Information Security Journal: A Global Perspective

ISSN: 1939-3555 (Print) 1939-3547 (Online) Journal homepage: www.tandfonline.com/journals/uiss20

Malware Detection in PDF and Office Documents: A survey

Priyansh Singh, Shashikala Tapaswi & Sanchit Gupta

To cite this article: Priyansh Singh, Shashikala Tapaswi & Sanchit Gupta (2020) Malware Detection in PDF and Office Documents: A survey, Information Security Journal: A Global Perspective, 29:3, 134-153, DOI: 10.1080/19393555.2020.1723747

To link to this article: https://doi.org/10.1080/19393555.2020.1723747

Published online: 13 Feb 2020.

INFORMATION SECURITY JOURNAL: A GLOBAL PERSPECTIVE
2020, VOL. 29, NO. 3, 134–153
https://doi.org/10.1080/19393555.2020.1723747

Malware Detection in PDF and Office Documents: A survey

Priyansh Singh[a], Shashikala Tapaswi[a], and Sanchit Gupta[b]

[a]Computer Science and Engineering, ABV- Indian Institute of Information Technology and Management, Gwalior, India; [b]Computer Science and Engineering, Scientific Analysis Group, Defence Research and Development Organisation, New Delhi, India

ABSTRACT
In 2018, with the internet being treated as a utility on equal grounds as clean water or air, the underground malicious software economy is flourishing with an influx of growth and sophistication in the attacks. The use of malicious documents has increased rapidly in the last five years along with a spectrum of attacks. They offer flexibility in document structure with numerous features for attackers to exploit. Despite efforts from industry and research communities, this remains a viable security threat. In this paper, a broad classification of malicious documents based attacks is provided along with a detailed description of the attack opportunities available using Portable Document Format (PDF) and Office documents. Detailed structures of both file formats, state of the art tools as well as the current research in automatic detection methods have been discussed.

KEYWORDS
Malware; PDF; Malicious Code; Detection; Cyber-Attack; Office Documents

1. Introduction

With the rapid infusion of internet-based software in everyday life, users have redefined how they use and perceive these services. They are willing to let these services facilitate work and personal relationships. Human nature inadvertently leads us to trust these services implicitly and end-up sharing personal information to that effect. This information is a valuable asset to marketing organizations, other companies as well as cybercriminals. Due to an implicit trust in the services on the web, users provide data to third-party services without hesitation or knowledge of how they manage this information. However, it is commonplace for these third-party services to sell information to other interested entities. Even if the software and services used are respectful of the user's data, there is always the probability of being on the wrong end of vulnerability or exploit. Cybercrime is profitable and requires little effort thus making users an easy and fruitful target. Cybercriminals and attackers are motivated financially, using any exploit they can gain access to a user's personal information. As a consequence of this, the cybercriminals push the envelope for evolution and variation of malicious software and other attacks propagated over the web.

Malicious Documents have been one of the growing methods used by attackers to propagate malware, all thanks to a growing unsuspecting audience and failure of detection by modern antivirus software. The recipients are motivated by the attacker to open the document through social engineering. Firstly, most users are not versed in the intricacies of malicious software and tend to execute links and documents they receive. Through osmosis, most recipients have absorbed executables downloaded from unrecognized sources may contain malware, but they do not hold the same depth of mistrust for non-executable documents. Secondly, most of the detection methods either use Signature-based or Heuristic-based methods for the detection of malicious software. It is known in the security community that signature-based detection can fail quickly and doesn't protect against new malware and zero-day vulnerabilities. Heuristic-based detection as described in Bazrafshan, Hashemi, Fard, and Hamzeh (2013) on the other hand use features like Application Programming Interface (API) calls, OpCode or Control Flow Graphs (CFGs) and machine learning models for detection. However, since most machine learning models get trained over a particular set of malicious software, an attacker can circumvent these safeguards to avoid detection.

CONTACT Priyansh Singh priyanshsingh@outlook.com

In part with websites and e-mail, document formats such as PDF and Office Open XML compose communication over the web. These file formats were created to provide a global platform that could represent the file on any platform irrespective of the hardware configurations. Due to their mass acceptance, these file formats have attracted the attention of attackers to propagate malicious code. All these file formats provide flexibility to the file structure and ability to embed content ranging from other documents, images, videos to scripting languages like JavaScript or Visual Basic. Attackers frequently use these properties for the diffusion of malware. On top of all this, the security community is still finding out critical vulnerabilities in common document readers like Adobe Acrobat Reader (CVE-2018-4895, CVE-2017-16398, CVE-2018-4917), Foxit Reader (CVE-2017-10994), and the whole of Microsoft Office Suite (CVE-2018-8162, CVE-2018-1030, CVE-2017-8550, CVE-2018-8157). This further concretes the fact, as stipulated by Kim, Hong, Oh, and Lee (2018) and Carmony, Zhang, Hu, Bhaskar, and Yin (2016), malicious documents remain a significant threat to users, especially over the internet.

With the advent of automation in malware creation, the generation of malicious programs is relatively rudimentary. With state-of-the-art tools like the Metasploit framework, an attacker can largely automate the generation of malware as well as include evasion countermeasures like obfuscation[1] or encryption. More can be read about the classification and functioning of such methods in Veerappan, Keong, Tang, and Tan (2018).

Social Engineering and e-mail phishing despite best efforts from the security community are still commonplace. It preys upon the gullibility and the lack of awareness of the user as mentioned in Wash and Cooper (2018). As discussed in Y. Cohen, Hendler, and Rubin (2018) malicious e-mail attachments are still at large, despite the strides made in spam detection and malware filters. In Sophos (2017), the authors mention the signs of a resurgence of Locky family of ransomware and its proliferation using spam e-mail, PDF files or a Microsoft Word document containing an embedded malicious macro.

This paper puts forward a survey of malware detection methodologies as pertaining to malicious documents. It discusses the attack surface and the opportunities a PDF or an Office document provides to the attacker. It provides a detailed summary of current challenges related to malware detection. To this effect, a broad cross-section of most relevant work was selected and analyzed. During the analysis, there was a particular focus on the underlying tools and scripts as research work may be directly dependent on these vulnerable tools and reader software. The article considers both static and dynamic analysis techniques along with decision paradigms such as signatures, heuristic and machine learning. The main novelty of the work comes from the first-ever classification and analysis of techniques and tools used for detection for malicious Office documents. The survey Nissim, Cohen, Glezer, and Elovici (2015) alludes to numerous academic contributions to PDF detection systems that take advantage of machine learning. When comparing to this work, it is believed that a more holistic approach would help further expand our understanding and find novel directions for future work.

The paper is organized as follows. Section 2 contains a brief explanation of broad attacks that are possible using malicious documents. Section 3, details numerous attacks that are possible with PDF format, tools that are available for forensic analysis along with a brief qualitative analysis of current research in detectors for malicious PDF files. Section 4 summarizes the format structure for Office formats along with state-of-the-art tools that are widely used for static and dynamic analysis. Section 4 also highlights the research pursued to detect the threat posed by Office file formats. Finally, Section 5 summarizes the observations made thus far, discusses future extensions and concludes the paper.

2. A Broad Classification of Malicious Document Attacks

As discussed in the previous section, all document formats are responsible for flexibility, complexity and variability of the attacks, effectively analyzing and recognizing such malicious documents is challenging. As described in Oracle (2015), there are two approaches to the document based attack (1) Structural attack using crafted content and (2) The exploitation of programming and interactive application features.

2.1. Structural Attack using Crafted Content

The structural attacks are used to target the software used to open the documents of a particular type. The attacker aims to change the structure of the document in order to cause an unexpected reaction from the software. Then piggybacking on this reaction the attacker can embed malicious code in the memory, for later execution. One conventional method to cause this unexpected behavior is to craft documents with values beyond the acceptable range, leading to a memory overflow. Return Oriented Programming (ROP)[2] attacks are common among structural attacks.

The trends in software engineering point to a more environment dependent development process, leading developers to use system libraries and APIs. Since these libraries are not in control of the developer and updated at a slower rate, they can be leveraged by the attacker to cause a structural attack.

Example: A text field could be a megabyte large without any logical reason and may contain malware code. A poorly written software could try to read the whole field into a smaller memory location causing a buffer overflow[3] and leading to heap spray.[4]

2.2. Exploitation of Programming and Interactive Application Features

Unlike the previous approach, instead of attacking the software used to open the documents, the attacker leverages the features available in the document structure and the software to their advantage. The core idea revolves around creating documents that uses interactive features, like embedding flash animations, ActionScripts or visual elements like video or audio, and programming features, like writing JavaScripts, Macros or Visual Basic Actions, to load and execute malware. These attacks can work in conjunction with the structural attack methods, and since these use features provided by the environment, the actions performed go unnoticed by the user.

Example: An attacker can create a document which uses the Visual Basic functions AutoOpen and Shell to launch a malicious shell script that was stored in the document. This launch behavior can be set up on various triggers and not just the opening of the file.

3. PDF Malware

The PDF format was developed in 1993 and was maintained by Adobe Software for the next thirteen years, is a common platform for transmitting malware as mentioned in Leurs (2017). One of the prime directives while developing this format was to make PDF files versatile so that sharing text formats, images and rich media like videos become software and hardware independent. In order to provide the aforementioned features, the architects at Adobe added programmable features. In 2008, the document format was standardized into an open format ISO 32000–1:2008. This change meant developers could implement the ISO (2008) standard in applications royalty-free. As discussed previously, attacks with malicious documents are versatile and effectively use the flexibility provided by the document structure to the full extent. Typically an amalgamation of both structural attack (embedded objects) and interactive feature (JavaScript or ActionScript) are used to target the reader software.

Malicious PDF files are very prevalent. During August 29, 2018, and September 4, 2018, Virustotal (2018), a free and unbiased appraiser of malware, received 380,453 PDF file submissions, fifth highest in the file type category. Cisco (2018), states the most commonly detected malicious file extensions worldwide during the period of January through September 2017. PDFs contributed to 14% of the file share, third highest among the group and trailing behind Office documents (38%) and archived files(37%).

Take for example, CVE-2018-4990, discovered in March 2018, where the exploit used an acrobat reader bug to trigger remote code execution in conjunction with MS Windows bug which leads to the privilege escalation of the execution. The JavaScript embedded in the malicious PDF file lays out memory for a heap spray. The acrobat reader had a bug in the JP2KLib.dll which has triggered using a malformed JPEG2000 file to cause an out-of-bounds access. As it can be seen from Program 1, the JavaScript that implements this attack is reported below. Note that the/Root object 5 is referring to object 6 with an/OpenAction tag.

Program 1 Code Excerpt from CVE-2018-4990

```
1 0 obj
≪ /Length 130 ≫
stream
function trigger(){
    var f = this.getField("Button1");
    if(f){
      f.display = display.visible;
    }
}
trigger();
endstream
endobj
…
…
5 0 obj
≪ /Outlines 2 0 R
/Pages 3 0 R
/OpenAction 6 0 R
/AcroForm 7 0 R
/Type /Catalog ≫
endobj
6 0 obj
≪ /JS 1 0 R
/Type /Action
/S /JavaScript ≫
endobj
```

During 2017 some exploits were discovered in the buffer overflow family (CVE-2017-2942, CVE-2017-2945, CVE-2017-2959). The characteristic of these exploits was the use of improper validation checks of unspecified inputs to cause a heap overflow condition. Once the attack is successful, the unauthorized remote attacker can perform code executions. Another method widely exploited is memory corruption. In CVE-2017-3010, where the rendering engine is corrupted using improper validation of input. The attacker can use this exploit for remote code executions or Denial-of-Service[5] attacks.

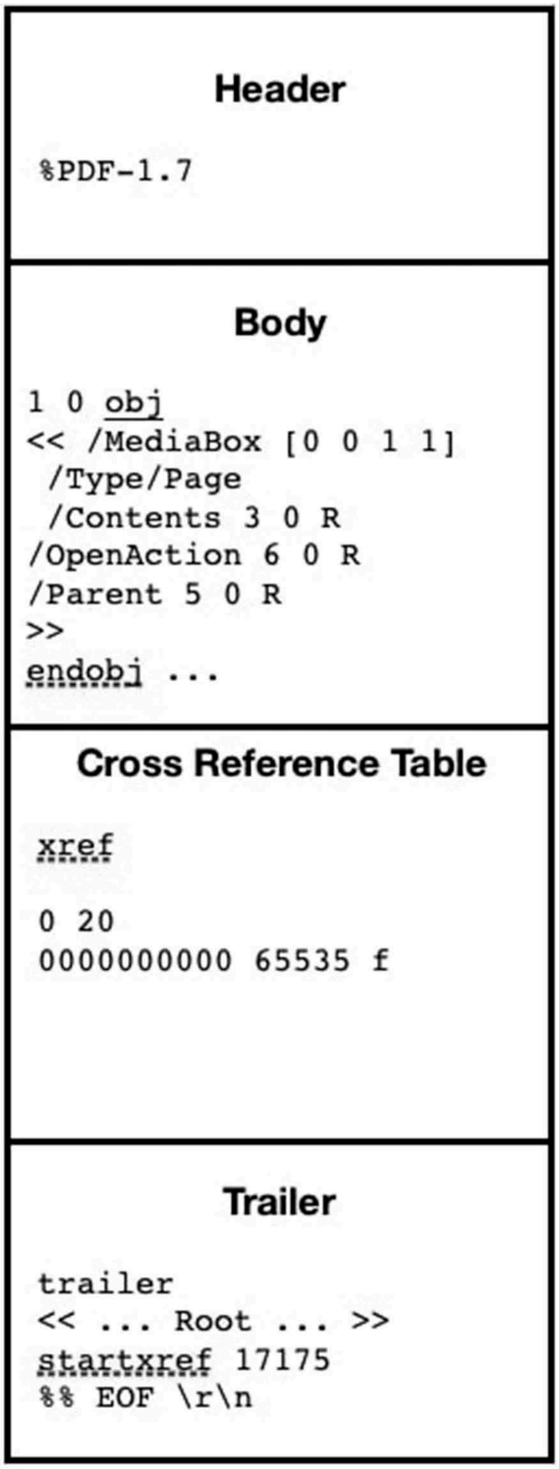
3.1. PDF Document Structure

Before diving deep into how various detection and analysis software work for PDF files there needs to be an understanding of the overall structure of the document. As seen in section 2 an attacker can hide and obfuscate malicious embeddings and scripts in a document, but the location of this content is unknown or how it can be extracted. The PDF standard was last modified in 2017, with the release of PDF 2.0 (ISO 32000–2:2017). The structure of a typical PDF file is as follows:

(1) **The header**: This contains the PDF version format as decided by the ISO standards body. For example, the header for a PDF 2.0 document would look like "%PDF-2.0."
(2) **The body**: This contains all the objects in a document which typically include, images, fonts and other multimedia and all the text streams. This section may contain additional features like security or animation. The body generally holds all the content displayed to the user. PDF can be updated after initial creation, in the process new body and x-ref sections are appended at the end.
(3) **The cross-reference (x-ref) table**: The PDF format allows the ability for incremental updates in the document. The presence of the x-ref table enables this feature. The table allows for random access of any object in the file. When the document is updated extra cross reference tables and trailers are appended at the end.
(4) **The trailer**: The trailer shows how the document viewer should render the file by pointing it to the sections such as cross-reference table and other objects. The trailer also contains the /Root tag which specifies where the document should be read from, /Size defining the size of the entry in the x-ref table. The last line of the file contains the end of file string '%%EOF'.

Figure 1 displays the structure of PDF and how when updated with new content; new sections get appended at the end. Typically a PDF file can be visualized as a directed graph, and section 3.2.1 refers to static tools that are used to create such graphs. This visualization is helpful for the user to understand how different parts of a PDF linked with each another.

The PDF file format supports eight objects: boolean, numeric, string, name, array, dictionary and streams. These are used to represent content in the PDF structure. Attackers can use these objects to hide their data in the form of obfuscated strings, array or mislabeled dictionary items.



```
%PDF-1.7
```

```
1 0 obj
<< /MediaBox [0 0 1 1]
 /Type/Page
 /Contents 3 0 R
/OpenAction 6 0 R
/Parent 5 0 R
>>
endobj ...
```

```
xref

0 20
0000000000 65535 f
```

```
trailer
<< ... Root ... >>
startxref 17175
%% EOF \r\n
```

Figure 1. Structure of PDF Document.

3.2. Detection and Analysis of PDF Malware

In the domain of cyber forensics, detection and analysis of malicious files are performed together. Detection of malware, for the most part, can be automated with the help of some signature-based, behavior-based and heuristic-based methods such as Bazrafshan et al. (2013) and Saeed, Selamat, and Abuagoub (2013). For an analyst to decide and unwrap a potential malicious document file, they need a list of steps to determine if a file is malicious and what the intended aim of the attacker is. Lucky for us, Zeltser (2017) and the SANS Institute provides this guideline.

(1) Examine the document for irregularities
(2) Find the location and extract of embedded suspicious code or objects
(3) Depending on the nature of the suspicious code,
(4) if it is a scripting language like JavaScript or VBA, deobfuscate and examine
(5) else, it is shellcode or portable executable then disassemble and debug

Prayudi et al. (2015) categorize analysis methods into, Static Analysis and Dynamic Analysis. An analyst employs both of these methods while trying to discern whether the document is malicious or not along with the intentions of the document.

3.2.1. Static Analysis

During static analysis, an analyst examines the potentially malicious file without ever executing it. They focus on the structure and the metadata stored in the document. The goal is to determine if a file is malicious without ever executing the code embedded in it. To this effect, some tools have been developed by the security community for the analysis of PDFs.

One of the most common and rudimentary analysis techniques is to look for keywords that may suggest malicious presence. More often than not, the analyst does not even need to decode or decompress the PDF document but only look at the strings present in the document for identification. The analysts are looking for keywords like, /JavaScript, /OpenAction, /GoTo, /URI and /RichMedia which are indicative of maliciousness. If keywords are not present, the analyst can confidently say the file under analysis is not malicious. PDFiD is one such tool; it performs a textual analysis of all the dictionaries present in the file. PDFid can even look at the components that may not be rendered by the reader software. However, it can be easily deceived by placing the JavaScript in object streams, where due to stream compression the keywords will not be available as plain text and thus hidden from the tool. These findings have been generalized by Prayudi and Riadi et al. (2015) for all basic static analysis methods. There are many other tools which can be used for this analysis, like AnalyzePDF.py, pdf-parser.py

As already discussed, PDF file can be visualized in a tree, with links showing the interconnection between various objects. Figure 2 gives an interpretation of such a graph using the pdfobjflow tool. PeePDF, Pdfobjflow and Origami are few of such utilities. The analysis revolves around looking for the /root location which is stored in the trailer. Iteratively follow all the references made by the /root object and add them to the tree. These tools give the analyst an opportunity to look at all the content streams and ascertain if any of them seem malicious. PeePDF can additionally be used to suspicious elements, extract metadata and various data streams.

3.2.2. Dynamic Analysis

During dynamic analysis, an analyst opens the potentially malicious file in a controlled environment to observe it's behavior. Using one of the tools like pdfextract, Origami or pdfstreamdumper, the analyst first can locate and extract the relevant object streams. The analyst examines the streams first for possible obfuscations and anti-debugger measures. In case there is a presence of such, they run deobfuscation and disassembly scripts. Once the malicious program is deobfuscated, the analyst can look for potential API calls and calls to various websites to get a better understanding of the malware. On the basis of knowledge-base, developed so far, analysis of malicious code is performed in a sandbox. There are numerous online sandboxes like MalDocs, Virustotal and hybrid-analysis, that provide dynamic analysis services. Analysts look for behavioral indicators of the malicious file, like trying to connect to the internet or launching subprocesses.

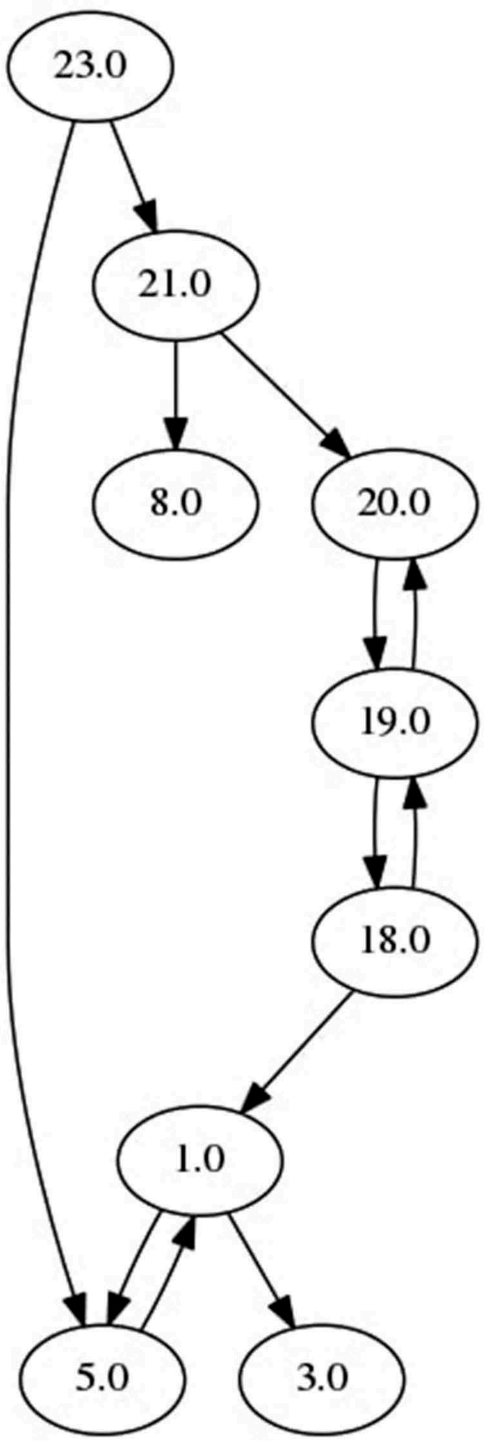


Figure 2. PDF file represented with a directed graph using Pdfobjflow.

PhoneyPDF and ParanoiDF are tools that perform dynamic analysis for PDF files with malicious JavaScript embedded in them. It detects and extracts JavaScript objects and executes them through a JS interpreter. It highlights the suspicious code for the analyst to verify.

3.3. Automated Detection of Malicious PDFs

Carmony et al. (2016) broadly categorize PDF detectors in three categories:

3.3.1. Signature Analysis & Pattern Matching

These methods try to model the malicious behavior of malware in order to use this model in the detection of malware. Signature is a unique feature for each family of malware. These models together form knowledge-base that is widely used by antivirus for detection. These detectors are faster and more efficient than other methods but are limited grossly by their knowledge base. They are especially vulnerable to zero-day vulnerabilities, that is an attack for which there is no corresponding signature stored in the repository.

Shafiq, Khayam, and Farooq (2008) use statistical anomaly using markov n-gram detector on the entire document. On the basis of characteristic change in the entropy, the markov n-grams, the detector classifies the malicious documents.

It is a widely known fact that signature-based PDF detectors like Lu, Zhuge, Wang, Cao, and Chen (2013), Schmitt, Gassen, and Gerhards-Padilla (2012) and antivirus programs are susceptible to various conventional polymorphism and metamorphism. Moser, Kruegel, and Kirda (2007) analyze few obfuscation methods and discuss how static signature-based detection methods are thwarted easily. Fogla, Sharif, Perdisci, Kolesnikov, and Lee (2006), Song, Locasto, Stavrou, Keromytis, and Stolfo (2010) and Fogla and Lee (2006) discuss the failure of signature-based PDF detectors due to polymorphism.

3.3.2. Metadata & Structural Features

This detection powered by machine learning models and focuses solely on the name objects present in a PDF document. Some of the methods use one or more of the static analysis tools that have already discussed for pre-processing. These methods are more generalized, as they also take into account different embedded programs (e.g. JavaScript or ActionScript) for the analysis.

Maiorca, Giacinto, and Corona (2012) proposes a metadata extraction approach using PDFid and Random forests for detection. It was soon found out that due to the inefficiency of the underlying extractor, the algorithm failed against polymorphic attacks. Maiorca, Ariu, Corona, and Giacinto (2015) use PeePDF for analysis of embedded files and Origami for integrity checks on file structure during the pre-processing stage. It can reveal embedded data streams as well as malformed files. It identifies set of most characteristic keywords found the during analysis of training set.

Unlike Maiorca et al. (2015), Šrndić and Laskov (2016) uses a library based parser Poppler, for preprocessing. It renders the file in question to examine it for metadata and structural faults. This works in four phases namely, Structural Path Consolidation, i.e. similar structural features are merged, Feature Selection, Vectorization, and Classification using random forests.

Smutz and Stavrou (2016) classifier use the ensemble learning model using a large number of decision trees to classify features extracted by its pre-processor. They extract PDF metadata like title and creation date and object properties like position and counts. Since it uses a custom parser for files extracting features, it does not need to follow the PDF specification. It can analyze the documents that may not be referred to in the x-ref table but are present, but at the same time, this can be used for evasion by malicious pdf files. It is noted in Falah, Pan, Abdelrazek, and Doss (2018) the accuracy of both Smutz and Stavrou (2016) and Maiorca et al. (2012) drop significantly when using different PDF standards.

Cuan, Aliénor, Delaplace, and Valois (2018) propose an SVM based classifier using Gaussian Radial Basis kernel function for malicious documents. The algorithm suggested in the research paper uses PDFiD to extract features mentioned by Didier Steven on their blog. Cuan et al. (2018) define a gradient-descent attack on the classifier, where the attacker adds benign features in order to fool the SVM classifier. They also propose feature-based and adversarial learning countermeasures, but more extensive studies have not been performed till now.

However, metadata and structural features based detection can be circumvented easily by desiring PDF files that appear similar to the benign files. Mimicry and reverse mimicry attacks are some such attacks. In the case of mimicry attacks, the attacker injects benign content in maliciously classified files. These attacks are performed with the hope that the probability of detection of a file would decrease. Conversely, in reverse mimicry attacks, the attacker injects malicious content in a benign file. Maiorca and Biggio (2017) have addressed these attacks on detection methods in a systematic way.

M. Xu and Kim (2017) propose a completely different technique with the stipulation that malicious documents present different behavior on different platforms. Their detection technique is based around platform disparity and the assumption, that most benign files would show similar behavioral indicators irrespective of the platform whereas malicious files are designed to attack a particular platform and thus shows disparity when examined on non targeted platforms. For detection, the suspected file is executed over multiple platforms while behavioral data is collected. The detection technique requires at least two virtual machines and may be susceptible to attacks that require user input.

3.3.3. Javascript Features

In recent years, embedding JavaScript code has become one of the most common methods to create effective malicious documents. Most often the target of the embedded JavaScript is either executed when the document is opened or stored in memory waiting for an opportune moment for execution. As seen in code segment 1 covers both these cases, i.e. how the JavaScript was opened at the beginning in order to prepare for code execution using the heap spray later. JavaScript code can either be extracted statically or dynamically; in the former case, any one of the tools previously discussed can be used, while in another case, file needs to parse in a sandboxed environment or with the use of rendering libraries.

Take the example of Corona, Maiorca, Ariu, and Giacinto (2014); it relies on PhoneyPDF for emulation of the file in Adobe DOM. PhoneyPDF keeps track of all the API references made by the file during static and dynamic analysis. Just like any other machine learning-based detector, Lux0r performs, API reference extraction, selection of relevant features and classification using random forests. They also discuss the vulnerability of the detector to mimicry attacks and how it can be somewhat contained using an adversarial learning approach.

Cova, Kruegel, and Vigna (2010) define ten distinct features by which it classifies a document as malicious. It dynamically analyzes the embedded JavaScript using JSand, then extract the exploit features and uses a Bayesian classifier to link feature values to one of the exploit classes with high confidence. Laskov and Šrndić (2011) extracts JavaScript using the libpdfjs, which is an extension of the third-party parser Poppler and uses a one class SVM for classification.

Li et al. (2017) presents a feature extraction method and compares to widely used feature extractors as Poppler, iText, JSUNPACK and PdfJsExtractor. This methodology takes into account for Header type, Refer-Chain, Hex-Digit, Compression, Redundant Information, Invalid-Obj, Xref feature, and Trailer feature. This system claims a precision of 97.57% and an accuracy of 95.11%.

Snow, Krishnan, Monrose, and Provos (2011) propose a method of detection using hardware visualization and design a new kernel ShellOS. The dynamic detection involves executing the suspicious file directly and inspecting buffers for the malicious instruction set. They tested this technique on PDF documents executing shellcode through javascript, but it can be extended to any code injection attack.

An active learning approach by Nissim et al. (2016), where the detection method worked in conjunction with antivirus to acquire both new PDF signatures for both benign and malicious documents. The acquired files were used to enhance the knowledge base of both antivirus and detection method by retraining the model on the new data points. They use an SVM classifier with a radial basis function (RBF) kernel.

In a recent publication, Hu, Cheng, Duan, Henderson, and Yin (2017) proposed an execution engine for JavaScript analysis irrespective of the environment or the sandbox variables. This is done by employing static code analysis for the exploration of all possible code paths and devising forced execution semantics for evaluation of JavaScript. As a result of this new system, there was an increase of 206.29% while using the same detection policy over HTML and PDF samples. The same technique can be extended for embedded JavaScript in Microsoft Office Documents. Although excellent, this method has some severe limitations; it completely stops the analysis if there is a syntax error. Part of forcing JS to execute is supplying it with faked strings; this may create a problem if the JS tries to use these in an *eval* function.

Table 1, gives a summary of malicious PDF detectors discussed so far. All JavaScript based detectors are limited by and are dependent on their respective extraction methods. It is the onus of the detector to perform not only proper parsing but also provide methods for identification of 'all' JavaScript components embedded in the document. This is difficult as JavaScript can be obfuscated, part of some extension which the parser cannot understand or just embedded in a section which is peculiar. We refer the reader to Carmony et al. (2016) for a detailed study of evasion techniques. Falah et al. (2018) point out the methods discussed are mostly client level and require a high level of user interaction.

3.4. Discussion and Conclusion

The Portable Document Format has been appropriated and used as a vector for malicious actions. The standardization of the format as an open standard in 2008 led to a larger attack surface. Numerous PDF readers and software are targetted continuously. Take for example, CVE-2018-14442, CVE-2017-10994 in Foxit Reader, CVE-2018-8350 in Microsoft Windows PDF Library, CVE-2019-5821, CVE-2019-5792 in Google's PDFium library.

Though sometimes incomplete, most tools try and provide an overview of irregularities presented in the document. An analyst or software interpret these for decision. As discussed with the case of PDFid for static analysis of documents, if the execution pattern or methodology is discoverable, then malicious actors can bypass and take advantage. The dynamic detectors are plagued with similar issues long with higher computational costs and sometimes ambiguous results. The academic research discussed was categoriesed into signature analysis and pattern matching, structural and metadata features and javascript based detectors. Signature-based detections are vulnerable metamorphism and polymorphic attacks. In most structural models as alluded in Maiorca, Corona, and Giacinto (2013), mimicry, reverse mimicry, and other polymorphic attacks play a very significant role and are quite prevalent. Maiorca and Biggio (2017) discuss the impact of these attacks in detail on PDF detectors. Javascript based detectors also face similar challenges.

In addition, detectors that use machine learning can be evaded by adversarial samples. These documents are created by adding distortion in benign documents and triggering a misclassification by preying upon the imperfect generalization of machine learning algorithms. These attacks are not limited to just traditional classification algorithms, but neural networks and deep learning are very susceptible. as discussed by Papernot et al. (2016). W. Xu, Qi, and Evans (2016) test a genetic algorithm for the creation of adversary samples against Šrndić and Laskov (2016) and Smutz and Stavrou (2016) approaches, and a proactive approach wherein the model is trained on adversarial samples. The proactive approach makes it easier for the models to ascertain the difference between an adversarial sample and a traditional document.

Table 1. Summary of the reviewed PDF detectors.

Detectors	Method	Static or Dynamic	Machine Learning
Shafiq et al. (2008)	Signature Analysis & Pattern Matching	Static	No
Cova et al. (2010)	JavaScript Features	Dynamic	Yes
Snow et al. (2011)	JavaScript Features	Dynamic	No
Schmitt et al. (2012)	Signature Analysis & Pattern Matching	Static	Yes
Maiorca et al. (2012)	Metadata & Structural Features	Static	Yes
Lu et al. (2013)	Signature Analysis & Pattern Matching	Static	Yes
Corona et al. (2014)	JavaScript Features	Static	Yes
Maiorca et al. (2015)	Metadata & Structural Features	Static	Yes
Šrndić and Laskov (2016)	Metadata & Structural Features	Static	Yes
Smutz and Stavrou (2016)	Metadata & Structural Features	Static	Yes
Nissim et al. (2016)	JavaScript Features	Static	Yes
Xu and Kim (2017)	Metadata & Structural Features	Dynamic	No
Li et al. (2017)	JavaScript Features	Static	Yes
Hu et al. (2017)	JavaScript Features	Dynamic	Yes
Cuan et al. (2018)	Metadata & Structural Features	Static	Yes

The choice of dependencies, analysis approach and decision algorithm at the time of design is crucial, as anyone of these can be vulnerable and lead to evasion or high false-negative rates.

4. MS Office Malware

Ever since coming to market, Microsoft Office applications such as Word, Excel, Powerpoint have been revered. These applications are commonplace with both enterprise users and well as personal computing users, thus making them a prime target for attacks just like the PDFs. Ever since the deployment of the software suite in 1990, there has been a rich history of being exploited by malicious authors.

In 1999, Melissa virus as described by Symantec (2007), a malicious macro infection attached to MS Word, was deployed causing mass hysteria and loss of thousands of work hours. The virus, which analysts still consider as one of the largest distributed malware adds itself to the user's macro definition library and itself to up to 50 people in their contact list. It is a widely accepted fact that despite security regulations placed by Microsoft, an attacker can bypass them. Dechaux, Filiol, and Fizaine (2010) showed, for an older version of Microsoft Office and Open Office, techniques through which an attacker can bypass default security measures preventing macro execution.

Unlike malicious documents that were created in the 1990s and early 2000s, attacks that leverage office documents today are often obfuscated or encrypted and thus harder to detect. Take for example the attack on Western Ukraine's power supply in 2016 as described in articles Zetter (2016) and Higgins (2016). The attackers exploited a known vulnerability from 2014 and malicious macro to bring down three power distribution centers and nearly 60 substations and leaving more than 230,000 residents without power. While introducing a more granular security control for macro in their enterprise vertical, Microsoft in Windows-Defender-Research (2016) noted, 98% of Office-targeted threats use macros.

Despite all the efforts both from the security community, antivirus developers, as well as software developers, attacks using malicious documents, are still rising. In Cisco (2018), the Cisco Annual Cybersecurity Report, authors state the most commonly detected malicious file extensions worldwide during the period of January through September 2017 were Office suite extensions which contributed 38% of the file share, the highest among the group. In the report McAfee (2018), total macro malware has seen a steady increase since Q2 of 2016. Aside from the more traditional attacks, malicious documents are being used to drop ransomware payload. Sophos (2017) mentions the use of MSWord files to spread ransomware of the Locky family. In Sherstobitoff (2018), the author indicates the use of malicious documents by the cybercrime group Lazarus for reconnaissance of users mining cryptocurrency.

In modern Office documents applications, the malware can be targeted to the victim through three methods:

(1) **Features**: Feature updates are common in every software cycle. When the software starts becoming close to obsolete new features are introduced to provide a new and better experience for the user. Sometimes these "features" may not be well thought out and leave room for exploitation. In 2017 the security community noticed a trend for using Dynamic Data Exchange (DDE) attacks using Office Documents. These attacks use Windows PowerShell, a feature provided to documents by Office Suite, for execution of the obfuscated scripts and were listed among the top 10 malware list by Laliberte (2018) in Q4, 2017.

(2) **Macro**: In general terms, macros are rule or pattern that specifies how a particular input sequence should be mapped to an output sequence on the basis of some procedure. Macros are widely used in the industry, but due to their common use in spreading viruses, they are blocked by default in Microsoft applications. Visual Basic Applications is the language used to program macros, and as such, they have most access to most Microsoft Windows calls.

(3) **Vulnerabilities**: The point has been made time and again that software vulnerabilities are the key reason why software fails under attacks. Microsoft updates their software

regularly but the due to the far reach of the software and lack of updates installed the vulnerabilities are quite easily targeted by the attackers.

Most often though there is a combination of one or more methods to form an attack. In addition to exploiting software vulnerabilities, attackers liberally use features like Macros, Object Linking and Embedding and Dynamic Data Exchange to form complex attacks.

4.1. Structure of Office Documents

Like in the case of PDF documents, it is necessary to understand how and where the malware can be embedded and what steps are needed to be taken to uncover said malicious files. Microsoft Office applications have followed two file formats, Compound File Binary Format (doc/ppt/xls) and Office Open XML (OOXML) format.

The Object Linking and Embedding (OLE) infrastructure is commonplace and is used to incorporate content from one application to the other. For example, incorporating Excel sheets into Powerpoint presentations. OLE objects can also be used to launch support modify and update objects. With this much power provided to OLE structures, they can be easily manipulated for malicious purposes. The most obvious type of malware in this category involves embedding an executable as an OLE object. Both old and modern office documents support and use this infrastructure.

4.1.1. Compound File Binary Format (CFB)

Compound File Binary Format as descibed in Microsoft-Corporation (2018), is the file format used by Office suites 97–2003 and sometimes referred to as OLE Compound Document Format. The format provided the software with a variety of independent data streams. The file format allows for the ability to store compiled programs (Portable Executables), media, and other files. This is the reason the file binary format container was widely used across Microsoft products at the time, like Office Documents (.doc/.ppt/.xls), Installers (.msi), Windows Thumbnails (thumb.db).

The structure of the file format was primarily inspired by the FAT filesystem, where blocks are defined and assigned to many allocation tables. Rentz (2007) describes a CFB document as, a hierarchical structure like a file system for data streams. There is a Root Storage that may link to other storage blocks or content streams. Each content stream gets divided into sectors, and these sectors are sometimes stored out of order as use sector chains to maintain coherence. The CFB header is of 512 bytes precisely and located in the first sector of the document. It contains all the data required for the document reader to open the file. Apart from these, there are sector allocation table and directory which help maintain the structure of the document system. Figure 3 displays 12 streams and their respective sizes in bytes. The streams were extracted from a random excel document, containing multiple sheets and macros, using the *oledump.py* tool.

4.1.2. Office Open XML Format (OOXML)

Unlike the CFB format, ZIP archiving standard was used to model Open Office XML format. This format became a standard after the release of Office 2007. The file can be visualized as an archive containing various files and folders. Majority of the streams are represented in the XML format, which follows global XML rules and is unique to OOXML. The content that is not text or text formatting such as image, multimedia, macro and other embedded content is represented as separate files and appears as such at the time of unarchiving.

A characteristic of this format is the particular emphasis it provides to the presence of macros. Whenever file containing some macro gets stored, it has the “m” suffix instead of the usual “x” suffix. Unlike CFB where embedded data is stored in streams and can be accessed directly, an Office Open XML document needs to expand, and the embedded structures expand with it. When it comes to the analysis of OOXML files, the analyst first has to “inflate” the file and then extract the various streams. Once extracted, they are checked for malicious content.

Figure 4 shows the directory structure of an empty MSWord document. The extraction contains, ‘[Content_Types].xml’, ‘_rels/’, ‘docProp/’ and ‘word/’ among other files depending on the contents as described in Microsoft (2007). The [content_type] file provides Multipurpose Internet Mail Extensions (MIME) type metadata for parts of the package. ‘_rels’

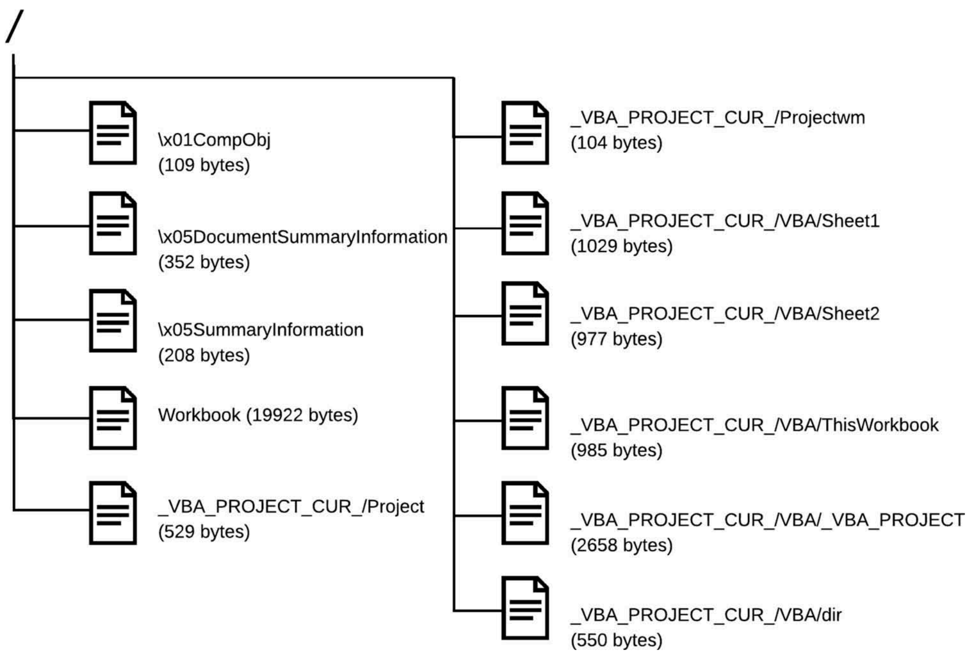

/
\x01CompObj (109 bytes)
\x05DocumentSummaryInformation (352 bytes)
\x05SummaryInformation (208 bytes)
Workbook (19922 bytes)
_VBA_PROJECT_CUR_/Project (529 bytes)
_VBA_PROJECT_CUR_/Projectwm (104 bytes)
_VBA_PROJECT_CUR_/VBA/Sheet1 (1029 bytes)
_VBA_PROJECT_CUR_/VBA/Sheet2 (977 bytes)
_VBA_PROJECT_CUR_/VBA/ThisWorkbook (985 bytes)
_VBA_PROJECT_CUR_/VBA/_VBA_PROJECT (2658 bytes)
_VBA_PROJECT_CUR_/VBA/dir (550 bytes)

Figure 3. Uncompressed structure of test.xls using extracted using *oledump* tool.

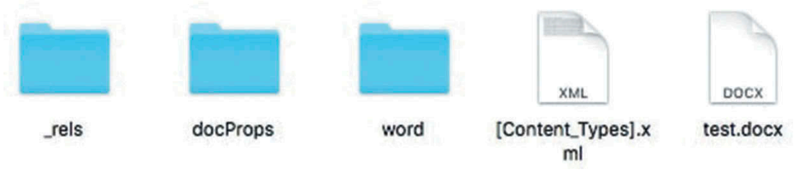


Figure 4. Uncompressed structure of test.docx.

contains all the relationships between various objects in the document in the package. 'docProps' folder contains properties related to the OOXML document and 'word' folder contains the core part of any Word document.

4.2. Detection and Analysis of Office Malware

Unlike PDFs where the attacker is leveraging the vulnerabilities primarily in the software to launch an attack, with office malware the launch of attack can be multifaceted and through many techniques as discussed earlier. It becomes impossible to create a system that can detect every single possible code execution path and test it for malicious activity. Visual Basic for Applications (VBA) is commonly used for the creation of macros, which are often used to create malicious documents.

The proliferation of office malware has become rampant and tools and utilities for the creation of these files are much more prevalent. As stated in Szappanos (2015), Microsoft Word Intruder (MWI) is one such infamous and utility which in the past few years has gained traction. It gives the attacker the ability to stack the use of few vulnerabilities on top of each as well as to create decoy documents. The proliferation of the malware is achieved via a dropper, i.e. the malicious document is embedded with a portable executable, or a downloader, i.e. the payload executes and downloads from the embedded URL and executes the file. Villeneuve and Homan (2015) reported the presence of command and control capabilities in the toolkit. Furthermore, an attacker could use utilities like Metasploit or one of the numbers of publicly available GitHub projects that available like Malicious Macro Generator and Generate-Macro.

Generally, the analyst follows the SANS guideline as discussed earlier and attempts an extraction of malicious objects. While creating malware, the attacker adds methods to avoid detection and analysis from standard tools. With static analysis, the analyst has the advantage of bypassing these defenses and extracting the relevant parts of the document.

4.2.1. *Static Analysis*

Just like static analysis of PDFs, there is a whole deck of tools available for static analysis of documents. Take, for example, Oletools it holds a package of tools to analyze CFB format documents, extract embedded objects, and analyzes the structure of OLE files. It was recently updated to recognize OOXML format and extract DDE links with msodde and extract VBA macros using olevba in OOXML files. OfficeMalScanner, is also a utility that does static analysis and provides support for shellcode and VBA extraction. It has subprocesses like MalHost-setup which allows for extraction of shellcode and embeds it in a portable executable (PE) for further analysis. OfficeDissector was one of the first parser explicitly designed for security analysis of OOXML documents. It can be used to extract all embedded contents present in an Office Document. There was a recent release of Vba2Graph utility that generates a VBA call graph based on static analysis. Figure 5 shows such a graph.

In the current security scenario just performing static analysis is not enough. Any attacker serious about their campaign obfuscates and encrypts their code which most often is not detected by static analysis.

4.2.2. *Dynamic Analysis*

Unlike PDFs, there are only have a handful tools to analyze Microsoft Office documents. This lack of dynamic analysis tools is linked to the dormancy of office-based attacks in the past. The analysis is generally done in two phases, extraction of the embedded file or feature and then execution in a sandboxed environment.

Many online sandboxes can be employed for the detection and analysis of malicious documents. MalDocs, anlyz.io, Cuckoo sandbox, VirusTotal, hybrid-analysis, Cryptam and others are some online dynamic analysis tools; the analyst submits their document to the platform of choice which analyzes and provides a report with details if malware was present or not, what family of malware.

There is a rich library of tools available for dynamic macro analysis; some of them are mentioned here. Lazy Office Analyzer takes office documents with obfuscation as inputs and tries to neutralize the obfuscation. It tries to extract complete VB scripts, JavaScripts and URLs from the document. Once extracted it is analyzed according to their respective class. ViperMonkey is a successful tool for the emulation of VBA in a sandboxed environment. It can be used to analyze and deobfuscate malicious Macros contained in Microsoft Office files. vhook a GitHub project, performs dynamic analysis of VBA code by looking for suspicious features like method calls to HTTP. Open, or API calls to URLDownloadToFileA, based on the report an analyst can suggest if a file seems malicious or not.

4.3. *Automated Detection of Malicious Office Documents*

The Microsoft Office document domain is fragmented, with two standard of file architectures and a number of file formats, it is difficult to create a detection algorithm that is global and broad spectrum. There are many PDF detectors that cannot be used for Office documents, for two reasons: the structure of document and the variability of attack. The following classification for malicious office document detectors is proposed (see Figure 6). (1) Entropy & Statistics Based Detection, (2) Signature-based Detection, (3) Metadata & Structural Features, (4) Shell code features and (5) Macro features.

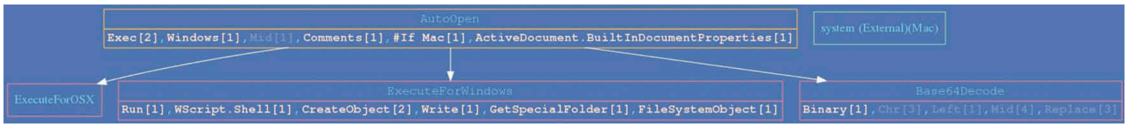

AutoOpen
Exec[2],Windows[1],Mid[1],Comments[1],#If Mac[1],ActiveDocument.BuiltInDocumentProperties[1]
system (External)(Mac)
ExecuteForOSX
ExecuteForWindows
Run[1],WScript.Shell[1],CreateObject[2],Write[1],GetSpecialFolder[1],FileSystemObject[1]
Base64Decode
Binary[1],Chr[3],Left[1],Mid[4],Replace[3]

Figure 5. Graph of VBA Macro made using vba2graph.

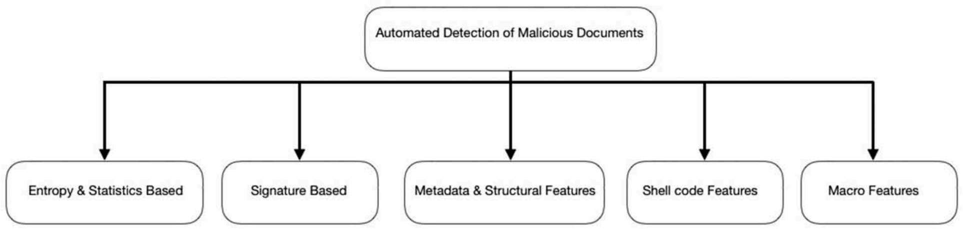

Automated Detection of Malicious Documents
Entropy & Statistics Based
Signature Based
Metadata & Structural Features
Shell code Features
Macro Features

Figure 6. Classification of Detectors for Office Documents.

4.3.1. Entropy and Statistics Based Features

In Li, Stolfo, Stavrou, Androulaki, and Keromytis (2007), authors suggest byte-level analysis on an unobfuscated document without any embedded content using n-grams and bloom filters. Methods based on byte-level analysis have long been studied and are known to fail against mimicry attacks, polymorphic samples and benign files embedded with malware. In Shafiq et al. (2008), a method for detection using Markov n-grams which shows an increase in detection rate but falls with similar shortcomings as Li et al. (2007).

In Gu et al. (2015) proposes a method of document classification based on wavelet package analysis. Their method is an amalgamation of entropy analysis and wavelet analysis where each document goes under a wavelet transformation and a feature extraction during the training phase. These features are stored in a library and are used in a feature match process with wavelet features of the unclassified document. This method has not been studied under various attacks scenarios, and therefore performance in a real-time detection is unknown.

4.3.2. Signature-Based Features

Schreck, Berger, and Göbel (2012) gave a signature-based method for document detection, where the detection part is divided into Automatic Exploit Detection and Vulnerability Identification. This method only takes into account the exploits that lead to extended instruction pointer (EIP) pointing outside the code segment and leaves all the other exploits undetected.

In the article Qbeitah and Aldwairi (2018), the authors discuss the use of dynamic analysis for detection of malicious files. The method proposed executes the file under analysis in either a honeypot or a Remnux sandbox. In either testing environment, they propose the use of utilities such as SNORT (2018), Dionaea (2018), Russinovich (2019), Wojner (2018), and Maddes et al. (2018) to observe the execution of the suspected file. They propose the generation and subsequent check of dynamic signatures based on observations made using these tools instead of using a signature-based on artifacts present in the file.

Tools like Alvarez (2018) and SNORT (2018) use Signature-based Features for detection of malicious documents. They store large databases of known malicious patterns and matches them with the file being analyzed. This detection method is relatively fast and needs less computational power. Malware easily evades such systems by employing obfuscation. Also, it fails to detect attacks which are not based on macros and scripts. Like OLE Remote Code Execution vulnerability CVE-2017-11882 is used by many attackers, but initially left undetected by all the Signature-based detection tools.

4.3.3. Metadata & Structural Features

Recently the focus has been shifted to using machine learning methods for the detection of malicious documents. Nissim, Cohen, and Elovici (2017), suggests the use of an active learning approach using Support Vector Machines with radial basis function kernel. They use the OOXML structure and A. Cohen, Nissim, Rokach, and Elovici (2016) to extract the most prominent structural paths that represent the documents under scrutiny. They compare their model with five different classifiers J48, Random-Forest, LogitBoost, Logistic Regression and Support Vector Machines and arrive at a 99.67% accuracy for the top 100 feature selections. They stipulate their method can detect mimicry and reverse mimicry attacks as in both cases there would be a change in structural features, but the accuracy of this statement has not been tested.

In Lin and Pao (2013), the proposed method bears a close resemblance to Maiorca et al. (2012), they propose a generalized method based on word frequency as features and use support vector machines for the classification of documents. They claim their method is robust against mimicry attacks, but no further studies have been conducted to prove the viability of this statement.

In 2018, in a Sophos sponsored research, Rudd, Harang, and Saxe (2018) took a corpus of 5 million documents from VirusTotal and 1 million documents from Common Crawl to classify using deep learning neural networks and decision tree ensemble using extreme Gradient Boosting (XGB). They chose four different byte-level features for implementation, which are platform and format agnostic. Their implementation does fall toward a higher false positive rate, which is not acceptable in a real-world scenario. They suggested but did not give empirical proof of protection of the robustness of features against adversarial attacks. It is hypothesized but never proven that despite numerous shortcomings of neural networks when challenged with adversarial examples, deep neural network method performs at decent efficiency in these scenarios.

4.3.4. Shellcode Features

Take Snow et al. (2011), this work laid down a technique for the detection of shellcode injection attacks using hardware virtualization. In the method proposed the analyst has the freedom of choosing any heuristic of their choice for the detection process. In the case of malicious documents, ShellOS lets the reader application render the document while ShellOS monitors the memory buffers created for shellcode. The method was tested on PDFs but can easily be translated for Office format documents. ShellOS used broad-spectrum heuristics and failed when challenged by heap spray attacks, social engineering and shellcode designed to execute under particular conditions.

Iwamoto and Wasaki (2016) in their work suggested a dynamic analysis work focused on the extraction of shell code from malicious office document based either on CFB or RTF. In the preprocessing stage, they perform analysis on the file format, disassemble byte sequences and create an order for emulation using entropy. The possible byte sequences are tested and if suspected of having shellcode converted to Microsoft Windows executable for further analysis. Like ShellOS it fails when encountered by ROP, malicious document droppers, and strongly environmentally dependent. It also fails in the presence of broken samples.

4.3.5. Macro Features

As seen previously from various examples, there has been a rise in the malicious macro files targeting. In Santos and Torres (2017), they discuss a framework for macro malware detection where the classification algorithm remains interchangeable. They have used python's OleFile and OleVBA for the extraction of features from 1,671 sample documents. They tested the samples using standard scikit-learn implementations of Neural Networks, SVM, Decision Trees, and random forests. They concluded that the best classifier works with an accuracy above 90% and can be used as a good filter for macro malware.

Bearden and Lo (2017), implement an n-gram feature selection approach with K nearest neighbor algorithm for detection and classification on *p*-code present in VBA macros. *p-code* represents the assembly language code generated for the execution of a macro; it is used to provide space-efficient storage and non-version specific execution of the macro. In their experiment, Bearden and Lo (2017) recognize the higher accuracy of 96.3% with 1-gram opcodes compared to 93.4% of 2-gram opcodes.

Obfuscation of malware is common and used to hide the intent of the script, most static analyzers fail to fully understand the nature of obfuscated malware and require either human intervention or automated classifiers using dynamic analysis. In Kim et al. (2018), they discuss the four types of obfuscation techniques that are prominently used. They used five different machine learning techniques, namely SVM, Random Forest, Multilayer perceptron, Linear Discriminant Analysis, and Bernoulli Naive Bayes, on 15 features to determine if a document is containing obfuscated macro programming. With these features set, the method achieved the highest accuracy of 97% with MLP and the highest precision of 98.2% with random forests. One should note here; malicious code detection is not the same as obfuscation detection. There is a strong relationship between the two, obfuscation in *most* cases hints a higher level of scrutiny is required but it does not prove the maliciousness of the file.

4.4. Prevention of Office Malware

Another school of thought while dealing with malicious documents is prevention instead of detection. By using prevention methods, a user can still view the content of the document without getting infected. Instead of detection and quarantine, the analysts seek to mitigate the damage by scrambling the payload stored in the document. Even with the case of malicious PDFs, as elaborated in Stevens (2011), software designers have adopted operating system technology such as Data Execution Prevention (DEP) and Address Space Layout Randomization (ASLR). These methods make it harder for an attacker to subvert protections either by making sections of memory non-executable as is the case of DEP or by randomizing the entry points of operating system and application-level functions at boot as is the case with ASLR.

The Microsoft office developers are also working actively in preventing malicious documents. Macro enabled documents is essential to have ‘m’ suffix in their extension like docm and xlsm. Since Office 2016, Microsoft (2019) has offered additional settings for managing macros called Group Policy Administrative Template files (ADMX/ADML). This template enforces policies like completely blocking macros, force verification by a virus scanner, allowing macros from trusted locations in the end-user system.

Li and Stolfo (2009) proposed Arbitrary Data Transformation (ADT) on the compound file binary format. ADT modifies the data stored in document binary arbitrarily in with the hope of damaging the malcode. If there is malicious code present in the document, the processing of damaged file would cause a software crash, while a benign document may present the document differently to the user but will remain stable. The methodology presented performs transformations in sandboxed environments. Due to the arbitrary nature of transformations, attacks that use brute-force guessing shall be thwarted.

These methods are targeted in protecting against memory-based attempts, one such method is document content randomization (DCR) as mentioned in Smutz and Stavrou (2015). The proposed methodology is based on ASLR. It incurs minimal computational cost during the transfer of document or before execution by the Office suite. Smutz and Stavrou (2015) studied the method on CFB documents and OOXML documents across six popular software vulnerabilities at the time with an average accuracy of 75.1%. The core thesis of the text revolves around performing DCR using document content fragment randomization (DCFR), and document content encoding randomization (DCER). The method fails if the exploit uses downloaders instead of droppers or using heap spray or other techniques for loading malware instead of document content. It depends on support from the operating system and the reader to stop these attacks. In the process of performing DCR, even though the content is presented the same, the structure of the document has been changed, this might create issues while matching digital signatures.

4.5. Discussion and Conclusion

Just as the case has been with PDF, Office documents have been used widely for a variety of attacks. They have also been linked to the proliferation of ransomware. It is the versatility of Office documents that makes it an appealing attack vector. As discussed, attackers can embed scripting languages like Javascript and Visual Basic as well as launch a command line or PowerShell terminal. In addition to this, there is a fragmentation of Office file formats with various extensions and different underlying architecture. Compared to even malicious PDF detectors, there has been less work published academically, especially with OOXML formatted files. It makes the format all the more enticing for malicious actors.

A common trend that can be noticed among all but one detection research for office documents is the lack of adequate training data; this fact can be gleaned easily from the text. Bearden and Lo (2017) attribute this problem to the corporate hierarchy. Malicious documents that are sent to corporations cannot be released until they have been analyzed and reported. Benign files, on the other hand, contain trade secrets and personal information and thus cannot be accessed. This lack of data could lead to statistically unstable results due to over-fitting. The results from these detectors do not provide us with a realistic estimation of detection rates, as also put forward by Rudd et al. (2018).

In the discussion of the academic contributions put forward for detecting malicious office documents, a classification was proposed. Table 2 provides a summary of Office document detectors

Table 2. Summary of the reviewed Office Document detectors.

Detector	Method	Static or Dynamic	Machine Learning
Li et al. (2007)	Entropy and Statistics Features	Static	No
Shafiq et al. (2008)	Entropy and Statistics Features	Static	No
Snow et al. (2011)	Shellcode Features	Dynamic	Yes
Schreck et al. (2012)	Signature-based Features	Dynamic	No
Lin and Pao (2013)	Metadata & Structural Features	Static	Yes
Gu et al. (2015)	Entropy and Statistics Features	Static	No
Iwamoto and Wasaki (2016)	Shellcode Features	Dynamic	No
Nissim et al. (2017)	Metadata & Structural Features	Static	Yes
Santos and Torres (2017)	Macro Features	Static	Yes
Bearden and Lo (2017)	Macro Features	Static	Yes
Rudd et al. (2018)	Metadata & Structural Features	Static	Yes
Qbeitah and Aldwairi (2018)	Signature-based Features	Dynamic	No

discussed. As was the case with PDF documents, Signature-based methods of detection have little success. They are severely dependent on their knowledge-base and are vulnerable to zero-day attacks. Just like metadata and structural features, entropy and statistics based features can also be overwhelmed by mimicry and other polymorphic attacks. Impact of these attacks and defenses against them are yet to be studied in the context of Office documents. Metadata and structural features may also be susceptible to adversarial samples. As mentioned in Section 3.4, detection strategies that use machine learning for generalization may fall vulnerable to adversarial learning. Kolosnjaji et al. (2018) propose a gradient-based approach to change byte-based features to test against a neural network-based malware detection. They achieved a maximum evasion rate of 60%. In order to minimize the effects of such exploits, the detector must be trained to perceive the difference between an adversarial sample and traditional sample.

Detectors discussed in Section 4.3.4 and 4.3.5 target a realtively small attack space. Systems that use shellcode features have been shown to fail against ROP techniques and dropper malware. Both shellcode and malicious macros detectors are often stumped by obfuscation of scripts. Evasion of detectors through obfuscation has become commonplace. Attackers are economically motivated to use all the obfuscation techniques present along with use novel techniques to bypass the detector. An attacker can take variety of approaches as seen in Kim et al. (2018) and Carmony et al. (2016).

Some of the PDF detectors can be modified such that they recognize Office documents instead. M. Xu and Kim (2017), Cuan et al. (2018), Hu et al. (2017) to name a few. In Bazrafshan et al. (2013), the authors discuss many heuristic-based malware detection techniques using heuristics, like API calls, CFGs and Opcodes, that have not been explored in the context of malicious documents. Take the case of Wu, Mao, Wei, Lee, and Wu (2012), and Ki, Kim, and Kim (2015), in both of these cases, the authors used API sequences to characterize and detect malware. This technique, among others, can be extended to malicious documents.

5. Conclusion

In this article, a comprehensive overview of existing methodologies for malicious document detection was presented. The underlying tools that are often used in detection methodologies were discussed. The classification of different methods was done on the basis of features selected and static/dynamic analysis. The research in PDF detectors was arranged into three classes, and the merits of each were discussed. A novel taxonomy for malicious office document detectors was also put forward. Given the design parameters, it can be difficult for practitioners to select the most appropriate methodology. It is believed that the broad cross-section of methods reviewed in this article would make this task comfortable. The discussion also hints at some gaps in the current state-of-the-art methods which can provide directions for future research.

Notes

1. Obfuscation refers to the process of deliberately hiding information, making it difficult to understand. Attackers use this method to hide malicious code,

thus evading detection. It may be implemented by using Base64 encoding, ROT13, XOR operations, or Runtime Packers on code segments.

2. Return Oriented Programming (ROP) provides functionality in terms of loops and conditional statements to the attacker. Instead of injecting malicious code in the memory, the attacker borrows already present code segments by changing the return address in the call stack.
3. In a buffer overflow or buffer overrun data is written beyond desired memory boundaries. It is often a starting point in an attack. Attacker deliberately puts in malformed inputs which lead the system to behave in unexpected ways. An exploit of this kind can lead to control-hijacking attacks, privilege escalation and malcode execution.
4. Heap spray is a method of writing some selected bytes at various places in the heap memory of the reader process, which can be accessed later as the vector of a separate attack.
5. Denial of Service (DoS) attacks target the availability of the resources. During the attack, the targeted resource becomes unavailable to legitimate users. It may be achieved by sending redundant and unnecessary requests for resources in order to overwhelm the system and prevent legitimate requests to be fulfilled.

ORCID

Priyansh Singh http://orcid.org/0000-0002-3901-7825
Sanchit Gupta http://orcid.org/0000-0002-6059-4352

References

Alvarez, V. (2018). *Yara*. Virustotal. Retrieved from https://virustotal.github.io/yara/

Bazrafshan, Z., Hashemi, H., Fard, S. M. H., & Hamzeh, A. (2013, 05). A survey on heuristic malware detection techniques. In *Ikt 2013-5th conference on information and knowledge technology* (p. 113–120), Shiraz, Iran. https://ieeexplore.ieee.org/abstract/document/6620049

Bearden, R., & Lo, D. C.-T. (2017). Automated microsoft office macro malware detection using machine learning. In *Big data (big data), 2017 ieee international conference on* (pp. 4448–4452), Boston, MA, USA. https://ieeexplore.ieee.org/abstract/document/8258483

Carmony, C., Zhang, M., Hu, X., Bhaskar, A. V., & Yin, H. (2016, 01). Extract me if you can: Abusing pdf parsers in malware detectors. In *Network and distributed system security symposium, San Diego, California*. https://www.ndss-symposium.org/ndss2016/

Cisco. (2018, 02). *Cisco 2018 annual cyber security report*. Retrieved from http://www.cisco.com/go/acr2018

Cohen, A., Nissim, N., Rokach, L., & Elovici, Y. (2016). Sfem: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods. *Expert Systems with Applications*, *63*, 324–343. doi:10.1016/j.eswa.2016.07.010

Cohen, Y., Hendler, D., & Rubin, A. (2018). Detection of malicious webmail attachments based on propagation patterns. *Knowledge-Based Systems*, *141*, 67–79. doi:10.1016/j.knosys.2017.11.011

Corona, I., Maiorca, D., Ariu, D., & Giacinto, G. (2014). Lux0r: Detection of malicious pdf- embedded javascript code through discriminant analysis of api references. In *Proceedings of the 2014 workshop on artificial intelligent and security workshop* (pp. 47–57), Scottsdale Arizona USA. https://dl.acm.org/doi/proceedings/10.1145/2666652

Cova, M., Kruegel, C., & Vigna, G. (2010). Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the 19th international conference on world wide web* (pp. 281–290), Raleigh North Carolina, USA. https://dl.acm.org/doi/proceedings/10.1145/1772690

Cuan, B., Aliénor, D., Delaplace, C., & Valois, M. (2018). *Malware detection in pdf files using machine learning* (Unpublished doctoral dissertation). REDOCS.

Dechaux, J., Filiol, E., & Fizaine, J.-P. (2010). *Office documents: New weapons of cyberwarfare*. Hack. Lu, Luxembourg, http://archive.hack.lu/2010/Filiol-Office-Documents-New-Weapons-of-Cyberwarfare-paper.pdf.

Dionaea. (2018). *Dionaea*. DinoTools. Retrieved from https://github.com/DinoTools/dionaea

Falah, A., Pan, L., Abdelrazek, M., & Doss, R. (2018). Identifying drawbacks in malicious pdf detectors. In *International conference on future network systems and security* (pp. 128–139), Paris, France. doi:10.1142/S2424835518720050

Fogla, P., & Lee, W. (2006). Evading network anomaly detection systems: Formal reasoning and practical techniques. In *Proceedings of the 13th acm conference on computer and communications security* (pp. 59–68), Alexandria, Virginia, USA.

Fogla, P., Sharif, M. I., Perdisci, R., Kolesnikov, O. M., & Lee, W. (2006). Polymorphic blending attacks. In *Usenix security symposium* (pp. 241–256), Vancouver, British Columbia, Canada.

Gu, B., Fang, Y., Jia, P., Liu, L., Zhang, L., & Wang, M. (2015). A new static detection method of malicious document based on wavelet package analysis. In *Intelligent information hiding and multimedia signal processing (iih-msp), 2015 international conference on* (pp. 333–336), Adelaide, SA, Australia.

Higgins, K. J. (2016, 01). *Macros, network sniffers, but still no 'smoking gun' in ukraine blackout*. Retrieved from https://www.darkreading.com/threat-intelligence/macros-network-sniffers-but-still-no-smoking-gun-in-ukraine-blackout/d/d-id/1324076

Hu, X., Cheng, Y., Duan, Y., Henderson, A., & Yin, H. (2017). Jsforce: A forced execution engine for malicious javascript detection. In *International conference on security and privacy in communication systems* (pp. 704–720), Niagara Falls, Ontario, Canada.

ISO, I. (2008). 32000-1: 2008, document management–portable document format–part 1: Pdf 1.7. *International Organization for Standardization, Geneva, Switzerland.*

Iwamoto, K., & Wasaki, K. (2016). A method for shellcode extraction from malicious document files using entropy and emulation. *International Journal of Engineering and Technology*, *8*(2), 101. doi:10.7763/IJET.2016.V8.866

Ki, Y., Kim, E., & Kim, H. K. (2015). A novel approach to detect malware based on api call sequence analysis. *International Journal of Distributed Sensor Networks*, *11*(6), 659101. doi:10.1155/2015/659101

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

Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Giacinto, G., Eckert, C., & Roli, F. (2018). Adversarial malware binaries: Evading deep learning for malware detection in executables. *arXiv preprint arXiv:1803.04173.*

Laliberte, M. (2018, 03). *Watchguard's q4 2017 internet security report released; malicious office document usage on the rise*. Retrieved from https://www.secplicity.org/2018/03/27/watchguards-q4-2017-internet-security-report-released-malicious-office-document-usage-on-the-rise/

Laskov, P., & Šrndić, N. (2011). Static detection of malicious javascript-bearing pdf documents. In *Proceedings of the 27th annual computer security applications conference* (pp. 373–382), Orlando Florida USA.

Leurs, L. (2017, 09). *Pdf versions*. Retrieved from https://www.prepressure.com/pdf/basics/version

Li, M., Liu, Y., Yu, M., Li, G., Wang, Y., & Liu, C. (2017). Fepdf: A robust feature extractor for malicious pdf detection. *Trustcom/bigdatase/icess*, *2017*(ieee), 218–224.

Li, W. J., & Stolfo, S. (2009). Thwarting attacks in malcode-bearing documents by altering data sector values.

Li, W. J., Stolfo, S., Stavrou, A., Androulaki, E., & Keromytis, A. D. (2007). A study of malcode-bearing documents. In *International conference on detection of intrusions and malware, and vulnerability assessment* (pp. 231–250), Lucerne, Switzerland.

Lin, J.-Y., & Pao, H.-K. (2013). Multi-view malicious document detection. In *Technologies and applications of artificial intelligence (taai), 2013 conference on* (pp. 170–175), Taipei, Taiwan.

Lu, X., Zhuge, J., Wang, R., Cao, Y., & Chen, Y. (2013). De-obfuscation and detection of malicious pdf files with high accuracy. In *System sciences (hicss), 2013 46th hawaii international conference on* (pp. 4890–4899), Wailea, Maui, HI, USA .

Maddes, Regshot, & Xhmikosr. (2018). *Regshot*. Retrieved from: https://sourceforge.net/projects/regshot

Maiorca, D., Ariu, D., Corona, I., & Giacinto, G. (2015). A structural and content-based approach for a precise and robust detection of malicious pdf files. In *2015 international conference on information systems security and privacy (icissp)* (pp. 27–36), Angers, France. doi:10.3389/fped.2015.00027

Maiorca, D., & Biggio, B. (2017). Digital investigation of pdf files: Unveiling traces of embedded malware. *IEEE Security & Privacy*, *17*(1), 63–71.

Maiorca, D., Corona, I., & Giacinto, G. (2013). Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious pdf files detection. In *Proceedings of the 8th acm sigsac symposium on information, computer and communications security* (pp. 119–130), Hangzhou China. doi:10.1016/j.jecp.2013.08.005

Maiorca, D., Giacinto, G., & Corona, I. (2012). A pattern recognition system for malicious pdf files detection. In *International workshop on machine learning and data mining in pattern recognition* (pp. 510–524), Berlin, Germany.

McAfee. (2018, 06). *Mcafee labs threats report*. Retrieved from https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-jun-2018.pdf

Microsoft. (2007, 10). *2007 office document: Open xml markup explained*. Retrieved from https://www.microsoft.com/en-za/download/details.aspx?id=15359

Microsoft. (2019). *Administrative template files (admx/adml) and office customization tool for office 365 proplus, office 2019, and office 2016*. Retrieved from https://www.microsoft.com/en-us/download/details.aspx?id=49030

Microsoft-Corporation. (2018, 08). *[ms-doc]: Word (.doc) binary file format*. Retrieved from https://interoperability.blob.core.windows.net/files/MS-DOC/[MS-DOC].pdf

Moser, A., Kruegel, C., & Kirda, E. (2007). Limits of static analysis for malware detection. In *Computer security applications conference, 2007. acsac 2007. twenty-third annual* (pp. 421–430), Miami Beach, Florida, USA . doi:10.1016/J.JHSE.2007.03.004

Nissim, N., Cohen, A., & Elovici, Y. (2017). 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), 631–646. doi:10.1109/TIFS.2016.2631905

Nissim, N., Cohen, A., Glezer, C., & Elovici, Y. (2015). Detection of malicious pdf files and directions for enhancements: A state-of-the art survey. *Computers & Security, 48*, 246–266. Nissim, N., Cohen, A., Moskovitch, R., Shabtai, A., Edri, M., BarAd, O., & Elovici, Y. (2016). Keeping pace with the creation of new malicious pdf files using an active-learning based detection framework. *Security Informatics*, *5*(1), 1.

Nissim, N., Cohen A., MoskovitchR.,Shabtai, A.,Edri, M., BarAd, O., & Elovici, Y.(2016). Keeping pace with the creation of new malicious PDF files using an active-learning based detection framework. *Security Informatics*,*5*(1),1

Oracle. (2015, 05). *Zero day malware threat prevention ensuring document safety with outside in clean content*. Retrieved from http://www.oracle.com/us/products/middleware/zero-day-malware-protection-brief-2607983.pdf

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016). The limitations of deep learning in adversarial settings. In *Security and privacy (euros&p), 2016 ieee european symposium on* (pp. 372–387), Saarbrucken, Germany. doi:10.4111/icu.2016.57.5.372

Prayudi, Y., Riadi, I., & Yusirwan, S. (2015). Implementation of malware analysis using static and dynamic analysis method. *International Journal of Computer Applications*, *117*(6), 11 –15 .

Qbeitah, M. A., & Aldwairi, M. (2018). Dynamic malware analysis of phishing emails. In *Information and communication systems* *(icics), 2018 9th international conference on* (pp. 18–24), Irbid, Jordan . doi:10.1142/S2424835518500029

Rentz, D. (2007, 08). *Microsoft compound document file format*. Retrieved from https://www.openoffice.org/sc/compdocfileformat.pdf

Rudd, E. M., Harang, R., & Saxe, J. (2018). Meade: Towards a malicious email attachment detection engine. arXiv preprint arXiv:1804.08162.

Russinovich, M. (2019, 03). *Process monitor v3.52*. Retrieved from https://docs.microsoft.com/en-us/sysinternals/downloads/procmon

Saeed, I. A., Selamat, A., & Abuagoub, A. M. (2013). A survey on malware and malware detection systems. *International Journal of Computer Applications*, *67*(16). doi:10.5120/11480-7108

Santos, S. D. L., & Torres, J. (2017). Macro malware detection using machine learning techniques - a new approach. In *Proceedings of the 3rd international conference on information systems security and privacy - volume 1: Icissp*, (p. 295–302). Proto, Portugal: SciTePress.

Schmitt, F., Gassen, J., & Gerhards-Padilla, E. (2012). Pdf scrutinizer: Detecting javascript- based attacks in pdf documents. In *2012 tenth annual international conference on privacy, security and trust* (pp. 104–111), Paris, France. doi:10.1016/j.ejphar.2011.10.045

Schreck, T., Berger, S., & Göbel, J. (2012). Bissam: Automatic vulnerability identification of office documents. In *International conference on detection of intrusions and malware, and vulnerability assessment* (pp. 204–213), Heraklion, Crete, Greece. doi:10.1177/1753193412446405

Shafiq, M. Z., Khayam, S. A., & Farooq, M. (2008). Embedded malware detection using markov n-grams. In *International conference on detection of intrusions and malware, and vulnerability assessment* (pp. 88–107), Paris, France.

Sherstobitoff, R. (2018, 02). *Lazarus resurfaces, targets global banks and bitcoin users*. Retrieved from https://www.mcafee.com/blogs/other-blogs/other-blogs/mcafee-labs/lazarus-resurfaces-targets-global-banks-bitcoin-users/

Smutz, C., & Stavrou, A. (2015). Preventing exploits in microsoft office documents through content randomization. In *International workshop on recent advances in intrusion detection* (pp. 225–246), Kyoto, Japan.

Smutz, C., & Stavrou, A. (2016). When a tree falls: Using diversity in ensemble classifiers to identify evasion in malware detectors. In *Ndss, San Diego, California*.

SNORT. (2018). *Snort*. Cisco-Systems. Retrieved from www.snort.org

Snow, K. Z., Krishnan, S., Monrose, F., & Provos, N. (2011). Shellos: Enabling fast detection and forensic analysis of code injection attacks. In *Usenix security symposium* (pp. 183–200), San Francisco, CA, United States.

Song, Y., Locasto, M. E., Stavrou, A., Keromytis, A. D., & Stolfo, S. J. (2010). On the infeasibility of modeling polymorphic shellcode. *Machine Learning*, *81*(2), 179–205. doi:10.1007/s10994-009-5143-5

Šrndić, N., & Laskov, P. (2016). Hidost: A static machine-learning-based detector of malicious files. *EURASIP Journal on Information Security*, *2016*(1), 22. doi:10.1186/s13635-016-0045-0

Sophos. (2017, 11). *Sophoslabs 2018 malware forecast*. Retrieved from https://www.sophos.com/en-us/en-us/medialibrary/PDFs/technical-papers/malware-forecast-2018.pdf

Stevens, D. (2011). Malicious pdf documents explained. *IEEE Security & Privacy*, *9*(1), 80–82. doi:10.1109/MSP.2011.14

Symantec. (2007, 02). *W97m.melissa.a*. Retrieved from https://www.symantec.com/security-center/writeup/2000-122113-1425-99

Szappanos, G. (2015, 08). *Microsoft word intruder revealed*. Retrieved from https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophos-microsoft-word-intruder-revealed.pdf

Veerappan, C. S., Keong, P. L. K., Tang, Z., & Tan, F. (2018). Taxonomy on malware evasion countermeasures techniques. In *Internet of things (wf-iot), 2018 ieee 4th world forum on* (pp. 558–563), Singapore, Singapore. doi:10.1177/1753193418768139

Villeneuve, N., & Homan, J. (2015, 01). *A new word document exploit kit*. Retrieved from https://www.fireeye.com/blog/threat-research/2015/04/a new word document.html

Virustotal. (2018, 09). *Submission statistics*. Retrieved from https://www.virustotal.com/statistics

Wash, R., & Cooper, M. M. (2018). Who provides phishing training?: Facts, stories, and people like me. In *Proceedings of the 2018 chi conference on human factors in computing systems* (p. 492), New York, NY, United States.

Windows-Defender-Research. (2016, 03). *New feature in office 2016 can block macros and help prevent infection*. Retrieved from https://cloudblogs.microsoft.com/microsoftsecure/2016/03/22/new-feature-inoffice-2016-can-block-macros-and-help-prevent-infection/

Wojner, C. (2018). *Welcome to procdot, a new way of visual malware analysis*. Retrieved from http://www.procdot.com/

Wu, D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., & Wu, K.-P. (2012). Droidmat: Android malware detection through manifest and api calls tracing. In *Information security (asia jcis), 2012 seventh asia joint conference on* (pp. 62–69), Tokyo, Japan.

Xu, M., & Kim, T. (2017). Platpal: Detecting malicious documents with platform diversity. In *26th usenix security symposium usenix security 17*) (pp. 271–287), Vancouver, BC, Canada.

Xu, W., Qi, Y., & Evans, D. (2016). Automatically evading classifiers. In *Proceedings of the 2016 network and distributed systems symposium, Dallas, Texas, USA*.

Zeltser, L. (2017, 09). *Malware analysis cheat sheet*. Retrieved from https://digital-forensics.sans.org/media/analyzing-malicious-document-files.pdf

Zetter, K. (2016, 03). *Inside the cunning, unprecedented hack of ukraine's power grid*. Retrieved from https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid