in 2018. He has been Visiting Researcher at GMU, UCLA, UCI, TU Darmstadt, UF, and FIU. He has been awarded with a Marie Curie Fellowship (2012) by the European Commission, and with a Fellowship by the German DAAD (2013). His research is also funded by companies, including Cisco, Intel, and Huawei. His main research interest is in the area of Security and Privacy. In this area, he published more than 350 papers in topmost international peer-reviewed journals and conferences. He is Area Editor-in-Chief for IEEE Communications Surveys & Tutorials, and Associate Editor for several journals, including IEEE Communications Surveys & Tutorials, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, and IEEE Transactions on Network and Service Management. He was Program Chair for TRUST 2015, ICISS 2016, WiSec 2017, ACNS 2020, and General Chair for SecureComm 2012, SACMAT 2013, CANS 2021, and ACNS 2022. He is Senior Member of the IEEE and ACM. He is member of the Blockchain Expert Panel of the Italian Government. He is Fellow of the Young Academy of Europe.

Constantinos Patsakis holds a B.Sc. in Mathematics from the University of Athens, Greece and a M.Sc. in Information Security from Royal Holloway, University of London. He obtained his Ph.D. in Cryptography and Malware from the Department of Informatics of University of Piraeus. His main areas of research include cryptography, malware, security, privacy, and cybercrime. He has participated in several national (Greek, Spanish, Catalan and Irish) and European R&D projects under the FP7 and H2020 framework. Additionally, he has worked as researcher at the UNESCO Chair in Data Privacy, Trinity College, Dublin Ireland and the Luxembourg Institute of Science and Technology. Currently, he is Associate Professor at University of Piraeus and adjunct researcher of Athena Research and Innovation Center.