Automated Microsoft Office Macro Malware Detection Using Machine Learning

Ruth Bearden
Department of Computer Science
Kennesaw State University
Marietta, GA, USA
rbeard13@students.kennesaw.edu

Dan Chai-Tien Lo
Department of Computer Science
Kennesaw State University
Marietta, GA, USA
dlo2@kennesaw.edu

***Abstract*— Macro malware in Microsoft (MS) Office files has long persisted as a cybersecurity threat. Though it ebbed after its initial rampages around the turn of the century, it has reemerged as threat. Attackers are taking a persuasive approach and using document engineering, aided by improved data mining methods, to make MS Office file malware appear legitimate. Recent attacks have targeted specific corporations with malicious documents containing unusually relevant information. This development undermines the ability of users to distinguish between malicious and legitimate MS Office files and intensifies the need for automating macro malware detection. This study proposes a method of classifying MS Office files containing macros as *malicious* or *benign* using the K-Nearest Neighbors machine learning algorithm, feature selection, and TFIDF where *p-code* opcode *n*-grams (translated VBA macro code) compose the file features. This study achieves a 96.3% file classification accuracy on a sample set of 40 malicious and 118 benign MS Office files containing macros, and it demonstrates the effectiveness of this approach as a potential defense against macro malware. Finally, it discusses the challenges automated macro malware detection faces and possible solutions.**

Keywords—macro, malware, Microsoft Office, machine learning, p-code

I. Introduction

Macro malware has returned as a viable threat to corporate security. This strain of malware all but disappeared when Microsoft added macro disabling features to their Office products in response to early attacks such as the Melissa virus in 1999 [5,6,8]. For years, efforts have been made to educate users on how to identify malicious documents, but the fact that Microsoft Office macro malware has persisted for so long speaks of a need to find a better solution for detecting malicious macros.

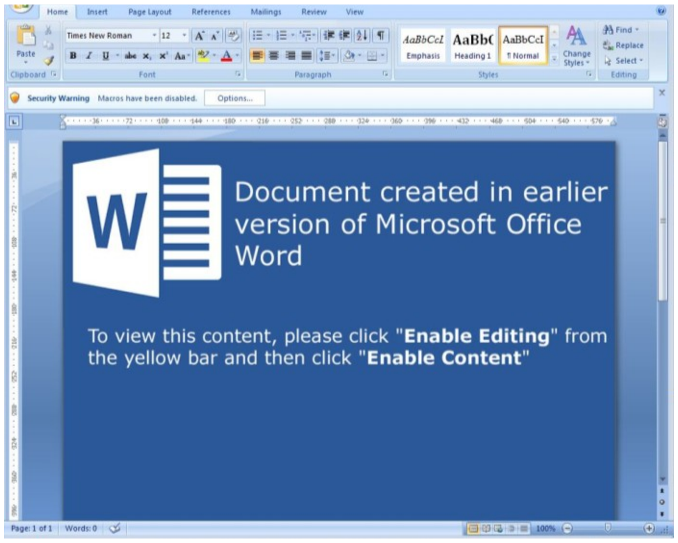
As far as we are aware, the realm of cybersecurity research lacks studies exploring the detection and prevention of macro malware in Microsoft Office files. To breach this gap, this paper demonstrates that using machine learning classifiers to automatically detect macro malware may prove a valuable tactic in the fight to improve cybersecurity.

II. Macro Malware Returns

Macro malware, programmed in Visual Basic for Applications (VBA), is a relatively easy type of malware to create. Microsoft Office products are used ubiquitously in corporations and MS Office files may easily be sent as email attachments to corporate employees. This makes it an appealing malware distribution vector. Attackers have invented creative ways to bypass the macro disabling safety feature. Macro malware no longer relies on brute force but on persuasion.

According to Grooten with Virus Bulletin [3], Word documents containing macro viruses typically try to persuade viewers to enable the macros. However, informing MS Office users of the dangers of enabling macros is no longer a sure defense. Attackers are applying social engineering techniques when designing malicious documents in an effort to make them appear legitimate. To increase the technique's effectiveness, attackers are using sophisticated information harvesting and data mining from creative places, such as a corporation document's metadata, to launch targeted attacks which are more successful than previous spam campaigns. These new techniques undermine the ability of malicious document receivers to make an informed decision on whether or not trust a document. [3]

This material is based in part upon work supported by the National Science Foundation under Grant Numbers 1623724, 1438858, 1244697, and 1241651. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.



Security Warning Macros have been disabled.

Document created in earlier version of Microsoft Office Word

To view this content, please click "**Enable Editing**" from the yellow bar and then click "**Enable Content**"

Fig. 1 MS Word document instructing user to enable macros

Fig. 1 is one sample taken from Payload Security [11], a hybrid malware analysis project combining static and dynamic analysis, which exemplifies the movement to make malicious documents persuasive. The document looks professional and even legitimate to an unwary MS Office user. It takes one corporate employee enabling the macro settings and running the malicious VBA code to compromise corporate security.

III. Background

Insights into current methods used to detect malicious MS Office files are difficult to obtain. As antivirus (AV) companies vie with one another for a place in the market, their methods quickly become carefully guarded trade secrets. There is evidence that the antivirus industry is beginning to use machine learning to provide an additional layer of protection, particularly against new attacks. For example, Windows Defender and Sophos, ranked top AV security by AV-Comparatives [16], both recently announced the incorporation of machine learning into their products [14][15]. Hence the demand for new ways incorporate machine learning into AV software is high.

Most antivirus software products detect previously identified malware instances by scanning them for signatures. AV software identifies malware based on the presence or absence of malicious signatures – a virus's "fingerprint" [2]. Signatures – often byte sequences, opcode sequences, or strings – are collected with both static and dynamic analysis [1] and stored in AV databases, a process often requiring a human being to observe a malware's behavior and identify the code that executed malicious behavior. To counter this, attackers have developed self-modifying malware such as polymorphic, metamorphic, and encrypted malware which obfuscate the signatures on which this detection method relies [10].

Furthermore, recent malware has been found to employ other elusive techniques – either scanning the environment for signs of being under observation and only executing on "normal" machines or waiting out antivirus and behavioral analysis software [3, 7]. If malware can bypass dynamic analysis, it can avoid having a signature generated and recorded. For this reason the need for automatic malware detection has increased.

It is important to note that signature-based malware detection is highly effective in detecting most previously identified instances. Automatic detection will never replace this method but rather add an extra layer of security and the ability to detect new malware. This study describes a potential way to add a layer of protection against macro malware attacks.

IV. Related Work

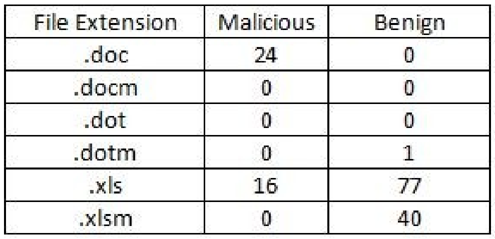
Malware detection with machine learning is an emerging solution to automatic malware detection in all types of malicious files, though more work has been done in its application to executable files (i.e. Portable Executable (PE) files). In the detection of malicious non-executables (i.e. PDF, Rich Text Documents, and Microsoft Office Documents) Cohen et al. conducted a study in 2016 in which compared various feature selection, feature representation, and top-feature selection methods and their combination with different machine learning classification algorithms. Though the study specifically targeted XML-based files, not those containing macros, it demonstrated the effectiveness of machine learning in file analysis and achieved a reported 97% accuracy [4].

Classifying documents based on *n*-gram frequency (where *n*-gram is a series of *n* words or other features) has become a popular data mining technique in text-based documents. However, Liu et al. applied this technique in classifying PE files. They de-compiled the PE files and treated the assembly opcodes as words (or grams), using series of *n*-grams to create sequences from which they derived features to train machine learning classifiers. This approach inherently captures both frequency and sequencing information from the file analyzed. On classifying new malware instances, they achieved an 86.7% accuracy [9]

In 2004, Abou-Assaleh et al. demonstrated high malware detection accuracy when analyzing a small sample of PE files using the K-Nearest Neighbors machine learning algorithm and opcode *n*-grams as features for the algorithm. This study reported 100% accuracy on the tests run [1]. This method is simple to implement and provides a basis on which to build a more complete classification method such as those created in the two previously mentioned studies. We modeled our approach after this last study.

V. Methodology

A. Data



File Extension	Malicious	Benign
.doc	24	0
.docm	0	0
.dot	0	0
.dotm	0	1
.xls	16	77
.xlsm	0	40

Fig. 2 File type frequency in sample set

Our data set was composed of 40 malicious and 118 benign files created in Office 2010. Fig. 2 shows the frequency of each file type in the sample set. (Note: All malicious files were marked as such by trusted spam filters and/or security analysts. Those retrieved from Payload Security [11] had a threat score of 85 or higher.)

The effectiveness of this study depends on the strength of correlation between our sample set and documents (malicious and non-malicious) in circulation in large corporations.

The majority of the *malicious* samples were retrieved from Payload Security [11], an automated malware analysis company. The files we selected were marked malicious by trusted spam filters and had a threat score of 85 or higher (on a scale of 0-100). Important to note is that each of the malicious files indicated auto-execute behavior according to their file analysis reports.

Benign files came from sample macro-containing files from Contextures [12]. These files implemented a variety of macro capabilities from ActiveX to database connection. It is acknowledged that benign macro-containing files created in the workplace will have a greater variety which this study could not represent. However, we do believe the benign part of the sample set contains sufficient variety to test our methods.

B. File Representation

Visual Basic for Applications (VBA) is the Visual Basic spin-off Microsoft has integrated with its Office products. Documents containing macros not only contain the VBA code itself but also *p-code*, an assembly language code generated by the VBA interpreter after the code has been run once. The purpose of p-code is to provide a non-version-specific and space efficient storage of the VBA script. We extracted the p-codes from each file using a python script available on GitHub [16] which we modified to obtain a list of p-code opcodes.

Our approach uses Term Frequency Inverse Document Frequency (TFIDF), a file representation method commonly used to represent word frequency and sequencing in text-based documents [18]. This method standardizes the frequency of a particular term according to the term's relative frequency in the document, TF (1), and its frequency in the entire sample set, or Document Frequency (DF). Each file is then defined as a vector, ***v***, in which $\boldsymbol{v_i}$ is a term weighted by its corresponding TFIDF, see eq. (2) where N is the number of documents in the sample set.

One strain of TFIDF file representation defines a term as a sequence of terms known as an *n*-gram, where *n* is the number of terms. This method has been applied, with success, to PE file malware detection. PE files consist of machine code which can be de-compiled to assembly language. The studies conducted in [1] and [9] represented PE files with TFIDF using assembly language opcodes sequences as *n*-grams.

$$TF = \frac{\text{term frequency}}{\text{max(term frequency in sample)}} \tag{1}$$

$$TFIDF = TF \times \log\left(\frac{\text{N}}{\text{DF}}\right) \tag{2}$$

C. Feature Selection

To select meaningful features, we use the DF feature selection method which ranks features according to the number of documents in which that appear. However, we also sought to incorporate the concepts of the Fisher Score selection method which scores features so that the variance in feature weights between benign and malicious files is maximized [13].

For feature extraction we rank features according to an equation that reflects the feature DF and its tendency to be present in either benign or malicious documents. We determine a feature's rank (R) as DF multiplied by C, a value which reflects the tendency of a feature to appear in either benign or malicious samples as shown in (3-6).

Equations (3) and (4) represent the DF of each feature within the malicious and benign file subsets. In (5), C ranges from 0, when $\text{DF}_{\text{benign}} = \text{DF}_{\text{malicious}}$, to ln(N + 1) when a feature frequents only one classification and not the other.

$$m = \frac{\text{DF}_{\text{malicious}}}{\text{number of malicious samples}} \tag{3}$$

$$b = \frac{\text{DF}_{\text{benign}}}{\text{number of benign samples}} \tag{4}$$

$$C = \begin{cases} \ln\left(\dfrac{\max(\text{m, b})}{\min(\text{m, b})}\right) & \text{where } m \neq 0 \text{ and } b \neq 0 \\ \ln(\text{N+1}) & \text{where } m = 0 \text{ or } b = 0 \end{cases} \tag{5}$$

$$R = DF \times C \tag{6}$$

D. Classification with Machine Learning

The number and variety of machine learning algorithms useful in classifying MS Word documents is extensive. We chose the K-Nearest Neighbors algorithm, one of the simplest, for while much can be gained from a testing the accuracy of different classifiers, the extent of this study is to provide a proof of concept for representing and classifying MS Word macros based on p-code. We hope to compare different algorithms in future works.

We separated the samples randomly into two subsets of vectors – a training set (90% of the samples) and testing set (10% of the samples) – where the training set classification was known. KNN, as a lazy learner, used only the randomized training set to make a prediction on each sample from the testing set. Given the small overall sample size, we combined each *n* (used in *n*-gram), K (used to find K nearest neighbors), and L (used to get L top features) and classified the testing set 50 times, each with newly randomized training and testing subsets. We used the average percent prediction accuracy to evaluate the overall accuracy of each combination.

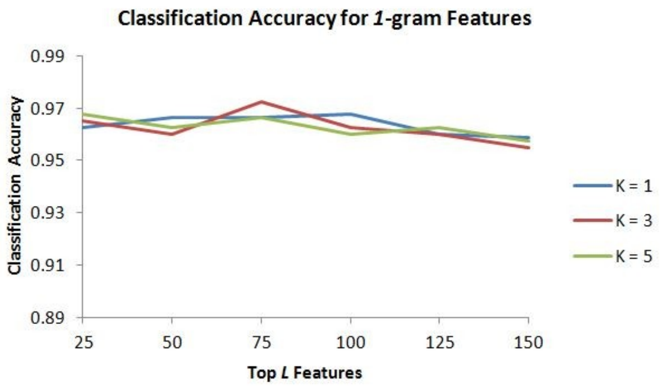


Fig. 3 Results for *n* = 1 (*1*-gram) Features

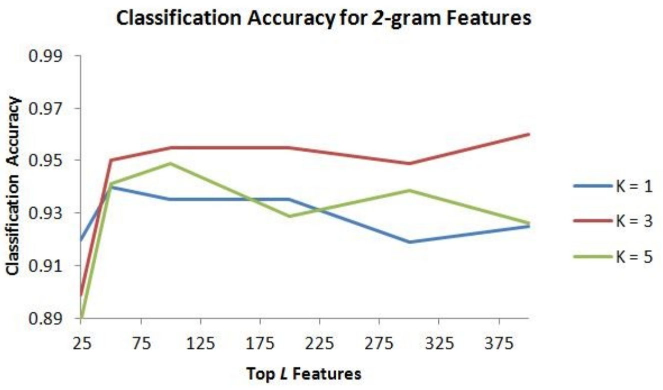


Fig. 4 Results for *n* = 2 (*2*-gram) Features

We classified samples for each n = 1, 2 and K = 1, 3, 5. For *n* = 1, *L* ranged from 25 to 150 and for *n* = 2 from 25 to 400. Fig. 3 and Fig. 4 show the results for each K and *L*. On average, the overall accuracy of predictions where n = 1 was 96.3% and where n = 2 was 93.4%. We gather from this that in general using *1*-gram opcodes as features has a higher accuracy that using *2*-gram opcodes. However, *2*-gram features for K = 3 where *L* >= 50 shows has average accuracy 95.4%.

VI. Future Work

As far as we are aware, there is a lack of available research exploring automatic macro malware detection in MS Office documents, yet this malware strain has lingered as a threat for over a decade. We hope this study will spark interest in further improvements in automated macro malware detection and provide a platform for research exploring ways to improve the detection accuracy when classified by machine learning algorithms. Our research implemented a simple machine learning algorithm and achieved 96.3% detection accuracy. This is not as accurate as the results Abou-Assaleh et. al achieved in classifying PE files with similar methods. It does come close to the Cohen et al XML-based document malware detection accuracy of 97%, and it improves upon the 86.7% Liu et al reported for PE file classification accuracy using an enhanced *n*-gram method.

Before closing, we lay out some challenges and potential solutions to future work in this field.

A. Challenges

Several challenges face automated macro malware detection. Most prominent is the difficulty of collecting adequate training data – both in number and in quality. Compared to PE files which are widely and freely distributed, MS Office* files are harder to find. *Malicious* MS Office files are sent to corporations, and researchers cannot access them until they have been identified, analyzed, and reported. *Benign* MS Office files are carefully guarded as they contain corporate information which could threaten corporate security and/or trade secrets if released. This makes it difficult to train machine learning classifiers on a large file sample set. Small sample sets may be used but in machine learning may cause data overfitting problems. Furthermore, small sample sets may not contain an even distribution of file types. The sample set this study uses is heavily weighted with MS Word files for *malicious* files and with MS Excel files for *benign* files.

B. Solutions

The beauty of the method of automated macro malware detection this study proposed is its reliance on information retrieved from files without opening them. We see the next step in macro malware detection development to involve outsourcing initial sample file processing to corporations via email attachment scanning or other means. This would enable collection of an accurate and large sample sets on which to train machine learning classifiers.

* MS Office files containing macros

VII. References

[1] T. Abou-Assaleh, N. Cercone, V. Keselj, R. Sweidan. "N-gram-based Detection of New Malicious Code." *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, *No. 2*, pp. 41-42

[2] S. S. Anju, P. Harmya, N. Jagadeesh, & R. Darsana. 2010. "Malware Detection Using Assembly Code and Control Flow Graph Optimization." Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India, No. 65. DOI: 10.1145/1858378.1858443

[3] G. Bunker. 2016. "Protecting Against Tomorrows Attacks Today." ISSA Journal, 14, 4. 19-23. https://www.clearswift.com/sites/default/files/Protecting%20against%20Tomorrow%E2%80%99s%20Malware_ISSA0416.pdf

[4] A. Cohen, N. Nissim, L. Rokach, & Y. Elovici. 2016. "SFEM: Structural Feature Extraction Methodology for the Detection of Malicious Office Documents Using Machine Learning Methods." Expert Systems with Applications, vol. 63, pp. 324-343. DOI: https://doi.org/10.1016/j.eswa.2016.07.010.

[5] Fandom, "Virus Information: Melissa," http://malware.wikia.com/wiki/Melissa Accessed: Nov. 19, 2017

[6] M. Grooten, "VBA is not dead," 2014, https://www.virusbulletin.com/virusbulletin/2014/07/vba-not-dead

[7] C. Kolbitsch, E. Kirda, & C. Kruegel. 2011. "The Power of Procrastination: Detection and Mitigation of Execution-Stalling Malicious Code." Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 285 – 296. DOI: 10.1145/2046707.2046740

[8] T. Smith, "Mac users hit by Melissa macro virus," 2001 Available: https://www.theregister.co.uk/2001/01/19/mac_users_hit_by_melissa/ Accessed: Apr. 26, 2017

[9] L. Liu, B. Wang, B. Yu, & Q. Zhong. 2016. "Automatic malware classification and new malware detection using machine learning." Frontiers of Information Technology & Electronic Engineering. www.zju.edu.cn/jzus/openiptxt.php?doi=10.1631/FITEE.1601325

[10] M. Lindorfer, A. Di Federico, M. Federico, P. M. Comparetti, & S. Zanero. 2012. "Lines of Malicious Code: Insights into the Malicious Software Industry." *Annual Computer Security Applications Conference*, pp. 349 – 358. DOI: 10.1145/2420950.2421001

[11] Payload-Security, 2017.Available: https://www.payload-security.com Accessed: Nov. 10, 2017

[12] Contextures, 2017.
Available: http://www.contextures.com/excelfiles.html
Accessed: Nov. 13, 2017

[13] A. Shabtai, R. Moskovitch, Y. Elovici, & C. Glezer. "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey." February 2009. *Information Security Technical Report*, vol. 14.1, pp16-29.

DOI: https://doi.org/10.1016/j.istr.2009.03.003

Microsoft TechNet. "Antivirus Evolved," 2017. [Online]. Available: https://blogs.technet.microsoft.com/mmpc/2017/05/08/antivirus-evolved/ [Accessed: Oct. 10, 2017]

[14] Sophos. Press Release. "Sophos Adds Advanced Machine Learning to Its Next-Generation Endpoint Protection Portfolio with Acquisition of Invincea." Februray 8, 2017. https://www.sophos.com/en-us/press-office/press-releases/2017/02/sophos-adds-advanced-machine-learning-to-its-next-generation-endpoint-protection-portfolio.aspx

[15] AV-Comparatives. "Independent Tests of Anti-Virus Software: File Detection Test September 2016," 2016. [Online] Available: https://www.av-comparatives.org/file-detection-test-september-2016/ [Accessed: Nov. 13, 2017]

[16] V. Bontchev. "Pcodedmp.py," 2016. [Online] Available: https://github.com/bontchev/pcodedmp. [Accessed: Nov. 13, 2017]