Improving Detection of Malicious Office Documents using One-Side Classifiers

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

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

Dumitru Bogdan Prelipcean
"Al.I. Cuza" University - Faculty of Computer Science
Bitdefender Cyber Threat Intelligence Lab
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, maximizes 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[1] 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.

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[2], OfficeCAT[3], Microsoft OffVis[4] and pyOLEScanner[5] are tools which can be used to scan for malicious traces and vulnerabilities inside Office files in OLE format (.doc, .xls, .ppt).

A machine learning approach to malicious macro detection was presented by Sangwoo Kim et al.[1] It

[1]Spear-phishing is a phishing attempt targeted to certain individuals or companies. Phising is a fraudulent practice that aims to obtain sensitive information from users (such as passwords, credit card information, etc.) or trick them in performing certain actions by concealing this practice under the impression of communicating with a trustworthy entity.

[2]http://www.reconstructer.org/about.html
[3]https://www.aldeid.com/wiki/Officecat
[4]https://www.aldeid.com/wiki/OffVis
[5]https://www.aldeid.com/wiki/PyOLEScanner

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 five 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[2]. 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. 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"[3], Sergio De los Santos and Jose Torres propose the idea of using the features extracted by the python libraries OleFile and OleVBA. 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.[4] 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.

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. Being written in Visual Basic, macros 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[6], 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. A common method of attack is based on the previously mentioned phishing attempts. The victim receives an email with a malicious attachment consisting of an Office document, claiming to be a legitimate file such as an invoice for a purchase or a legal document. Upon opening it, the user is prompted with a security warning stating that the macros have been disabled. In order to convince the target to enable the execution of the malicious code, the attackers add a message in the document which states that macros have to be enabled in order to access its content. This message represents the social engineering component of the attack. If the user is tricked into enabling the malicious macros, these will perform unwanted actions, such as encrypting a user's data, downloading and executing other malicious files, steal data, etc.

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.

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. Our system is based on a derived version of the perceptron, OSC[5]. The system extracts the macros within a file, applies a lexical processing, extracts the sequence of features and feeds them to the machine learning algorithm. In this way, we provide a complete

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

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 into an AV solution. 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 atomic element of the VBA language, such as a keyword, a variable name, a constant value or an operator. 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. Based on our analysis, these type of samples produces a very small numer of features which is not sufficient for evaluating their nature.

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 building a fast evaluation method. A dynamic feature extraction procedure would imply the execution of a sample in a controlled environment in order to observe its behaviour, which is very time consuming. By processing the strings found in the code illustrated in Listing 1, we can observe the presence of the "mshta" utility[7] and the existence of an URL.

```
1 Sub Document_Open()
2
3 Shell "mshta javascript:""\textbackslash..\textbackslash
      ,RunHTMLApplication "";GetObject(""script:http:/" +
       Replace(abadondend, "\textbackslash", "/") + """);
      close();"
4
5 End Sub
```

Listing 1. Example of malicious code inside a macro

[7]The Mshta.exe utility allows the execution of Microsoft HTML Applications. These are standalone applications which execute using the same models as Internet Explorer, but outside the browser. They can include malicious VBScript or JScript code.

If we correlate these observations with the presence of the "Shell" command, we can extract the feature "MSHTA_REMOTE_EXECUTION". In this example, we can also extract a feature which counts the number of strings in a concatenation operation. 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. By analyzing samples from both categories and developping processing methods we extracted 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) and also 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. The discretization process of a feature involves creating a set of intervals which enclose its values. The inclusion of a value within one of these intervals will constitute a new feature. The process used for discretizing the values of a feature aimed to create intervals which include elements whose frequency and value are very close. For example, we observed that some malicious samples have a higher number of small functions than 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-functions"

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[6]. The goal of this process is to select a subset of features which carries as much information as possible while minimizing the information shared between these features in order to eliminate redundant information.

V. Results

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 in the context of an AV solution because the detection of a benign sample can have serious consequences on a user's working environment. Consequently, we chose to use a modified version of the Perceptron algorithm named OSC, which incorporates an extra stage in the training algorithm to ensure that all the samples marked as benign are correctly classified. At each iteration, the OSC version of the Perceptron algorithm performs an extra training procedure for the records of a certain class, which is finished when the number of falsely classified entries from that class drops to a certain quota. In our training, this minimization process is applied over the class of benign samples and the quota is set to zero. 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, with two discretization versions. The first dicretization method is the one mentioned in section IV and the second method is a modified version of the first method, which produces larger intervals for a certain feature. 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. Using the first discretization method, we trained our model (OSC_{D1}) for 500 iterations and we developed an accuracy of 99.68% and sensitivity of 98.73%.

The second approach, 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 (OSC_{UD1}) yielded an accuracy of 99.65% and sensitivity of 98.25%.

We have also experimented with a slightly different discretization method which has the effect of increasing the length of the intervals associated with the features. Using this new method, we trained another two models (OSC_{D2} and OSC_{UD2}) for 500 iterations. The (OSC_{D2} model has developed an accuracy of 97.72% and sensitivity of 98.91% and the OSC_{UD2} had an accuracy of 99.69% and sensitivity of 98.44%. We specify that the filtering process used for OSC_{UD2} reduced the training dataset to 14255 samples, 11,493 benign and 2762 malicious.

We constructed a test set of samples in order to compare our two proposed models. We selected 8668 benign samples and 10,000 malicious samples. The results of these tests can be observed in the following table:

Name	Sensitivity	Accuracy
OSC_{D1} — 500	98.73%	99.68%
OSC_{UD1} — 500	98.25%	99.65%
OSC_{D2} — 500	98.91%	99.72%
OSC_{UD2} — 500	98.44%	99.69%

Table II
Comparative training results

Name	FP	FP Rate	TP	Sensitivity
OSC_{D1} — 500	23	0.26%	9231	92.31%
OSC_{UD1} — 500	16	0.18%	9194	91.94%
OSC_{D2} — 500	28	0.32%	9299	92.99%
OSC_{UD2} — 500	9	0.10%	9037	90.37%

Table III
Results on test dataset

Considering these results, we can state that there is no model that can be considered the best. Although the models built using the second discretization method have a higher accuracy and sensitivity in the testing phase, they perform worse in terms of FP rate (OSC_{D2} has a higher FP rate than OSC_{D1}) or sensitivity (OSC_{UD2} has a lower sensitivity than OSC_{UD1}) on the test dataset. Taking into account these results, we can state that the choice of "the best model" means making a trade-off between the FP rate and the sensitivity and one must choose a model to best fit his needs. Due to the fact that one of our goals is to construct a model that has a low FP rate, the best model in our case is OSC_{UD2}.

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.

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] S. Kim, S. Hong, J. Oh, and H. Lee, "Obfuscated VBA macro detection using machine learning," in 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018, Luxembourg City, Luxembourg, June 25-28, 2018, 2018, pp. 490–501. [Online]. Available: https://doi.org/10.1109/DSN.2018.00057

[2] E. Aboud and D. O'Brien, "Detection of malicious VBA macros using machine learning methods," in Proceedings for the 26th AIAI Irish Conference on Artificial Intelligence and Cognitive Science Trinity College Dublin, Dublin, Ireland, December 6-7th, 2018., 2018, pp. 374–385. [Online]. Available: http://doras.dcu.ie/22879/1/aics_34.pdf

[3] S. de los Santos and J. Torres, "Macro malware detection using machine learning techniques - A new approach," in Proceedings of the 3rd International Conference on Information Systems Security and Privacy, ICISSP 2017, Porto, Portugal, February 19-21, 2017., 2017, pp. 295–302. [Online]. Available: https://doi.org/10.5220/0006132202950302

[4] N. Nissim, A. Cohen, and Y. Elovici, "ALDOCX: detection of unknown malicious microsoft office documents using designated active learning methods based on new structural feature extraction methodology," IEEE Trans. Information Forensics and Security, vol. 12, no. 3, pp. 631–646, 2017. [Online]. Available: https://doi.org/10.1109/TIFS.2016.2631905

[5] D. A. Dragoş Gavriluţ, Mihai Cimpoeşu and L. Ciortuz, "Malware detection using perceptrons and support vector machines," in Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/5359599/

[6] F. Fleuret, "Fast binary feature selection with conditional mutual information." [Online]. Available: http://jmlr.csail.mit.edu/papers/v5/fleuret04a.html