Detection of malicious Office documents using Machine Learning Techniques

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

***Abstract*—In recent years, VBA macro malware has been growing in popularity and so has the necessity of detecting it. While 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 API. 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 response to these measures, the attackers distribute the malicious documents through spear-phishing emails and use social engineering techniques in order to persuade the victim to enable the execution of the malicious macros.**

This paper aims to explain the current state of detections regarding these types of malicious files and analyzes the possibility of improving existing macro detections via machine learning techniques. The new detection method is based around the Perceptron algorithm which builds a detection model focused on the properties of the macro code extracted from the VBA project of Office files. The paper also explores the advantages of using features that are not based around a particularity of the malicious file.

***Keywords*-Machine learning, feature extraction, feature selection, VBA macro, malicous macros**

I. Introduction

Introduction

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 approaches to detect malicious files. The main focus of these approaches are the structural properties of the macro code that derive from obfuscation or general 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 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 the 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%), Bernoulli Naive Bayes (89,1%).

A similar approach to macro detection is described by Ed Aboud and Darragh OBrien. They are using features based on the macro found inside Office files, but these features 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 attack such as Enable Content and Previous Version. The images which contain these phrases are extracted by searching image magic numbers in the OLE binary. Using the extracted features, 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: Random Forest (98,9875%), KNeighbors (97,527%), DecisionTree (98,225%), GaussianNB (97,02%). In Macro Malware Detection using Machine Learning Tech-

niques, Sergio De los Santos and Jose Torres propose using the features extracted by the python libraries OleFile and OleVBA. They establish a feature vector composed of 45 features, but they highlight the fact that these features are not distinct, some being obtained by discretizing features that are not boolean in nature. Their system was trained uses the following classifiers with the following accuracy values during 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 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).

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 mention 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.

Related work

III. Problem description

problem

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 a 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 kept all the bening samples and selected 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 characteristic 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)

By discretizing the values, from an initial set of 536 features, we arrived at a feature vector close of 2977 properties. By design, our approach will use only 256 features from the initial 3000. 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. Exemplu de ciat [1] Db and features

V. Results

Lorem ipsum, result1

VI. Conclusion

Conclusion

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/