Improving Detection of Malicious Office Documents using One-Side Classifiers

Silviu - Constantin Viţel
"Al.I. Cuza" University - Faculty of Computer Science
Bitdefender Laboratory
Iaşi, România
Email: svitel@bitdefender.com

Gheorghe Balan
"Al.I. Cuza" University - Faculty of Computer Science
Bitdefender Laboratory
Iaşi, România
Email: gbalan@bitdefender.com

Dumitru Bogdan Prelipcean
"Al.I. Cuza" University - Faculty of Computer Science
Bitdefender Laboratory
Iaşi, România
Email: bprelipcean@bitdefender.com

***Abstract*—The current threat landscape is diverse and has lately been shifting from the binary executable application to a more light-coded and data-oriented approach. Considering this, the use of Microsoft Office documents in attacks has increased. The number of malicious samples is high and the complexity of evasion techniques is also challenging. The VBA macros are highly used in enterprise environments with benign purposes, so, in terms of detection, the number of false alarms should be close to zero.**

In this paper we discuss and propose a solution which focuses on keeping the rate of false positives as low as possible and, at the same time, to maximize the detection rate.

***Keywords*-Malware detection, classifier, feature extraction, feature selection, VBA macro, malicious macros, Office documents**

I. Introduction

In recent years, the attacks based on malicious scripts (VBA) from Microsoft Office documents have been growing in popularity and so has the necessity of detecting them. While the VBA language was originally developed as a powerful scripting language to help users automate tasks and create macro-driven applications, it became a prevalent infection vector due to its extended capabilities, such as accessing the native Windows system calls. As a response to the growing threat of malicious macros, Microsoft has implemented several security features to prevent the execution of unwanted code. Nowadays, as a countermeasure, the attackers distribute the malicious documents through spear-phishing emails and use social engineering techniques in order to trick the victim into enabling the execution of the malicious macros.

This paper aims to explain the current state of detections regarding this type of malicious files and analyzes the possibility of improving existing macro detections via machine learning techniques. We will consider a lightweight approach of this idea due to the limitations of it being used in an anti-malware solution where performance is a highly important requirement. Our solution is based on a derived version of the Perceptron algorithm which builds a detection model focused on the properties of the macro code extracted from the VBA project of Microsoft Office files. The paper also explores the advantages of using features that are not based around a particularity of the malicious files.

II. Related work

The problem of detecting malicious macros inside Office documents has determined the creation of multiple analysis tools meant to help researchers and, in recent years, multiple studies were conducted to test new methods of detecting malicious files. The main focus of these methods are the structural properties of the macro code which derive from obfuscation or features that try to describe general aspects of the code and behavioral features.

In addition to having analysis capabilities, the mentioned tools are also aimed at detecting malicious traits inside Office files. OfficeMalScanner is a free forensic tool that can be used to scan for malicious traces such as shellcode heuristics, PE-files or embedded OLE streams in legacy files which use the old OLE binary format. Another tool aimed at detecting vulnerabilities is OfficeCAT which is an Office file checker based on searching specific signatures to determine if a legacy file is unsafe. Microsoft OffVis is a tool that helps with the visualisation and understanding of Office files in OLE format (.doc, .xls, .ppt) and can also identify exploits by checking vulnerability signatures. pyOLEScanner is a python script inspired by OfficeMalScanner which has the

ability to scan and evaluate a file in order to determine if it could be malicious.

A machine learning approach to malicious macro detection was presented by Sangwoo Kim et al. It is based on 15 discriminant static features that describe certain obfuscation techniques used in the code. These features are associated with four types of obfuscation methods: random obfuscation, split obfuscation, encoding obfuscation and logic obfuscation. The training was performed on a set of 2,537 files (773 benign, 1,764 malicious) and four classifiers were tested: SVM (95,5%), Random Forest (96,5%), Multi-Layer Perceptron (97%), Linear Discriminant Analysis (90,1%) and Bernoulli Naive Bayes (89,1%).

A similar approach to macro detection is described by Ed Aboud and Darragh O'Brien. They are using features based on the macro found inside Office files which can describe more generic properties of the VBA code, such as Macro Keywords, Count of Integer Variables, etc. Additionally, an OCR library is used to identify certain phrases used in social engineering attacks such as Enable Content and Previous Version. The images containing these phrases are extracted by searching image magic numbers in the OLE binary. Using the extracted characteristics, they created a feature vector which was the input for five different classifiers. The training was performed on a set of 400 samples (200 benign, 200 malicious) and the testing phase consisted of 528 malicious samples and 83 benign samples. The performance of these classifiers was evaluated in terms of the TPR (True Positives Rate): Random Forest (98,9875%), KNeighbors (97,527%), DecisionTree (98,225%), GaussianNB (97,02%). In Macro Malware Detection using Machine Learning Techniques, Sergio De los Santos and Jose Torres propose the idea of using the features extracted by the python libraries OleFile and OleVBA. They establish a feature vector composed of 45 properties, but they highlight the fact that they are not distinct, some being obtained by discretizing features that are not boolean in nature. Their system was trained using the following classifiers with the following accuracy values during the test phase: SVM (89%), Decision Tree (95%), Random Forest (94%) and Neural Networks (99%).

ALDOCX is a framework implemented by Nisam et al. based on machine learning classifiers, which uses features related to the structure of the paths in a ZIP archive, a methodology named SFEM. The extracted properties serve as input for an SVM classifier paired with an Active Learning component, which allows the system to continuously update the detection model. The classifier was trained on a set of 16,811 samples (327 malicious, 16,484 benign) and achieved a TPR of 93.34%, FPR (False Positives Rate) of 0.19% and 99.67% accuracy. It is important to point out that, due to the process of choosing the features (they are paths inside a zip file), this framework only supports the newer OOXML file format (.docx, .xlsx) of Office files.

Although not a detection method itself, Microsoft implemented back in 2015 the Antimalware Scan Interface (AMSI) which allows applications on Windows 10 to request a scan of the memory buffer by an AV solution. This allows the logging of macro behaviour at runtime, which can be used to analyze macro code in its deobfuscated state.

As we can observe from the research work and tools mentioned above, most of the detection techniques are based around obfuscation, dynamic analysis, static, signature-based evaluation or a rather low number of features extracted from the macro code inside the VBA project found in Office files. In the following sections, we will present our solution regarding the detection of malicious macro code using features which can describe the sample space in a more generic manner.

III. Problem description

The key points of the VBA-based malicious software are the accessibility and the popularity of Office Products. This is due to the fact that those products offer a scripting component named *macro*. A *macro* is a scripting system initially aimed at automating certain tasks. Being written in Visual Basic, it has access to the majority of the functions implemented in Windows API. This, along with the mass usage of Office products, both in business and non-business fields, makes it one of the favourite tools for malware creators. According to a statistic made by Spiceworks[1], nearly 82% of companies use programs from the Microsoft Office package. Knowing this, it is easier for an attacker to build a malicious program which will be granted to run, with a certain probability, on the target machine. However, over the past years, Microsoft implemented different security patches which fixed known vulnerabilities and nowadays the execution of a macro without user content is denied. Because of these measures, the attackers had to implement social engineering methods to trick users into executing malicious macros.

Malicious macros are used both as a delivery method and as a standalone malicious entity. If, in the past, the main target of a malicious document was to spread itself across the network, nowadays the potential offered by those products made the malware creators to rethink their usage. It is more common to see a document whose target is to steal sensitive information from companies or to deliver a destructive component, for example, a ransomware. According to CISCO[2], more than 38% of malicious file extensions found in 2017 in groups of malicious prevalent files were Microsoft Office formats.

Considering the above observation, we developed a solution to detect malicious macros inside an office file and keep the rate of false positives as low as possible, all with

[1]https://community.spiceworks.com/software/articles/2873-data-snapshot-the-state-of-productivity-suites-in-the-workplace

[2]https://www.cisco.com/c/dam/m/digital/elq-cmcglobal/witb/acr2018/acr2018final.pdf

respect to performance requirements. Our system is based on a derived version of the perceptron, OSC[**?**], and it is used over a set of features extracted from the binary Visual Basic Application, which is found inside an office document. The system extracts the macros within a file, "tokenizes" them, extracts the sequence of features and feeds them to the machine learning algorithm. In this way, we provide a complete system to detect if an Office file is indeed malicious or not.

Moreover, taking into account that the OSC is keeping the rate of false positives for its models at a small percentage, we can easily use and integrate them in Bitdefender's technologies. Having a small amount of false positives is mandatory due to the fact that Office products are used in critical business infrastructures.

IV. Database and features

The classifier described in this paper was tested on a set of samples provided by the Bitdefender Cyber Threat Intelligence Laboratory. The initial dataset consisted of 217,557 Office files and VBA projects (.doc, .docx, .xls, .xlsx) collected from June 2017 to May 2019, 176,314 malicious and 41,243 benign. In the further presentation of the features and samples selection we will use the word token to describe an entity delimited by whitespaces. The samples were passed through a filter which removed documents whose macros had 32 tokens or less, because we considered that those files do not provide relevant information for our training process.

As a result of the filtering process, the number of malicious samples was unchanged and the number of benign samples was reduced to 38,668. For the training process, we selected 30,000 of the 38,668 benign samples and 10,000 malicious samples in a random fashion.

The features used in the training process have been extracted by various processing methods applied to the tokens found in the macro code of the Office files. This static approach of the features extraction is motivated by the need of accommodating certain performance parameters. In order to create a model that is as generic as possible, we extracted properties that are specific to clean samples, to malicious ones and properties which can describe both classes of samples. We arrived at a set of 536 features which describe general characteristics of the macro code (for example, the number of variables containing digits in their name), obfuscation features (for example, the number of statements in an assignment) but they also describe behavioral characteristics, such as a function call with suspicious parameters.

Due to the aforementioned performance restrictions, we will use only boolean features. This method also reduces the space needed to store the values of the features. Because some these values are not inherently boolean, the features need to be discretized. For example, we observed that in the case of some malicious samples, a higher number of small functions is present in malicious samples than in benign ones. This information led us to the selection of 12 intervals which represent the values of the feature in a way that provides more relevant data. (Table I)

Name	Interval	Feature Name
NR SMALL FUNCTIONS	$[1, 3)$	*"under-3-functions"*
NR SMALL FUNCTIONS	$[3, 4)$	*"under-4-functions"*
...	...	...
NR SMALL FUNCTIONS	$[953, 954)$	*"under-954-methods"*

Table I
EXAMPLE OF DISCRETIZATION

By discretizing the values, from an initial set of 536 features, we arrived at a feature vector of 2977 properties. By design, our approach will use only 256 features. These features are chosen using a process named Conditional Mutual Information Maximization, because we want our model to be constructed with features which bring maximum information gain.

V. Results

As we previously stated, we built our detection model around the Perceptron algorithm and our objective is to construct the model such that it represents a generic description of the chosen dataset.

Before we present the results, it is important to state that one of the main focuses of the training and testing phases is to achieve a very low false positives rate. This is especially important if we plan to further develop our model for commercial use. Consequently, we chose to use a modified version of the Perceptron algorithm name OSC, which incorporates an extra stage in the training algorithm to insure that all the samples marked as benign are correctly classified. Furthermore, the evaluation of a sample in the OSC algorithm is identical to the one used in the Perceptron algorithm, which is an advantage from the standpoint of performance.

In order to train our model we used two approaches: OSC and OSC_U. The first approach involves using almost all our records. It is important to express the fact that our training data does not involve any inconsistencies. We trained our model for 500 iterations and we developed an accuracy of 99.68% and sensitivity of 98.73%.

The second method used, named OSC_U, involved removing duplicate records from the entries used as inputs for the training algorithm. In this regard, if we encountered multiple records with the same feature vector (and labeled the same) we only kept one record.

As a result of this filtering, the number of training samples was reduced to 14,495 (11,636 benign, 2859 malicious). This new model yielded an accuracy of 99.65% and sensitivity of 98.25%.

We have also constructed a test set of samples in order to compare our two proposed models. We selected 8668

Name	Sensitivity	Accuracy
OSC — 500	98.73%	99.86%
OSC_U — 500	98.25%	99.65%

Table II
COMPARATIVE TRAINING RESULTS

benign samples and 10,000 malicious samples. On this set we observed a detection rate of PLACEHOLDER with a false positives rate of PLACEHOLDER.

Name	FP	FP Rate	TP	Sensitivity
OSC — 500	23	rate%	truepos	sens%
OSC_U — 500	16	rate%	truepos	sens

Table III
RESULTS ON TEST DATASET

VI. CONCLUSION

If we evaluate the current threat landscape, we observe that macro malware has become a prevalent security problem. Traditional signature-based detections have the advantage of being a safe detection method with low false positive rates, being easy to generate and maintain. Although effective on short-term, these methods do not produce a solution which has the capacity to generalize the malicious features of new threats and can be easy to bypass. Analyzing the results presented in this paper, we can state that our solution, along with existing protection measures is a good step toward a highly generic and accurate detection system.

We acknowledge the fact that our perceptron model is more difficult to maintain and train and requires a large collection of samples and compute power. Despite these requirements, this type of model is not as volatile and is capable of better generalizing the characteristics of different samples.

In a real world scenario, we consider that a middle way should be chosen. The traditional methods provide stability to the detection process of known malware families, while the perceptron model can be more exploratory in nature and discover new unknown threats.

Consequently, the solution presented in this paper is essential to a cutting edge security solution and an actual necessity in the current state of growing macro malware threats.

REFERENCES

[1] "Combating a spate of java malware with machine learning in real-time." [Online]. Available: https://cloudblogs.microsoft.com/microsoftsecure/2017/04/20/combating-a-wave-of-java-malware-with-machine-learning-in-real-time/