Analysis and Correlation of Visual Evidence in Campaigns of Malicious Office Documents

FRAN CASINO, Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili and Information Management Systems Institute of Athena Research Center
NIKOLAOS TOTOSIS, Hatching
THEODOROS APOSTOLOPOULOS and NIKOLAOS LYKOUSAS, Department of Informatics, University Piraeus
CONSTANTINOS PATSAKIS, Department of Informatics, University Piraeus and Information Management Systems Institute of Athena Research Center

Many malware campaigns use Microsoft (MS) Office documents as droppers to download and execute their malicious payload. Such campaigns often use these documents because MS Office is installed on billions of devices and that these files allow the execution of arbitrary VBA code. Recent versions of MS Office prevent the automatic execution of VBA macros, so malware authors try to convince users into enabling the content via images that, e.g., forge system or technical errors.

In this article, we propose a mechanism to extract and analyse the different components of the files, including these visual elements, and construct lightweight signatures based on them. These visual elements are used as input for a text extraction pipeline which, in combination with the signatures, is able to capture the intent of MS Office files and the campaign they belong to. We test and validate our approach using an extensive database of malware samples, obtaining an accuracy above 99% in the task of distinguishing between benign and malicious files. Furthermore, our signature-based scheme allowed us to identify correlations between different campaigns, illustrating that some campaigns are either using the same tools or collaborating between them.

CCS Concepts: • **Security and privacy** → **Malware and its mitigation**; **Phishing;**

Additional Key Words and Phrases: Malware, Microsoft Office, machine learning, VBA, phishing, macro malware

ACM Reference format:
Fran Casino, Nikolaos Totosis, Theodoros Apostolopoulos, Nikolaos Lykousas, and Constantinos Patsakis. 2023. Analysis and Correlation of Visual Evidence in Campaigns of Malicious Office Documents. *Digit. Threat.: Res. Pract.* 4, 2, Article 26 (August 2023), 19 pages.
https://doi.org/10.1145/3513025

This work was supported by the European Commission under the Horizon 2020 Programme (H2020), as part of the projects *CyberSec4Europe* (Grant Agreement no. 830929) and *LOCARD* (Grant Agreement no. 832735). F. Casino was supported by the Beatriu de Pinós programme of the Government of Catalonia (Grant No. 2020 BP 00035).
Authors' addresses: F. Casino, Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Tarragona, Spain and Information Management Systems Institute of Athena Research Center, Greece; email: franciscojose.casino@urv.cat, N. Totosis, Hatching, Netherlands; T. Apostolopoulos and N. Lykousas, Department of Informatics, University Piraeus, 80 Karaoli & Dimitriou str, Piraeus 18534, Greece; C. Patsakis, Department of Informatics, University Piraeus, 80 Karaoli & Dimitriou str, 18534 Piraeus, Greece and Information Management Systems Institute of Athena Research Center, Greece; email: kpatsak@unipi.gr.

2576-5337/2023/08-ART26 $15.00
https://doi.org/10.1145/3513025

1 INTRODUCTION

While cybercrime has always been a threat, it has evolved into a multi-billion underground economy over the past few years. The economic impact of cybercrime [18, 28] is so is devastating that according to the World Economic Forum considers it the second most-concerning risk for global commerce over the next decade [13]. Moreover, the recent COVID-19 pandemic and the spike in usage of digital services has also resulted in an analogous increase of cybercrime activities as reported by multiple sources.[1]

As cybercrime evolves in terms of scale and sophistication, **artificial intelligence** (**AI**) helps resource-intensive security operations by using technologies such as machine learning, pattern recognition, and natural language processing, which are capable of ingesting terabytes of unstructured data to enhance response times and expand the capacities of security operations. Nevertheless, attackers tend to be a step ahead due to the continuous appearance of novel technologies, industrialisation processes, the difficulty of collecting data from different sources in orchestrated campaigns and timely detection, and the lack of proactive security mechanisms. Undoubtedly, beyond the underground economy exchange (drugs, trafficking etc.), a significant share of this impact stems from the exploitation of security issues that allow an adversary to monetise vulnerabilities, e.g., by injecting commands and manipulating a compromised system or network traffic, by performing extortions, and so on. Pervasive and sustained cyber-attacks could have a potentially devastating impact on national and international organisations, disrupting the operations of governments and businesses and the lives of private individuals [4].

One of the most used cybercrime-related activities is phishing, as it can be used to launch a series of attacks on the victims [1, 6]. The adversary tries to exploit the human factor by presenting an e-mail that looks benign, e.g., appears to originate from a trusted source or having a harmless attachment; however, the attachment has a malicious payload. While attaching an executable may provide the adversary with immediate access to the victim's machine, this method is not used a lot. The reason is that executables are most often blocked from mail servers and that users do not usually receive such content via e-mail. Therefore, the victim is less likely to receive it and open it. On the contrary, an e-mail containing a **Microsoft** (**MS**) Office Document, e.g., a Word document or an Excel spreadsheet, is more likely to be opened.

Motivation and Contribution: Malware packed in MS Office documents was quite common in the past as the embedded macros were automatically executed when the corresponding trigger is launched, e.g., document is opened/closed. To address this issue, recent versions of MS Office have macros disabled by default, reducing such attacks' success rate. However, the problem is not by any chance solved as repeatedly shown by the impact of the associated malicious campaigns.

In most malware campaigns that are based on malicious MS Office documents, the *mondus operandi* is quite typical. The adversary tries to trick the user into opening an MS Office Document that comes in an attachment or a link [29]. While the exploitation of **Dynamic Data Exchange** (**DDE**) may offer automatic code execution, in most campaigns, the malware authors opt for macros as several patches prevent DDE execution. Nonetheless, this choice requires the victim to accept the macro's execution in her device, as in most cases, this is disabled by default. Therefore, the adversary has embedded an image in the document, which is the only thing that the user sees and tries to mislead the victim and convince it to enable the content. The image in the bulk of the cases falsely states that either some technical error has occurred or the document's data is not accessible, and only by enabling the content it can be resolved, see Figure 1. Should the victim be tricked into enabling the content, a macro is activated, which executes a malicious payload (either contained in the file or more often downloaded) using some **Living Off The Land Binaries and Scripts** (**LOLBAS**).[2] Upon infection, the executed malware may proceed to its core operation. The steps described before are depicted in Figure 2.

[1] https://www.europol.europa.eu/newsroom/news/covid-19-sparks-upward-trend-in-cybercrime, https://www.interpol.int/en/News-and-Events/News/2020/INTERPOL-report-shows-alarming-rate-of-cyberattacks-during-COVID-19.
[2] https://lolbas-project.github.io/.

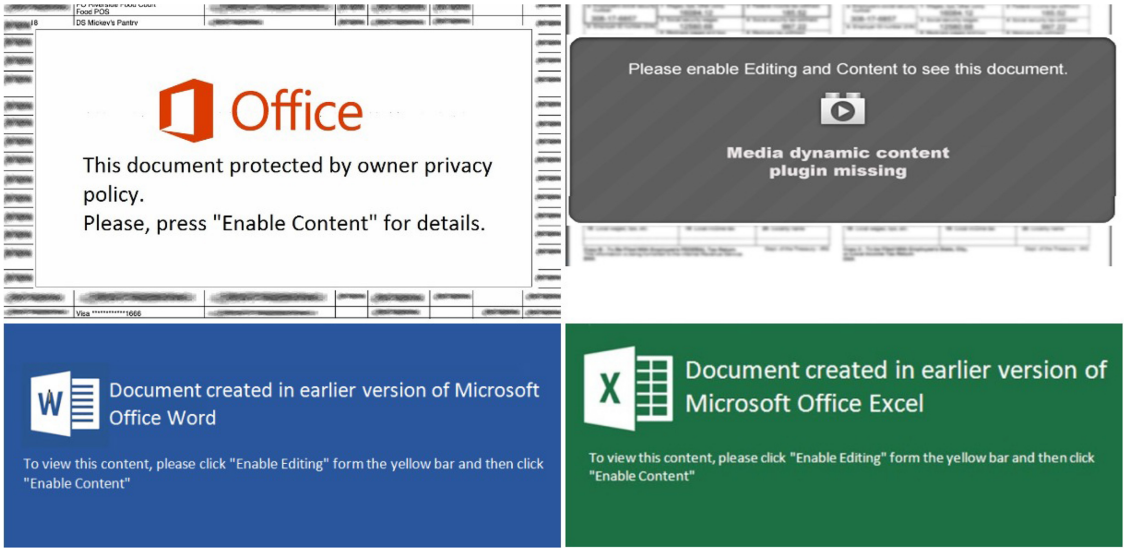


Fig. 1. Deceptive images in weaponized MS Office documents to convince the user to enable content.

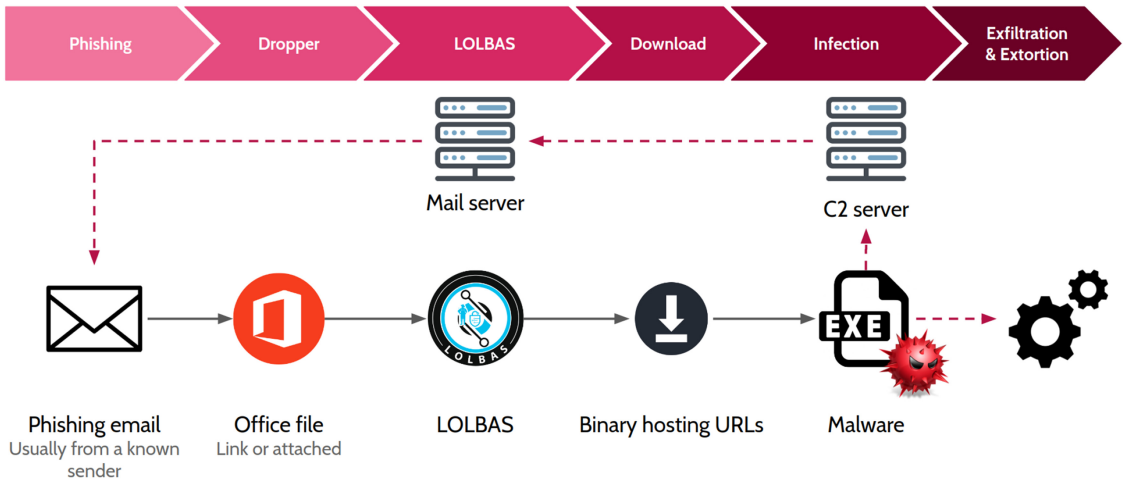


Fig. 2. Modus operandi of malware campaigns based on MS Office documents.

Based on the above, it is clear that one of the critical components in the attack is the image that is displayed to the victim. This work aims at investigating the possible correlations between malware campaigns based on this visual piece of evidence. To this end, we have collected an extensive and broad dataset which consists of more than 14 thousand malicious documents from 17 malicious campaigns. We extracted the embedded images, and we leveraged them to cluster the documents per malware and campaign. We argue that this approach may act as an **indicator of compromise** (**IOC**) and prevent many attacks from the mail server or the end-point, depending on where the mechanism is deployed. Thus, contrary to the state of the art methods that are based on natural language processing (see Section 2.2), we perform more lightweight calculations, e.g., compute the perceptual

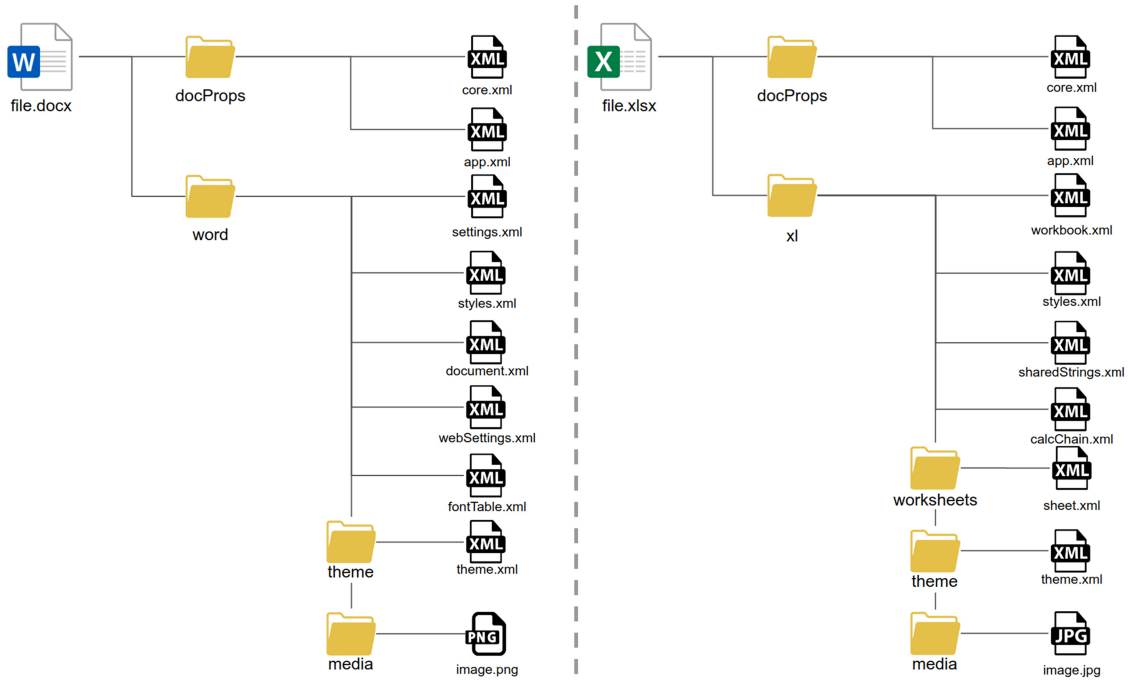


Fig. 3. Typical structures of a document and a spreadsheet OOXML files.

hash of an image and only detect the presence of VBA, XLM macros,[3] p-code, or DDE to determine whether a file belongs to known malware campaign. If the image is not known, we analyse it to extract relevant text known to be associated with malicious campaigns and flag it appropriately. As a result, we create a two-layer lightweight filter that efficiently identifies malicious MS Office documents and with high accuracy.

The rest of the manuscript is structured as follows. In the next section, we present the related work, giving an overview of the structure of MS Office document files, methods used to weaponise them and to detect them. Moreover, we present the core idea of perceptual hashing, which will be used in our work. In Section 3, we discuss the concept behind our approach and detail our methodology. Then, in Section 4, we present our experimental results. In Section 5, we discuss the outcomes of our experiments and our findings. Finally, the manuscript concludes summarising our contributions.

2 RELATED WORK

2.1 MS Office Document Files

MS Office is considered the default suite for writing documents, working with spreadsheets, and making presentations. To allow interoperability, MS has adopted the *Office Open XML*, also known as OpenXML or OOXML, format. As the name implies, it is an XML-based format for all office documents. The specification was developed by MS, has been adopted by ECMA International as (ECMA-376) [12] and became an ISO and IEC standard (ISO/IEC 29500) [17]. In principle, all OOXML files are stored in a compressed ZIP file. Therefore, each office file contains a set of XML files and stores the necessary files along with the schema. Images, audio or other multimedia files, as well as scripts stored in the document, are also stored inside the same ZIP file. The typical structure of an OOXML file is illustrated in Figure 3.

[3] https://support.microsoft.com/en-us/office/working-with-excel-4-0-macros-ba8924d4-e157-4bb2-8d76-2c07ff02e0b8.

Notably, several other formats are supported by MS Office for output, such as the **Compound File Binary Format** (**CFBF**), which are structured storage files.[4] In this regard, there are XLSB files supported by MS Excel. Due to their binary format, they are much faster than traditional XLS/XLSX files. The latter introduces a very interesting twist as files can be encrypted. Therefore, the content cannot be directly examined by an intermediate security mechanism. While Emotet has recently used encrypted ZIP files in its campaign [25], in this scenario, there is no need for a ZIP file as the office file is directly encrypted. In fact, this approach was recently used in several recent malware campaigns, which were further facilitated by an old Excel bug [32]. This bug allows Excel to automatically decrypt an Excel spreadsheet if the used password is *VelvetSweatshop*. Thus, the file was encrypted with this password, the security mechanisms could not scan the file, yet the file opened seamlessly in the user's device.

2.2 Malicious MS Office Files and Their Detection

MS Office documents are weaponised in a very straightforward way. Since most MS office support VBA code for macros and XLM macros, the most commonly used method to create a malicious document is add a macro to the document, which is executed automatically, e.g., upon opening or closing a workbook or document with `Workbook_Open()`, `AutoOpen()`, and `AutoClose()` functions. The code can then use the Shell command to execute a shell command or even download data from the Internet. Therefore, malicious MS Office documents are often used as droppers, that is, they download malicious binaries from the Internet and execute them, passing the control to them. Their authors obfuscate their code by adding unused code, base 64, hex and octal encoding, breaking strings into smaller ones or results of functions, and even abusing MS Office related functions to prevent their analysis.

A method to hide VBA code's execution is to destroy the VBA source code in the document but leave the compiled version of the macro code known as p-code. This method, known as VBA stomping, originally by V. Bontchev,[5] bypasses static analysers which try to extract the VBA code from a document. However, Office will execute the payload from the p-code [14]. Finally, one may inject a malicious payload using DDE exploiting several MS Office vulnerabilities linked to it, such as CVE-2017-8759, CVE-2017-11292, and CVE-2017-11826, or use XLM macros.

To detect malicious office documents, many researchers are using natural language processing methods [19, 20, 22, 31] to detect the presence of obfuscated code in VBA macros which is linked with the document being malicious. The XML structure can also serve as a good indicator of whether a file is malicious [9]. Due to the imbalances in the available datasets with such documents to train machine learning algorithms to detect such files, Mimura [21] recently proposed a method using Generative Adversarial Networks to generate fake samples with similar properties.

In another research line, researchers try to exploit n-grams [2] of the documents or other similar features such as entropy and other byte-level statistics over fragments of the data stream [26]. In [24], the authors have also exploited the more perplexing structure of the folder and files in the OOXML structure to classify documents as malicious. For more details on the threats from malicious documents [23] the detection of malicious documents, the interested reader may refer to [27]. Nonetheless, from the above, it is apparent that current state of the art methods are focused on binary classification (benign/malicious) and not the campaign's detection. Most of the checks are rather computationally intensive, as they require the extraction of several features.

2.3 Perceptual Hashing

Traditionally, hash functions are used to create an easy way to deterministically collect a small "sample" from a data stream that can be used to identify it and differentiate it from others with overwhelming probability.

[4] https://docs.microsoft.com/en-gb/windows/win32/stg/compound-files?redirectedfrom=MSDN.
[5] https://github.com/bontchev/pcodedmp.

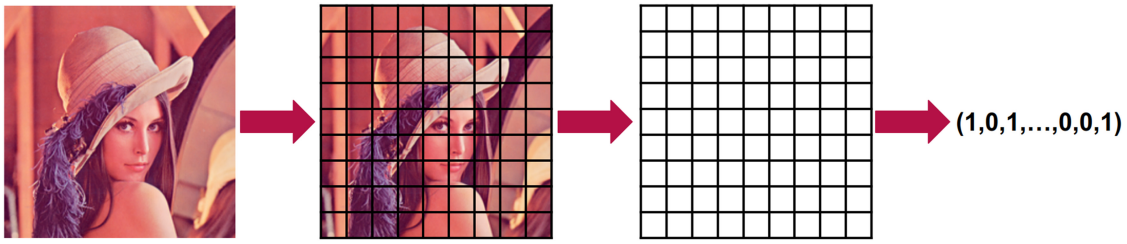


Fig. 4. Overview of how perceptual hashing works.

Nevertheless, we may tolerate slight variations for images as long as the actual content remains the same. In this regard, we want to create a hash that remains the same after minor image manipulations, e.g., rotation, cropping, or some light colour distortion. This type of hashing is called perceptual hashing, and it is widely used to facilitate tasks such as image search, retrieval, and authentication. Therefore, perceptual hashing is ideal for finding similar images see Figure 4.

Generally, we split the image on a grid and extract features from each segment. These are then transformed into a vector which is translated into the perceptual hash of the image. There are various methods for perceptual hashing, which in general fall into five main categories. According to Du et al. [11], these are Invariant feature transform-based methods, Local feature points based methods, dimension reduction based methods, Statistics features based methods, and Learning based methods.

3 METHODOLOGY

In what follows, we present the basic concept of our methodology and then our approach, consisting of our methodology and the collected dataset.

3.1 Concept

In many attacks, the adversary tries to create an asymmetry in the balance between his needed effort to launch an attack and the victim's effort to detect and mitigate it. For instance, in the scenarios that we are investigating, the attacker tries to create many documents with minor variations on the VBA code, some metadata, and text, which will lead to different hashes of the attachment. This way, one cannot simply blacklist a specific hash of a file or some code fragment.

While we understand that the latter task can be easily achieved by an adversary which already has many tools to achieve this, we argue that the same does not apply for the visual part of the document. For the execution of the VBA, the attacker has to convince the victim to enable the content. Therefore, random images, not clear, and without visual guidance to enable content would significantly decrease the attack's success. Based on the above, we argue that by detecting the visual part of the attack, we increase the adversary's effort, and we may create much more efficient filters. As a result, we aim at extracting the images that are contained in a document or a spreadsheet and correlate the information to (i) create more efficient IoCs, (ii) identify the use of common tools as well as cooperation between different malicious actors and campaigns. Therefore, our hypotheses are the following:

(1) The images contained in an MS Office document can be a good indicator that a file is malicious.
(2) Malicious campaigns and threat actors reuse images in campaigns due to the use of the same tools as well as cooperation.

The above introduce a strategy which consists of a set of steps that will be transformed into an algorithm in the next section. To realise this, we assume that we have already constructed a database that contains the hash of an image file (e.g., SHA-256), the perceptual hash of the image, and the verdict, which is either a malware family

ALGORITHM 1: Sample Classification Algorithm.

```
Input  : MS Office document D, Database DB, Threshold T
Output : Classification of D as malicious (family/generic), suspicious, or uncategorised.
1  Compute the hash of the document H;
2  if H ∈ DB then return tags(H);
3  if D contains dynamic features (VBA, DDE, XLM macros) then
4      Extract images from D;
5      tags = ∅;
6      for each extracted image img do
7          Compute hash h of img;
8          if h ∈ DB then
9              tags+ = tags(h);
10         else
11             Compute perceptual hash PH of img;
12             for each perceptual hash ph ∈ DB do
13                 if dist(PH, ph) < T then tags+ = tags(ph);
14             end
15         end
16         imgTXT = OCR(img);
17         if suspicious keywords sk ∈ imgTXT then tags+ = "Malicious";
18     end
19     if tags ≠ ∅ then
20         return tags
21     else
22         return Suspicious
23     end
24 else
25     Return Uncategorised
26 end
```

or a generic flag (malicious, unknown). Moreover, we assume a threshold T for clustering the perceptual hashes of the images.

First, we check whether the document has any dynamic feature (e.g., VBA, DDE, XLM macros) that can be used to execute malicious code. Then, we extract the image files from the MS Office document. For each extracted image, we check whether its hash is included in the database to retrieve the verdict (Algorithm 1, line 8). If this is not the case, we compute the perceptual hash of the image and compare it with the ones tagged as malicious or belonging to a specific malware family. Given threshold T, we classify the new perceptual hash in such cluster if the difference between the new value and the one existing in the database is below T (Algorithm 1, line 13). If the image does not belong in any such cluster, then we use OCR to determine whether there is any text included in a suspicious set of keywords, such as an *Enable content* suggestion. If this is the case, the sample is marked as malicious without any malware specific tag. If the new sample falls out of any previous categorisations and includes dynamic content, it is automatically marked as suspicious. In all other cases, the sample is labelled as uncategorised. This method is mapped in Algorithm 1.

4 EXPERIMENTAL RESULTS

In the following sections, we provide an exploratory analysis of our dataset as well as a description of the methods used to process the data and the corresponding outcomes.

Table 1. Composition of Our Dataset as of Samples Per Family

Malware Family	Samples
AgentTesla	396
Dridex	409
Emotet	4,268
formbook	467
Hancitor	53
IcedID	2,391
Loki	328
masslogger	26
Netwire	45
Qbot	3,699
Remcos	52
Smokeloader	998
TA505	201
TaurusStealer	30
TrickBot	227
Gozi_Isfb	212
ZLoader	729
Total	**14,531**

4.1 Dataset

To assess the applicability and efficacy of our hypotheses, we collected a large dataset of MS Office files from different sources to create a corpus of benign and malicious samples. In the case of malicious samples (14,531 in total), we collected files from Triage[6] and Malware Bazaar.[7] The Office documents belong to 17 different campaigns, as illustrated in Table 1, each of which uses different formats and attack methods.

After collecting these malicious samples, we exploited the fact that most office files are ZIP files, so the stored multimedia can be easily exported without tampering with the files and without opening the files in any virtual machine. Since several of the samples were XLSB files encrypted with the *VelvetSweatshop* password to avoid detection (see Section 2.1), we utilised msoffcrypto-tool[8] to decrypt the content and create a typical Excel file, without distorting its contents. This method is very efficient and lightweight; it guarantees that we are extracting the artefacts in a forensically secure manner and that no malicious code can be executed from the sample. Once the images have been collected from each sample, we compute its hash (using SHA-256), the perceptual hash of each extracted image, and then proceed with the text of the image. To this end, we detect the text's language, extract the text of the image, and translate it, where applicable. The extracted information is stored in a database and used for correlating, assessing, and clustering, as discussed in the following paragraphs see Figure 5.

In the case of benign samples, we randomly crawled different official public sites, namely governmental and educational sites belonging to different 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. In fact, we submitted these files to VirusTotal and Triage to validate that they are not malicious. After such a procedure, a total of 890

[6] http://tria.ge/.
[7] https://bazaar.abuse.ch/.
[8] https://github.com/nolze/msoffcrypto-tool.

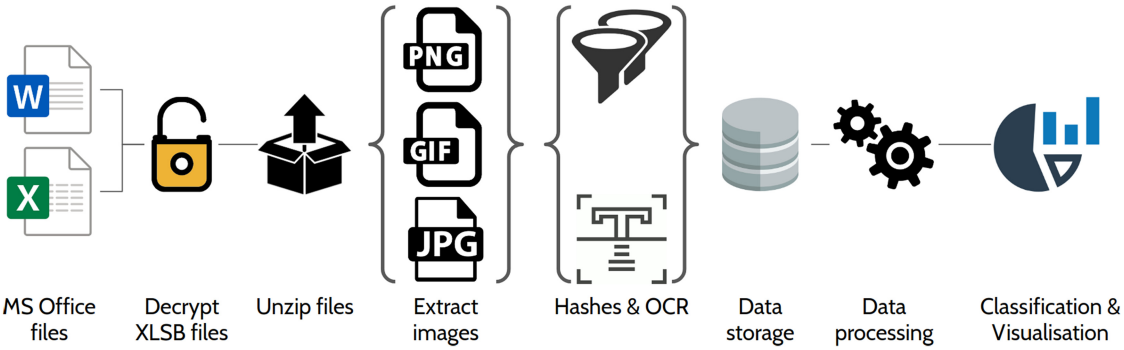


Fig. 5. Overview of our methodology.

benign samples were collected. Therefore, our database is created to exemplify the worst case scenario, in which we try to distinguish malicious MS Office files from benign files, in both cases using macros and DDE. For the reputability of our results and to advance the research in the field, the dataset is provided in Zenodo [7].

4.2 Exploratory Analysis

As described in Table 1, we collected samples from different malware families. Moreover, each of such samples contained a set of images. In this regard, Figure 6 depicts the number of images collected from each family. As it can be observed, Qbot, Emotet, and IcedID were the most populated families. It is relevant to note that some families used more images per sample on average. For instance, Emotet has more samples than Qbot in our dataset, yet the latter exhibited a higher number of images.

The following experiment focused on observing how many times the same image was used in the collected samples. In this regard, Figure 7 shows the number of times that an image was detected in our dataset, according to its SHA-256 hash. Clearly, there are two hashes, namely `49ad87680a...` and `3eb3cd078172...`, which appeared more than 2,000 times, the latter belonging mainly to Qbot and smoketbot, which used it exactly 3638 and 998 times. Moreover, they were used by ZLoader and Hancitor, yet only in very few samples. Therefore, these families used the same image multiple times in their campaigns. The next most used image appeared 100 times, and the pace decreased smoothly after that value. In total, we found 862 images appearing only once from the 1,488 unique images collected from our samples.

However, the latter does not depict the actual truth as malicious actors manipulate the images to have minor distortions, most of which do not have any observable change for the human eye. This way, even if the payload is the same, the resulting file has a different hash. Therefore, to correlate the collected images and bypass the slight modifications, we computed their perceptual hashes and leveraged the same experiments that we previously performed for the SHA-256 hashes. In this regard, Figure 8 shows the number of images and their appearance in our dataset. As it can be observed, many of them appear more than 100 times, contrary to the behaviour depicted in Figure 6. Moreover, the number of perceptual hashes appearing only once is 402 from a total of 733 unique perceptual hashes, which also showcases the strong similarity of some images and supports the use of perceptual hash in this context. In addition, we also depicted the number of images per family, according to their perceptual hash in Figure 9.

From the comparison between the statistics shown in Figure 6 and the ones represented in Figure 9, it is evident that several families used the same images across different campaigns due to the dramatic reduction of the number of images when removing duplicates. However, the set of unique images differs substantially between families. For instance, a considerable amount of Qbot, Smokeloader and Emotet samples were processed, yet such families tend to use the same subset of images in their corresponding samples, a fact which is evinced in Figures 9 and 8, and also by the number of images appearing only once (i.e., from 862 to 402).

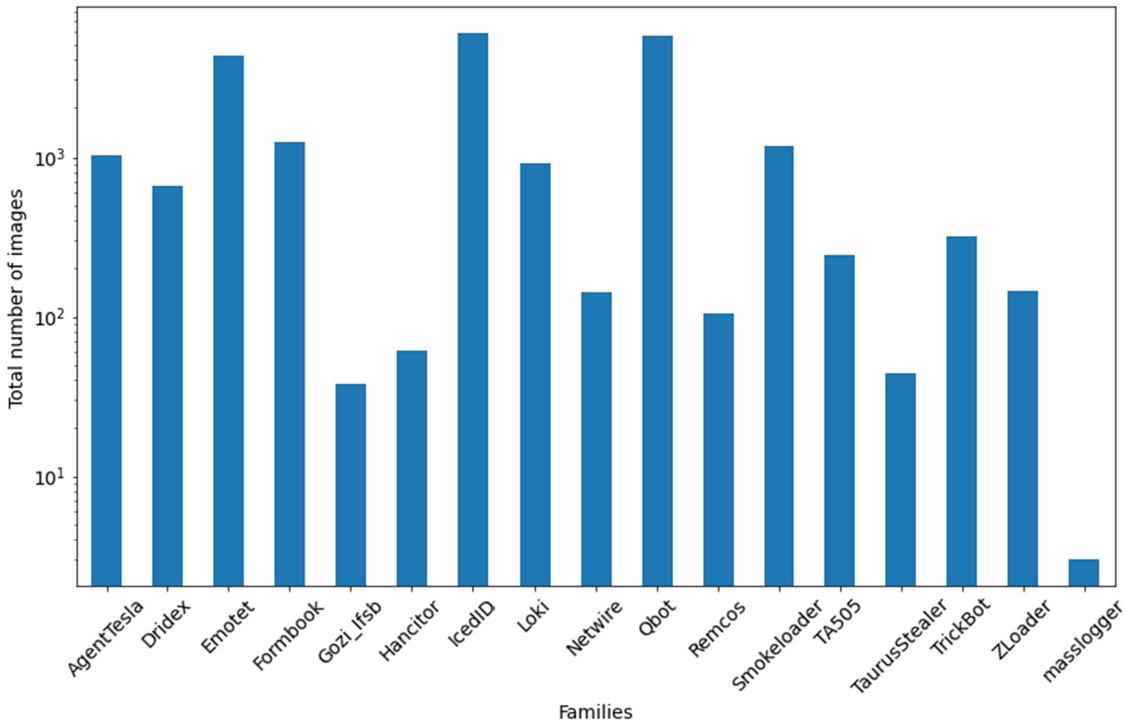


Fig. 6. Number of images retrieved from each corresponding family, according to unique SHA-256 values. Values are represented in logarithmic scale.

In the next paragraphs, we will explore the connections between the different families and the subsets of images used by more than one family.

4.3 Correlations between Different Campaigns

Figure 10 shows the number of times that a unique image was used by different families. The use SHA-256 reports a total of 32 different coincidences between families, while the use of perceptual hash reports 41. The latter means that some of the pictures were the same but minimal modification, trying to prevent the correlation of such campaigns. Note that the length of the perceptual hash value is 64 hexadecimal characters, and thus, using a smaller size could help discover further correlations. Nevertheless, the study of the perceptual hash variability and its impact on the correlations is left to future work since our setup already illustrates the relevance of using perceptual hash. Therefore, as it can be seen in Figure 10, the heat map is denser in the case of perceptual hashes (cf. Figure 10(b)), denoting that different families are using almost identical images with slight modifications, which are obvious enough to modify the SHA-256 representation of the image but not their perceptual hash. Note that some images differ only in some bits due to, e.g., a minor colour change in some pixels. Moreover, note that in Figure 10, we do not consider the repeated use of an image by different families, which would yield much higher numbers due to the use of the same or almost identical images in different campaigns.

According to Figure 10(a), the families that shared the highest number of unique images were Loki and AgentTesla, Formbook and Loki and Loki and Frombook. AgentTesla and Remcos are also using the same subset of images in several campaigns. These correlations are strengthened when using perceptual hashes to represent the images, as observed in Figure 10(b). In this regard, images that were slightly different were now captured and correlated between, for instance, Gozi_Isfb and Dridex, and ZLoader and TrickBot. In addition, we

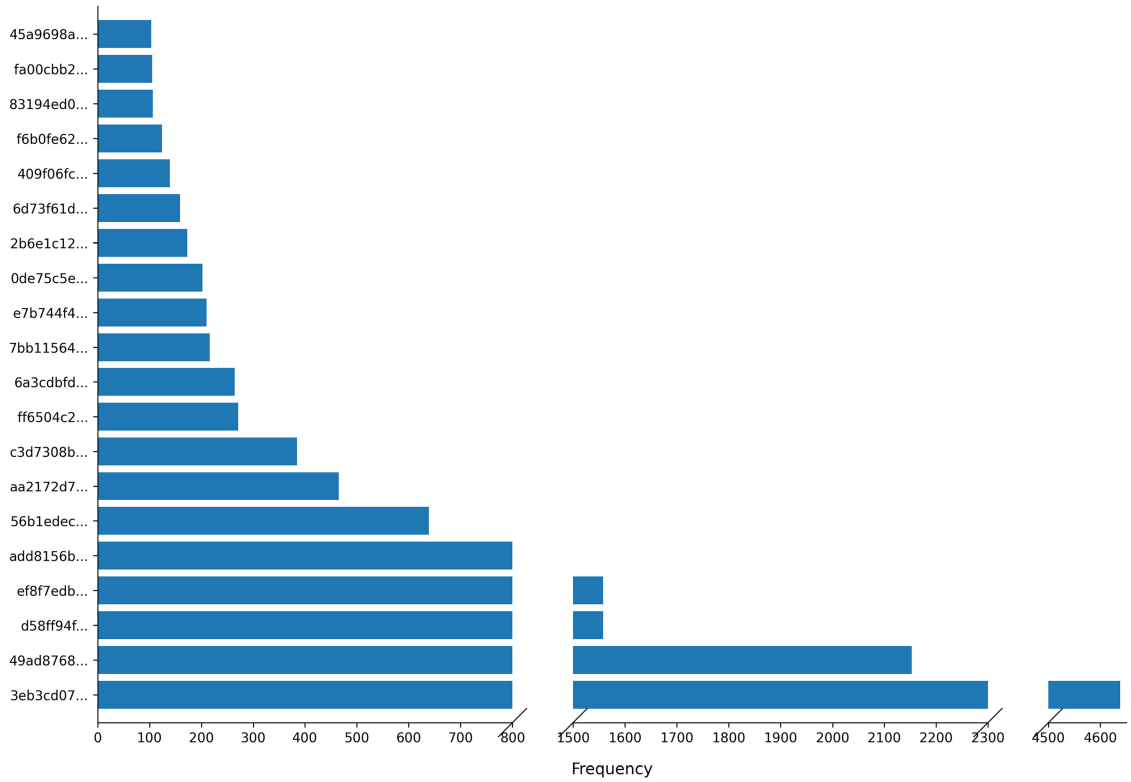


Fig. 7. Top number of images (according to their SHA-256) retrieved from each corresponding family.

discovered that very similar images were used by different families, showcasing the possibility that either the same perpetrators are behind different campaigns or that malicious actors are using previously published samples and materials to enhance their malware.

To consider the quantity of the coincidences between different images and families, we performed another measurement. In this case, we collected the images according to their perceptual hash, searched for how many different samples were used, and counted the co-occurrences between pairs of families. For instance, if $image_a$ appears in 10 samples of Qbot family and in 10 samples of Smokeloader family, we will count 20 co-occurrences between Qbot and Smokeloader. Note that we do not consider the direction of the co-occurrence, and thus the co-occurrences [Qbot, Smokeloader] and [Smokeloader, Qbot] are the same. The outcomes of this measurement are depicted in logarithmic scale in Figure 11.

As it can be observed, there are some families atop the number of co-occurrences, namely Qbot, ZLoader, Smokeloader. Moreover, IcedID, Trickbot, and Dridex also exhibited a high number of co-occurrences. Therefore, we can establish a high interconnection between such families considering both the number of unique hashes (cf. Figure 10(b)), and the number of samples in which they were used (cf. Figure 11).

It is worth noticing that, after establishing a connection between Qbot, ZLoader, and Smokeloader documents and upon closer inspection, we observed that apart from the image, they shared more characteristics. More precisely, the name of their first sheet is always `DocuSign`, and they usually have two more hidden sheets. One of these sheets has the XLM macros, and the other one the data used by these macros. Moreover, inspecting more samples revealed that for every different `DocuSign` image, there is a specific set of macros that accompanies it

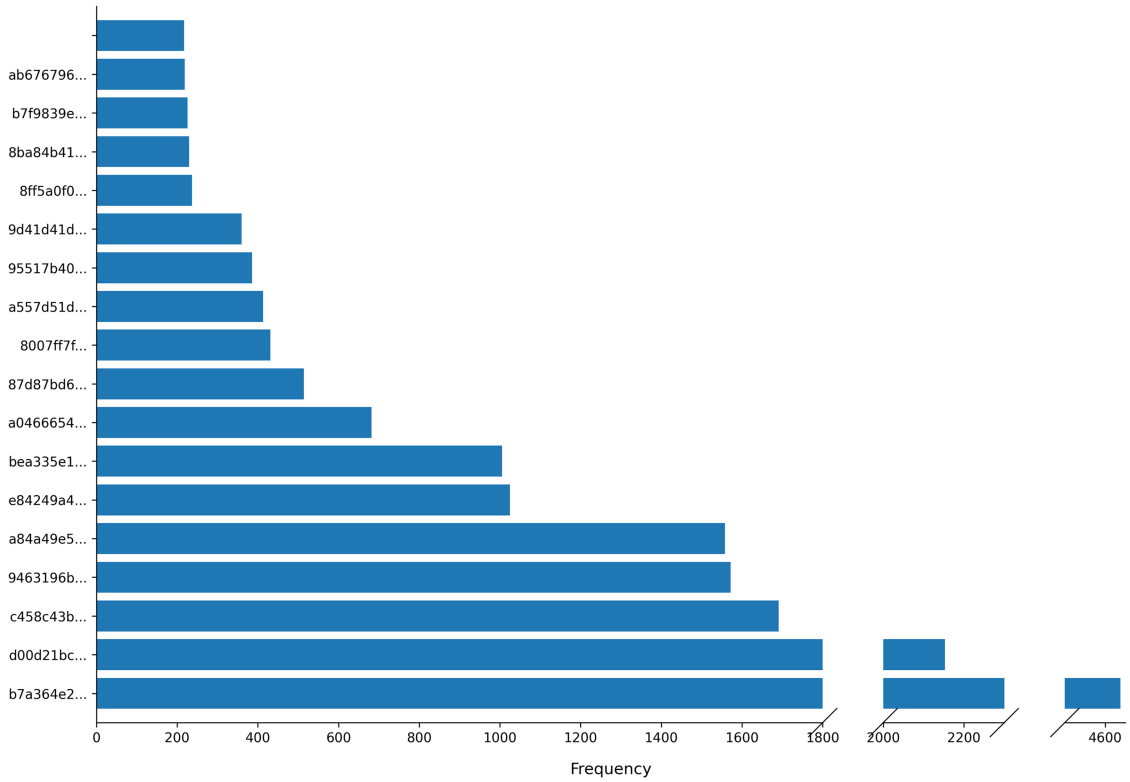


Fig. 8. Top number of images (according to their perceptual hash) retrieved from each corresponding family.

enables us to predict the behaviour of the document. The community has assigned the name `SilentBuilder` for these XLS(S|M) documents.

4.4 Visual Analysis of Samples and Comparison

To assess the limits of our hypothesis regarding the malicious intent of images, we opted to perform a *blind* test on images that are used in benign files. The blind test involves determining whether a file is benign or malicious based on its images and the fact that it has macros or DDE. While millions of MS documents are shared online, typical users do not use macros or DDE in their documents. Therefore, public samples of such documents are very sparse. Nevertheless, we collected 890 such documents from random sources, as stated in Section 4.1.

Following the methodology described in Section 3, we extracted 2,497 unique images from the 890 benign samples collected. To automatically detect the malicious intent of any image existing in a sample *blindly*, that is, without knowing any ground truth about the file, we leverage a text detection pipeline by using Tesseract.[9] The first step of our analysis consisted of manually annotating the images extracted from the malicious samples collected as reported in Table 1 that were asking the users to activate or enable content to grant access to a fully functional version of a document. Subsequently, we marked as malicious a total of 213 images and collected the corpus of words used to convince the users to enable the full functionality of the document, creating a vector of keywords of small size, since similar keywords were used across all samples (e.g., “enable content” or “enable macros”). Next, we used our text pipeline extraction method, which first applies a transformation to the input image

[9]https://github.com/tesseract-ocr/tesseract.

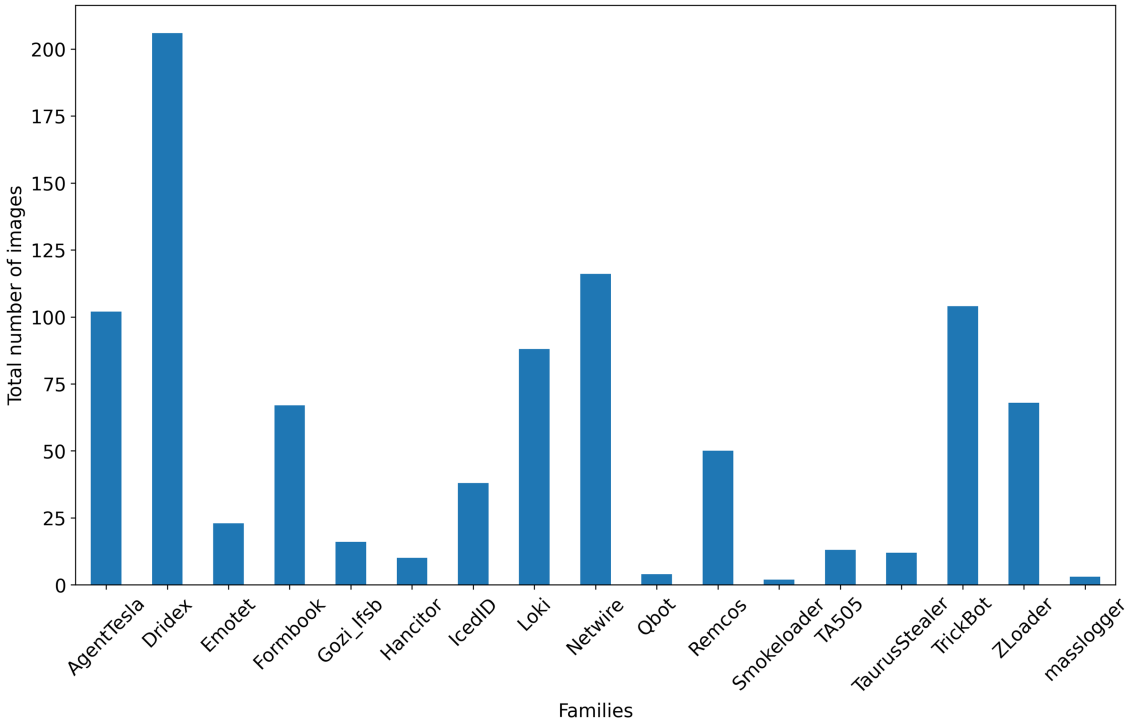
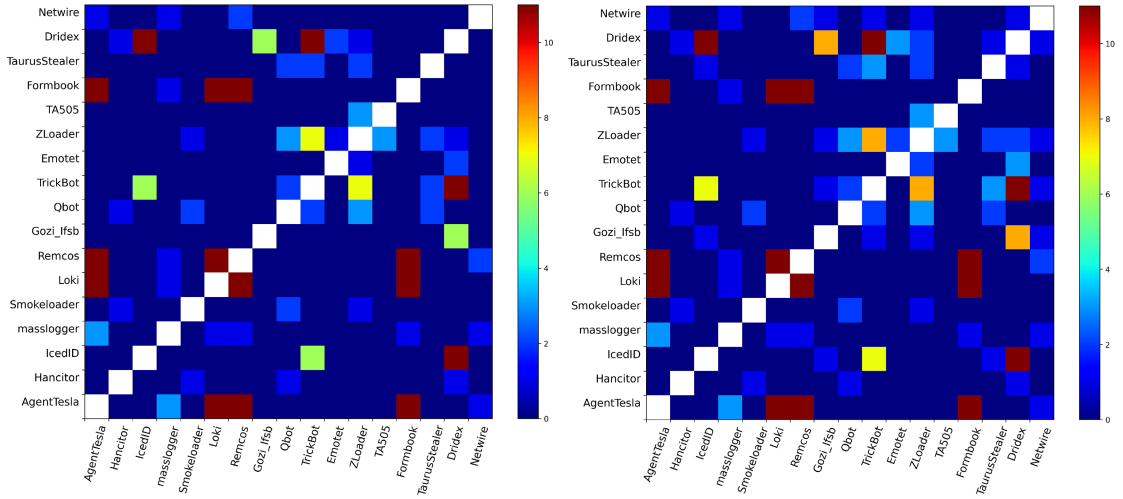


Fig. 9. Number of images retrieved from each corresponding family according to unique perceptual hash values.



(a) Heat map of the amount of unique images used by different families, according to the SHA-256 of the images.

(b) Heat map of the amount of unique images used by different families, according to the perceptual hash of the images

Fig. 10. Cross-family interactions according to the SHA-256 and the perceptual hash of the images collected in our dataset

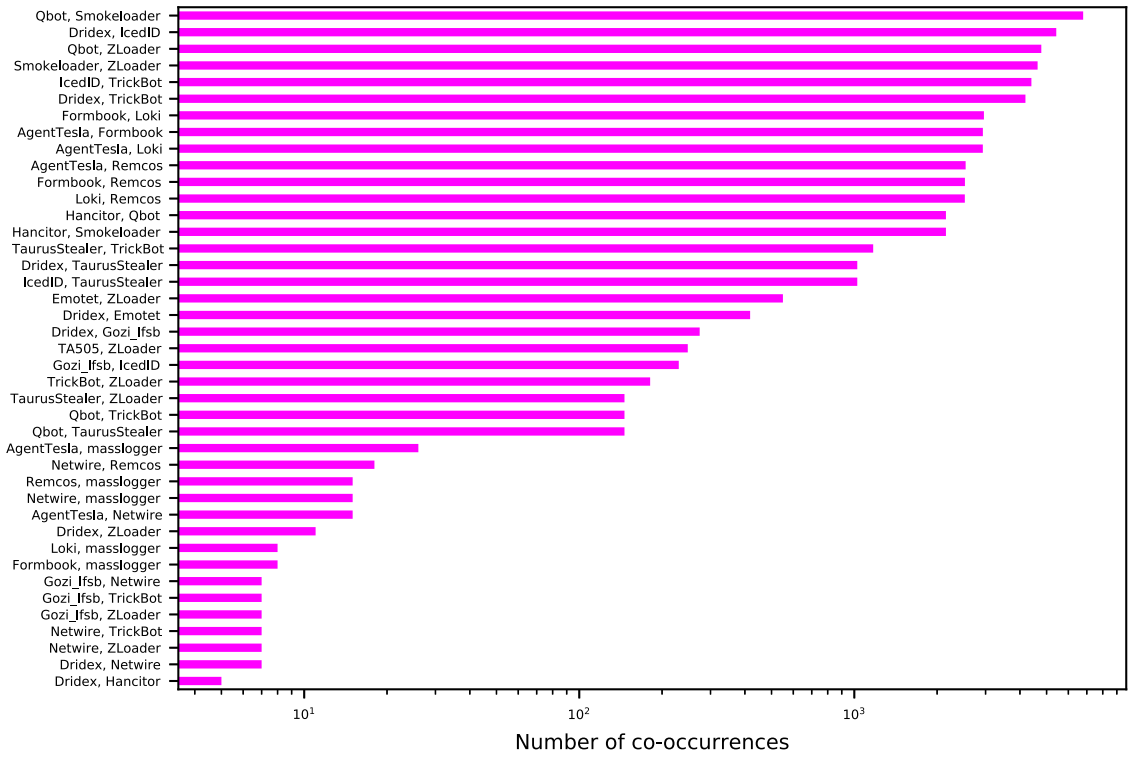


Fig. 11. Co-occurrences between families considering images and samples in logarithmic scale.

(i.e., removing the transparent layer and applying a threshold-based binarisation). Subsequently, we used Tesseract to extract the text of the image. The next step was to apply a lemmatisation of the words, enhancing the robustness of our method and avoiding that malicious actors can bypass our filters with slight modifications in text. Then, we translated the text to English where applicable to search for the specific keywords stated above. If an image contained one of the keywords tagged as malicious, we classified it as malicious, as explained in Section 3.

Given the previous classification criteria, we used our dataset (i.e., 2,497 benign images and 213 malicious images) to perform a binary classification experiment (i.e., into benign and malicious classes). Since our method does not require training, there is no need to split our dataset, and we used it all for testing. It is worth noting that the purpose of this experiment is twofold: (i) to test whether the use of our sample processing pipeline is able to extract the images and their corresponding text without errors, regardless of the language used and possible distortions (e.g., use of blurred text and/or overlapping transparencies designed by the malicious actors) and (ii) to test whether the selected keywords extracted from the malicious images collected from different campaigns are not used in the benign ones (which were collected from random websites) and thus, conform a robust feature for classification in a real-world setup. The outcomes of such binary classification are depicted in Table 2. Our accuracy is above 0.99, with only six false negatives. It should be highlighted that these errors were due to character recognition (e.g., "enabte content") as a result of overlapping transparencies or blurred low-resolution text. One solution for such a problem could be to, e.g., accept strings differing by one character, yet this kind of post-processing is left for future work. Note that, since the total amount of malicious files is 213, the amount of false positives has an impact in the precision value (i.e., the number of malicious files is less than 10% of the dataset). In the case of misclassified benign files, we had only 11 false positives. Such images contained several

Table 2. Outcomes of Our Proposed Method

Precision	Recall	Accuracy	F1-score
0.950	0.972	0.994	0.960

suspicious keywords, and 10 of them directly suggested users enable macros, yet these images were found in benign samples.

Note that such an experiment covers the worst-case scenario, and thus, our average outcomes could reach better values if we used, e.g., subsampling or n-fold splits. Nevertheless, we wanted to state the absolute numbers for the sake of clarity and to stress the efficacy of our method.

The predominant images that were found in our dataset are variants of MS Office-like text boxes asking users to enable content. Another subset of images used several combinations of colours, highlighted text and text with transparency to hinder the text detection task. A further significant subset of images is composed of images with blurred backgrounds, which creates the illusion that an authentic document will be unlocked if the user grants the corresponding permission. Finally, we also found other harmless images such as icons, business logos, and images belonging to step by step tutorials. It is worth noting that we found different languages corresponding to different international campaigns, including languages that used different scripts, including, but not limited to, Latin, Greek, Cyrillic, Bengali, Japanese.

Based on the above, our proposed method can achieve outstanding results even in the case of new unknown malicious campaigns. In fact, the benign dataset that we used cannot be considered representative of the real-world samples, as benign documents often do not use macros and DDE. Therefore, our *blind* test reflects the worst-case scenario of such an experiment. Notably, due to the way our method works, especially for known campaigns, fair comparison with the state of the art methods would not be fair. Firstly, our work goes further than merely labelling a document as benign or malicious by detecting the possible campaign, something that no current method is doing. Moreover, for known campaigns, the detection rate is 100%, which is something that no other method can claim since the visual elements can be accurately extracted and remain the same.

A proof of concept of our solution has been developed and is publicly available at http://blinddoctor.eu/. We have not integrated the OCR functionality yet in the current version, but it will be soon added. Nevertheless, to prove the efficacy of our approach, we extract the dynamic features that an MS Office document can have, also extracting some IOCs (e.g., URLs). Notably, although the service runs on a very limited server with one core and 4 GB RAM, it can correlate the information and accurately classify the malware sample on average at 4.57 sec.

Despite that there are no other methods that follow the methodology proposed in this article, and thus, direct comparison is infeasible, 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 3.

The method presented in [24] extracts and analyses 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 the validation experiments contains 16811 DOCX samples curated using VirusTotal, 16,484 of them benign, with less than 0.5% of these benign files containing macros. The above implies that the underlying dataset is highly unbalanced. The authors used different machine learning classifiers, namely J48, RF, LogitBoost, Logistic Regression, and SVM, and reported recall values of 0.93 in the best case with SVM. In addition, they obtained an accuracy value of 0.99. Yet, such value is hard to interpret since only 1.9% of the dataset consists of malicious files despite balancing the measurements. Unfortunately, the authors do not report the F_1-score to provide a more fair measurement and comparison with other studies. Moreover, while our approach 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 anymore. Many of the tools that are used nowadays do not generate such suspicious paths. In addition, weaponisation with DDE or VBA stomping may not generate any new path, deprecating such an approach.

Table 3. A Descriptive Comparison of Our Work with the Most Relevant State-of-the-Art Approaches

Reference	Dataset	Features	Method(s)	Outcomes
[24]	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	SVM obtained the best outcomes with a recall of 0.93 and an accuracy above 0.99.
[2]	40 malicious and 118 benign files created in Office 2010 retrieved from Contexture[10] and Payload Security[11]	n-gram analysis of the p-code of VBA	Term Frequency Inverse Document Frequency and KNN	Reported accuracy of 0.96.
[19]	Random collection of MS office files, curated according to VirusTotal analysis. A total of 2,537 files and 4,212 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, NB	The different machine learning approaches report accuracies around 0.9 in the task of identifying obfuscated macros. MLP was the most prominent with a 0.92 F_2-score.
[20]	7,145 samples including macros retrieved from VirtusTotal	Different language processing-related features, including SCDV, LSI, Doc2vec, Bag-of-words	SVM	The best F_1-score reported is 0.93.
Our approach	Benign samples with macros collected from official sites and malicious samples collected from Triage and Malware Bazaar, for a total of 890 benign and 14,531 malicious	perceptual hash and OCR recognition	Visual Analysis	Best F_1-score 0.96 and Accuracy above 0.99.

In [2], the authors collected a set of MS Office documents and used an automated procedure to extract the p-codes from each file and analyse them using Term Frequency Inverse Document Frequency to extract a set of lexicographical features. Next, the authors used a K-Nearest Neighbours algorithm to classify the samples, obtaining an accuracy of 0.96, below the one we obtained. Notably, the dataset used by the authors (only 40 malicious samples and 118 benign) is two orders of magnitude smaller than ours. Thus, the outcomes could vary if more samples were used, especially considering novel, sophisticated malware families.

In [19], the authors proposed 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 **Naive Bayes** (**NB**). The authors collected a total of 2,537 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 0.98, while MLP recorded a recall of 0.91. However, both LDA and NB 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.

The work in [20] 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. To detect malicious VBA macros, the authors proposed a method that extracts words from the source code and represents such VBA macros by using a language model that can be fed to the classifiers. The performance of such an approach was measured using 5-fold cross-validation, highlighting the accuracy of the Doc2vec model over the rest, with an F_1-score of 0.93.

It is clear that our method outperforms [2, 19, 20, 24] in all relevant metrics.

5 DISCUSSION

The evolution of cybercrime into a huge underground economy has turned cybercrime into an actual industry. The above is justified by the collaboration between malware authors and the emergence of *Malware-as-a-Service*

[10] https://www.contextures.com/excelfiles.html.
[11] https://www.payload-security.com.

(MaaS) or *Access-as-a-Service (AaaS)* models where, for instance, malware authors "rent" or pass the control of the compromised devices to their peers [5]. Moreover, for many malware families, the attribution of malware to an actor is not straightforward, and due to the malware evolution and code exchange in groups, it becomes a very challenging task. For instance, Gozi has several variations with different capabilities [8], with occasional parallel campaigns of its variants. Emotet, one of the most notorious malware, is another fine example of these exchanges. It shares the same loader with Gozi_Isfb, Dridex and BitPayme [30], it has bonds with Qbot [15], Trickbot [10], and more recently with Ryuk [16, 25]. Our experiments support the latter since we found correlations between such families in Section 4.3. More concretely, we found further correlations between families, especially in the case of, e.g., Qbot, ZLoader, Smokeloader, and Hancitor, which shared images across a high number of samples. Moreover, other families such as Dridex and Emotet also shared a relevant number of images between them and with further families such as ZLoader. Therefore, the information shared between families is higher than expected since other families like Loki and formbook were also correlated with, e.g., Qbot and ZLoader. In summary, macro malware campaigns share more similarities than one would envision, compared with other malware campaigns.

In the aforementioned campaigns, as well as the rest included in our dataset, the first step to launch the attack is made once the user opens an MS Office document and she enables the content. If the user is not convinced to enable the content, then the attack will not start. Therefore, the malicious actors try to present a convincing message that such action is necessary, such as e.g., a system error.

The current state-of-the-art methods, as analysed in Section 2 and [27] are simply trying to tag an MS Office document as malicious once a dynamic feature is found, obfuscation is detected, and/or presence of the exploit of a known vulnerability is detected. Thus, they are merely providing a binary classification which in a corporate environment may trigger many false alarms, as such MS Office documents are more dynamic to fit into templates, predefined structures, and guarantee proper information flow and input. Furthermore, the classification of an MS Office document to a specific malware family is mostly made through dynamic analysis, unless a specific static code-level feature can be recovered, e.g., hard-coded URL or use of the same macro. Thus, to classify an MS Office document to a specific malware family, one has to open it in a sandboxed environment that is time and resource-wise costly.

That said, endpoint security mechanisms do not support the classification of the malicious document in terms of family. Moreover, obfuscation, VBA-stomping, encryption, and other such methods allow them to bypass the checks of endpoint security mechanisms that are triggered once the document executes its malicious payload, which is in many cases too late. The latter has been repeatedly proved over the past few years with the success of malicious campaigns such as Emotet, IcedID, Dridex, and the like.

On the contrary, our proposed solution is very lightweight and accurate. Rather than resorting to dynamic analysis, one may extract the included images in the document to quickly assess whether the document is malicious and to which malicious campaigns they belong. Even if the document does not belong to a known malicious campaign, the document, due to its image elements, would be quickly flagged as malicious using OCR, preventing this way spear phishing attacks. It should be highlighted that these images are currently an indispensable part of such documents as the inherent security mechanisms of Office prevent such documents from automatically executing their code and are included to trick the user into enabling their execution. Since our detection method is based on perceptual hashes, it is positioned at the bottom of the pyramid of pain [3]. However, while such IOCs can be easily bypassed, e.g., in binaries with packers, the same does not apply for this specific IOC in its context. The differentiating factor is that these images cannot be arbitrarily manipulated. Heavily distorted images or ones that are not convincing enough, e.g., containing the proper logos and wording, would not lure the user in making the proper actions. Therefore, while the IOC can be easily changed, the resulting image would not be convincing for the victim. Moreover, the time needed to create many convincing such images introduces a disproportional effort to perpetrators.

6 CONCLUSIONS

Our proposed method is very lightweight as to determine whether a file is malicious, one uses a small signature which consists of the perceptual hashes of the images that it contains. If the image is in the database and the file contains VBA code, it is automatically flagged as malicious without the need to investigate the code. The latter can be easily checked with, e.g., the presence of the `vbaProject.bin` in the compressed files that comprise the document. Moreover, while the presence of obfuscated code implies that the document must be executed in a sandbox to determine in which family it belongs, our approach can classify it far easier by the hashes. If the perceptual hash does not exist in our database, we apply the method described in Section 4.4 to determine the threat level of the sample's images.

Perhaps the most significant contribution of our work is that our approach introduces a significant effort to malware authors. The images that they use are easily flagged, and creating new and convincing ones is far from trivial. This way, even if a sample contains images that are not known, one may easily introduce them to the blacklist in the presence of specific keywords. Indeed, the latter illustrated an almost perfect efficacy. In this regard, the effort of the adversary is significantly increased. The automatic use and minor tampering of images are prevented, and the generation of convincing images cannot be automated.

Finally, our work showcases the commonalities of malicious campaigns from another perspective. The existence of so many common images among campaigns shows that either the same tools are being used or that the same people are behind them, as the manipulation of the images cannot be the same for all of them. It should also be noted that some families, e.g., Emotet, are making each image unique, which signifies that they are aware that these images can serve as a signature, so each image has a unique hash.

Future work will focus on the forensic analysis of tools used to generate malicious MS documents and deobfuscation methods. To this end, tools like `Evil Clippy`[12] and `LuckyStrike`[13] will be examined, to determine their use in malicious campaigns. Moreover, we plan to examine further the attack surface that can be provided by an MS Office document as, e.g., the use of remote resources from XML has already been used to leak information, without even opening the file.[14]

ACKNOWLEDGMENTS

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

[1] Mamoun Alazab and Roderic Broadhurst. 2016. Spam and criminal activity. *Trends and Issues in Crime and Criminal Justice* 526 (2016), 1–20.

[2] Ruth Bearden and Dan Chai-Tien Lo. 2017. Automated microsoft office macro malware detection using machine learning. In *Proceedings of the 2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 4448–4452.

[3] David Bianco. 2013. The Pyramid of Pain. Retrieved from https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html.

[4] Fran Casino, Tom Dasaklis, Georgios Spathoulas, Marios Anagnostopoulos, Amrita Ghosal, Istvan Borocz, Agusti Solanas, Mauro Conti, and Constantinos Patsakis. 2021. A cross-domain qualitative meta-analysis of digital forensics: Research trends, challenges, and emerging topics. arXiv:2108.04634. Retrieved from https://arxiv.org/abs/2108.04634.

[5] Fran Casino, Nikolaos Lykousas, Ivan Homoliak, Constantinos Patsakis, and Julio Hernandez-Castro. 2021. Intercepting hail hydra: Real-time detection of algorithmically generated domains. *Journal of Network and Computer Applications* 190 (2021), 103135.

[6] Fran Casino, Nikolaos Lykousas, Vasilios Katos, and Constantinos Patsakis. 2021. Unearthing malicious campaigns and actors from the blockchain DNS ecosystem. *Computer Communications* 179 (2021), 217–230.

[7] Fran Casino, Nikolaos Totosis, Theodoros Apostolopoulos, Nikolaos Lykousas, and Constantinos Patsakis. 2021. MS Office decoy images. Retrieved from https://doi.org/10.5281/zenodo.5718684

[12] https://github.com/outflanknl/EvilClippy.

[13] https://github.com/curi0usJack/luckystrike.

[14] https://medium.com/@curtbraz/getting-malicious-office-documents-to-fire-with-protected-view-4de18668c386.

[8] Check Point Research. 2020. Gozi: The Malware with a Thousand Faces. Retrieved from https://research.checkpoint.com/2020/gozi-the-malware-with-a-thousand-faces/.
[9] Aviad Cohen, Nir Nissim, Lior Rokach, and Yuval Elovici. 2016. SFEM: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods. *Expert Systems with Applications* 63 (2016), 324–343.
[10] Cybereason. 2019. A One-two Punch of Emotet, TrickBot, & Ryuk Stealing & Ransoming Data. Retrieved from https://www.cybereason.com/blog/one-two-punch-emotet-trickbot-and-ryuk-steal-then-ransom-data.
[11] Ling Du, Anthony T. S. Ho, and Runmin Cong. 2020. Perceptual hashing for image authentication: A survey. *Signal Processing: Image Communication* 81 (2020), 115713.
[12] Ecma International. 2006. Office Open XML File Formats. Retrieved from https://www.ecma-international.org/publications/standards/Ecma-376.htm.
[13] World Economic Forum. 2020. Wild Wide Web Consequences of Digital Fragmentation. Retrieved from https://reports.weforum.org/global-risks-report-2020/wild-wide-web/.
[14] Carrie Roberts Harold Ogden, Kirk Sayre. 2018. VBA stomping: Advanced malicious document techniques. Retrieved from https://github.com/clr2of8/Presentations/blob/master/DerbyCon2018-VBAstomp-Final-WalmartRedact.pdf.
[15] Alex Ilgayev. 2020. An Old Bot's Nasty New Tricks: Exploring Qbot's Latest Attack Methods. Retrieved from https://research.checkpoint.com/2020/exploring-qbots-latest-attack-methods/.
[16] Intel 471. 2020. Understanding the relationship between Emotet, Ryuk and TrickBot. Retrieved from https://public.intel471.com/blog/understanding-the-relationship-between-emotet-ryuk-and-trickbot.
[17] International Organization for Standardization. 2016. Information technology – Document description and processing languages – Office Open XML File Formats – Part 1: Fundamentals and Markup Language Reference. Retrieved from https://www.iso.org/standard/71691.html.
[18] Internet Crime Complaint Center (IC3). 2019. 2019 INTERNET CRIME REPORT. Retrieved from https://pdf.ic3.gov/2019_IC3Report.pdf.
[19] Sangwoo Kim, Seokmyung Hong, Jaesang Oh, and Heejo Lee. 2018. Obfuscated VBA macro detection using machine learning. In *Proceedings of the 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 490–501.
[20] Mamoru Mimura. 2019. Using sparse composite document vectors to classify VBA macros. In *Proceedings of the International Conference on Network and System Security*. Springer, 714–720.
[21] Mamoru Mimura. 2020. Using fake text vectors to improve the sensitivity of minority class for macro malware detection. *Journal of Information Security and Applications* 54 (2020), 102600.
[22] Mamoru Mimura and Taro Ohminami. 2019. Towards efficient detection of malicious VBA macros with LSI. In *Proceedings of the International Workshop on Security*. Springer, 168–185.
[23] Jens Müller, Fabian Ising, Christian Mainka, Vladislav Mladenov, Sebastian Schinzel, and Jörg Schwenk. 2020. Office document security and privacy. In *Proceedings of the 14th USENIX Workshop on Offensive Technologies (WOOT 20)*. USENIX Association. Retrieved from https://www.usenix.org/conference/woot20/presentation/muller.
[24] Nir Nissim, Aviad Cohen, and Yuval Elovici. 2016. ALDOCX: Detection of unknown malicious microsoft office documents using designated active learning methods based on new structural feature extraction methodology. *IEEE Transactions on Information Forensics and Security* 12, 3 (2016), 631–646.
[25] Constantinos Patsakis and Anargyros Chrysanthou. 2020. Analysing the fall 2020 emotet campaign. arXiv:2011.06479. Retrieved from https://arxiv.org/abs/2011.06479.
[26] Ethan M. Rudd, Richard Harang, and Joshua Saxe. 2018. Meade: Towards a malicious email attachment detection engine. In *Proceedings of the 2018 IEEE International Symposium on Technologies for Homeland Security (HST)*. IEEE, 1–7.
[27] Priyansh Singh, Shashikala Tapaswi, and Sanchit Gupta. 2020. Malware detection in pdf and office documents: A survey. *Information Security Journal: A Global Perspective* 29, 3 (2020), 134–153. DOI: https://doi.org/10.1080/19393555.2020.1723747
[28] Douglas S. Thomas. 2020. Cybercrime Losses: An Examination of US Manufacturing and the Total Economy.
[29] Khoi-Nguyen Tran, Mamoun Alazab, Roderic Broadhurst, et al. 2014. Towards a feature rich model for predicting spam emails containing malicious attachments and urls.
[30] Trend Micro Research. 2018. Retrieved from https://www.trendmicro.com/en_us/research/18/l/ursnif-emotet-dridex-and-bitpaymer-gangs-linked-by-a-similar-loader.html.
[31] Muhammd Mudassar Yamin and Basel Katt. 2018. Detecting malicious windows commands using natural language processing techniques. In *Proceedings of the International Conference on Security for Information Technology and Communications*. Springer, 157–169.
[32] Jason Zhang. 2020. VelvetSweatshop: Default Passwords Can Still Make a Difference. Retrieved from https://blogs.vmware.com/networkvirtualization/2020/11/velvetsweatshop-when-default-passwords-can-still-make-a-difference.html/.

Received 30 April 2021; revised 23 November 2021; accepted 20 January 2022