Learning Tool for Applying Static Vulnerability Analysis of Office Documents Based On Automatically Extracted Information from the Associated Macro Code

Radu Rădescu[1], Mălina Andreea Roșu[1]

(1) University POLITEHNICA of Bucharest,
Faculty of Electronics, Telecommunications and Information Technology,
Applied Electronics and Information Engineering Department,
313, Splaiul Independenţei, Sector 6, RO-060042, ROMANIA
E-mail: radu.radescu[at]upb.ro

Abstract

The idea from which this paper started is the interest for the learning part of the security elements of Office documents, but also the one related to the main vectors of infection of ordinary users. This field is constantly evolving and therefore learning to use any automated tool can be useful for both users and those who work actively with malware components to make their work easier. The notion of machine learning implies the need for minimal human interaction in order to achieve certain objectives presented from the beginning. The paper aims to present the steps taken in implementing and learning the application of an automatic tool for analyzing Office documents based on information automatically extracted from the macro code associated with them. To complete the implementation of this tool, notions of Machine Learning were used, but the learning process involves the accumulation of experience in malware analysis, notions of programming in the Python language and much other IT knowledge.

Keywords: Static Vulnerabilities, Cyber-attack, Educational Application, Automatic Platform, Security Systems, Advanced Technologies, Machine Learning, Decision

1 Introduction

Compound File Binary Format (CFBF) files are those that are based on storing multiple file structures (streams) inside a single parent file, as is the case with Office files. Normally, a regular user recognizes the type of files by their extension, but a common way of infecting it is to change the file extension in order to fool the user and influence him to open a malicious file faster. Many documents come in the form of images, although they are not basically images, but executable files or malicious documents.

This is possible due to the fact that the operating system runs the files in different programs, depending on the file signature or its "magic number", after which any type of file is recognized by the computer. This field is usually represented by the first bits of the file that are used by the operating system to know with which program to open a file.

This paper focuses primarily on specific Microsoft Office formats for Word and Excel. Macro code is essentially a piece of code written in the Visual Basic for Application (VBA) programming language. Its original purpose was to help automate certain tasks in a document, such as opening a new sheet in Excel, while recalculating a formula.

2 Malware and Solutions

2.1 The Concept of Malware

The term *malware* comes from the combination of the words malicious and software. These two words mean malicious software, and by definition malware is a type of software program designed to damage or infiltrate a computer, and/or damage or infiltrate an entire computer network without the owner's consent. Malware files are classified according to their purpose and how they are used to achieve this purpose (Malware1, 2020). There are many types of malware,

but some of them are considered *grayware*, i.e., applications or files that are not considered exactly malware, but which can have quite pronounced malicious influences (Malware1, 2020).

2.2 Ways of Malware Spreading and Methods of Prevention

When it comes to prevention, a common idea, regardless of the type of malware targeted, is very important that any device has an antivirus installed. This action must be done right from the first use, to run on the device a software program specialized in anti-malware protection because it can be infected from the first browsing on the internet or when running a program. This software should always be used to scan any email, disk, or new file before saving and running it on the device.

Another way to protect the computer from malware attacks is to prevent the user from using an account with administrator privileges for daily use because the malware files work with the account privileges from which they are installed. There are malware that can gain this privilege regardless of the account it was run with, but most fail to do so.

One thing that facilitates the spread of malware is that most users prefer to use pirated versions of applications and even operating systems. Unfortunately, they do not come with various types of security exploits and they do not have all the prevention methods that the original software has, thus making both infection and spread much more likely.

Depending on how certain types of malware files spread and act, prevention can be done in a more specific way for each (Malware1, 2020).

Any device infected with malware will give various signs that may vary depending on different aspects such as the type of virus, the specifications of the device and how to use it. Some types of malware, such as *ransomware*, for example, are highly visible and even aim to make the infection obvious. These usually come with a prompt notifying the user that they have been infected and the steps they need to follow to receive the decryption key needed to recover the files.

Other types of malware, on the other hand, aim to spread as quickly and widely as possible, so those who design them want them to run in the most invisible way possible. At the base of this desire is both the idea of spreading, because practically every device is like a node of the network responsible for spreading the file, but also for reasons to combat it.

Once they find out about the existence of a new type of malware, it doesn't take long for specialized institutions and companies to find ways to remove and publicly expose the file. Regardless of the structure of the malware file and the purpose of the attacker, signs of the fact that a device has been infected will always exist. Some are obvious, and others will be noticed only by those who have in-depth knowledge of the systems (Malware1, 2020). The impact that viruses and implicitly cyber-attacks have on both ordinary people and companies is much greater than anyone can imagine (Malware2, 2020).

3 The Structure of Office Documents and the Magic Number

Compound File Binary Format (CFBF) files are those that rely on storing multiple file structures / streams within a single parent file, as is the case with Office files. There are mainly two major types of file structures. Office: with binary file structure and archive structure containing XML documents (OOXML, 2020).

Normally, a regular user recognizes the type of files by their extension, but a common way of infecting it is to change the file extension in order to fool the user and influence him to open a malicious file faster. Many documents come in the form of images, although they are not exactly images, but executable files or malicious documents. This is possible due to the fact that the operating system runs the files in different programs, depending on the signature of the file or its "magic number" (Magic Number, 2020).

Any type of file is recognized by the computer by the signature of the file or the magic number. This field is usually represented by the first bits of the file and is used by the operating system to know with which program to open a file. In this paper we will focus mainly on specific Office formats for Word and Excel.

4 The Macro Code and Its Malware Component

Malware found multiple vulnerabilities, which they exploited. The malicious behavior comes from the fact that the existing activity in the mail servers has increased exponentially. Another very serious problem was that before the macros in a document were run automatically, so that users were infected by simply opening the document. This changed with the introduction of the enable/disable property. This means that macros (Macro, 2020) are disabled by default, and to activate them the user must follow an optional step to request this. Through VBA code, a document can interact with the entire system, having access to the console, PowerShell, internet connection, DLL upload, etc., so accessing system functions, all through a few lines of code. Because such code is very easy to understand and therefore detectable by AV companies, malware developers have resorted to other methods in order to make the code more complex and difficult to understand. One of these methods is obfuscation. Obfuscation is the deliberate act of making it difficult to understand a code, by replacing it with confusing and ambiguous language, without changing its functionality.

There are several types of obfuscation:

- By changing the names of functions and variables with random strings. This type of obfuscation is mainly aimed at fooling AV programs that have some generic rules for searching for word combinations such as: PowerShell, HTTP and download, for example, blocking the running of that file.
- By separating commands/words and linking them by variable names and join operators.
- By using coding algorithms or operations such as replace or shift.

5 The Classification Algorithms

The term *machine learning* is very well known and common, especially in recent years. Machine learning is a branch of artificial intelligence that aims to learn computers, based on various mathematical calculations, to draw certain conclusions automatically, without human influence.

In this paper, the results of four different algorithms were compared:

- NB (Naive Bayes) is a family of probabilistic classifiers based on the well-known Bayes theorem, which determines the probability of belonging of events and objects to a particular class.
- SVM (Support Vector Machine) builds a spatial representation of the examples from the training phase and tries to delimit them on classes from a spatial point of view with a demarcation area, as large as possible, in order to make the classification as obvious and easy as possible.
- MLP (Multilayer Perceptron) is part of artificial neural networks and uses the backpropagation algorithm. It consists of three layers of neurons – the input, the output and the hidden layer, placed between the input and output.
- RF (Random Forest) is an algorithm based on decision trees formed by several branches of type (if ... then ...), whose representation suggests the tree branches.

6 The Implementation Based On Automatic Macro Code Extracted Information

The final result of this paper is a learning tool implemented in the Python programming language, based on Spyder, an open source cross-platform IDE, implemented specifically for the

Python programming language, which has at the entry an Office document and at the exit a malware/clean verdict obtained following the various automatic classifications made by it.

6.1 Choice of the Document Subject to Classification

The document for which the classification is to be made must meet several important criteria for the final result to be correct:

- It must contain undamaged VBA macro code;
- It must not contain characters other than UNICODE (Unicode, 2020) because various functions are used inside the code that cannot interpret characters.

6.2 Extracting the Macro Code

A lot of tools have been implemented to extract the macro code. Among the best known are Office Mal Scanner (Reconstructer, 2020) and OleDump (OleDump, 2020) . Both are written in Python, which is the main reason why this programming language was chosen for the development of this static analysis tool. OleDump is a tool that analyzes CFBF files, and from the result of running this tool on an XLS file, depending on the parameters used, the VBA code can be extracted for analysis.

This tool is most used for automatic analysis of Office files. For this reason, a special Python package with its functionalities, called *oletools* (OleTools, 2020), has been implemented. The function used to extract the macro code calls two functions from the *oletools* package:

- VBA_Parser – it is used to parse the document from the disc. Before that, it can be seen that it was read with the *open()* function using the 'rb' – read bytes parameter.
- Extract_macros – it returns the VBA macro code as text, then written inside a text file to be further processed.

The exact reason why all macro codes extracted from the document are transformed into text files is mainly about security. Because it works with malicious code, it is very possible that if the .vba extension is kept, at the simple click of a button it will be executed and the computer will be infected.

6.3 Preprocessing the Macro Code

There are several steps in which the macro code is processed, including the classification algorithms in the following steps. At this stage, however, preprocessing refers to the fact that if several different files with VBA code are extracted from a document, they will be concatenated into one and it will be further introduced in the classification algorithms, depending on its characteristics. Another preprocessing step is the immediate change of the file extension extracted from .vba to .txt, both for security criteria (see above), but also to be easily read by the tool using the *read()* function.

6.4 The Algorithm for Obfuscated/Non-obfuscated Classification

The dataset was retrieved from (Virustotal, 2020), using the following search criteria:

- For the obfuscated documents, the documents received from 2018 until now were chosen, which contain the tags "macros" and "obfuscated". To ensure the selection of really obfuscated files, the criterion "detection no." > 20 has been added.
- For non-obfuscated documents, documents received in the same period were chosen, which contain the tag "macros" and do not contain the tag "obfuscated". To ensure the selection of truly non-obfuscated files, "detection no." was added.

Following these queries, files from Table 1 were selected.

Table 1. The data set for the obfuscated/non-obfuscated classification algorithm

	Nr. of files	Nr. of macros	Nr. of unique macros	Dimension
Obfuscated	500	865	341	13 KB – 1.52 MB
Non-obfuscated	300	685	349	50 KB – 3.5 MB

The VBA macro code is based on a code in a programming language so, in general, it is represented by text. Mainly, classification algorithms work with numerical values. For this reason, a way must be found to convert the text into numbers. To do this, some distinctive features were extracted between the obfuscated and the non-obscured code (Sangwoo et al, 2020):

- Text entropy, a very useful criterion for detecting random obfuscation. In the case of VBA macro code analysis, the analyzed events are actually the appearance of each character in the code text. The entropy values are in the range [0.8]. The higher the entropy, the more likely the file is to be obfuscated (non-obfuscated files generally have entropy values between 4.5 and 5). (Entropy, 2020)
- Average number of words per line – a code with more than 100 characters per single line is often obfuscated (Obfuscation, 2020).
- The average number of characters in each word – when there are very long words in the code, they are either coded portions of code or the code is obfuscated by the technique of renaming variables or functions.
- Total code length – when the code hides many features or other malicious portions of code it may be larger than others.
- Number of special characters – punctuation marks, numbers or symbols.
- Number of functions used in obfuscation techniques for text manipulation.
- Number of arithmetic functions – widely used in various types of obfuscation.
- Number of character or string-level conversion functions – used for different types of encoding, encryption or obfuscation.

The basic idea behind the implementation of the classification algorithm (Sangwoo et al, 2020) is to choose two sets of features extracted from the VBA macro code of an Office document database divided into two classes: obfuscated/non-obfuscated in order to implement automatic algorithms of machine learning.

Based on this data set, the four algorithms mentioned above were tested to make a comparison in terms of classification accuracy on the selected database. The results obtained are presented in Table 2.

Table 2. Accuracy values obtained by classification algorithms

Algorithm	Accuracy
Naive Bayes	73.39%
SVM	43.84%
MLP	73.89%
RF	98.52%

It is easy to see that the RF algorithm had the best results. For this reason, this will be the algorithm used in the classification, the accuracy of over 98% being a very satisfactory one. The implementation of the obfuscated/non-obfuscated classification algorithm contains two distinct parts, represented by the two main parts, the training part and the prediction part. These are very

similar, but there are some minor differences. The criteria for extracting the feature vector are obviously identical. The only difference is that there is no second associated label vector, because the purpose of the classification step is to classify the file in one of the two categories obfuscated/non-obfuscated.

6.5 The Algorithm for Malware/Clean Classification

In this case, unlike the case of obfuscated files, in which we concluded quite directly that they are malicious, now we can no longer do so. There are malware files that are not obfuscated and use various other techniques to trick the user or antivirus software. These are as dangerous as the obfuscated ones, so they must receive the same attention as the others. In the same category are the clean files, which are written clearly for the obvious purpose of automating tasks or completing certain activities (extracts, reports, etc.).

The dataset was retrieved from (Virustotal, 2020), using the following search criteria:

- In order to find the clean documents, the time interval from 2018 until now and the “macro” tag were chosen. To make sure the files are really clean the criterion "detection no." > 2 was added.
- To find the malicious documents the same period and the same “macro” tag were chosen. To make sure that the files are really malware, "detection no." > 25 was added.

The Bag of Words technique (Bag of Words1, 2020), (Bag of Words2, 2020) fits very well the classification of non-obfuscated documents because they, being written clearly, the words in the macro have exactly the meaning they have in terms of functionality. This may not be the case with obfuscated files, where function names are hidden under different names just to camouflage their functionality and therefore their malicious intentions.

The Bag of Words technique can be used in a lot of other circumstances, which is why there are a multitude of libraries that have implemented this functionality in a single line of code – such as the *sklearn* library – CountVectorizer. This technique aims to turn words into numbers, so that they can then be used by different classification algorithms.

The steps of the Bag of Words algorithm are:

Step 1 – Divide each sentence into words;

Step 2 – Create a dictionary with the frequency of occurrence of each word.

Step 3 – Create the model: it represents the implementation of a matrix that has as columns the words chosen in the previous step, after filtering the words used for analysis (using the *regular expression* function, keeping only letters in the code and turning them all into lowercase letters), and as lines each sentence.

Since we obtained very good results with the classification algorithms in the previous section, we decided to try for the first time the same ones, and if the results are not satisfactory, to try others. From our point of view, an accuracy of 90% or more is desirable.

Table 3. The values of the accuracy obtained by each classification algorithm

Algorithm	**Accuracy**
Naïve Bayes	83.14 %
SVM	85.39 %
Linear SVM	85.39 %
MLP	91.01 %

RF	87.64 %

As can be seen, in this situation there are not such big differences when comparing the values of the obtained accuracy, but the best value was obtained using MLP, which fulfills the criterion initially proposed. In conclusion, MLP will be used for the final malware/clean decision algorithm. The feature analysis part is, in short, a comparison between the features extracted in the training phase and those existing in the given file as input.

To make this comparison, the aforementioned CountVectorizer function was also used, but for the vocabulary parameter the already extracted vocabulary was given as a reference. Based on this step, the algorithm classifies the input file into one of the two clean/malware classes.

If, after classification, the document is considered obfuscated, it is automatically concluded that it is also malware. Obfuscation is also used in clean applications because those who implement applications want to hide the source code to avoid copying it or even using it for malicious purposes. This is mainly the case with PE applications, but in the case of documents, applications that can be implemented using VBA code cannot issue copyright claims because their functionalities cannot be compared to those of an executable. In the case of Office documents, the desire to obfuscate hides malicious intentions in 99% of cases.

8 Conclusions

The paper presents the steps taken to implement a tool for learning to apply static analysis of the vulnerability of Office documents based on information automatically extracted from the associated macro code. In most cases, the obfuscated code is also malicious, and the experimental results of the paper prove the detection efficiency by applying the detection and protection methods proposed in this paper.

References

Sangwoo, K., Hong, S. K., Oh, J. and Lee, H. (2018): *Obfuscated VBA Macro Detection Using Machine Learning*. Licentiate of engineering thesis: Korea University, Seoul.
Virustotal: https://www.virustotal.com, accessed 2020
Entropy: https://www.cyberbit.com/blog/endpoint-security/malware-terms-code-entropy, accessed 2020
Obfuscation: https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1492186586.pdf, accessed 2020
Unicode: https://unicode-table.com/ro/#unified-canadian-aboriginal-syllabics, accessed 2020
Reconstructer: http://www.reconstructer.org/, accessed 2020
OleDump: https://blog.didierstevens.com/programs/oledump-py/, accessed 2020
Bag of Words1: https://stackabuse.com/python-for-nlp-creating-bag-of-words-model-from-scratch, accessed 2020
Bag of Words2: https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-în-python-for-nlp-282e87a9da04, accessed 2020
Phishing: https://resources.infosecinstitute.com/category/enterprise/phishing/#gref, accessed 2020
OleTools: https://github.com/decalage2/oletools/wiki/olevba, accessed 2020
Malware1: https://www.gdatasoftware.com/blog/, accessed 2020
Malware2: https://opendatasecurity.io/how-much-does-a-cyberattack-cost-companies/, accessed 2020
Magic Number: https://www.filesignatures.net/index.php?search=zip&mode=EXT, accessed 2020
OOXML: https://www.oxygenxml.com/xml_editor/ooxml_office_2007.html, accessed 2020
Macro: https://insights.sei.cmu.edu/cert/2016/06/who-needs-to-exploit-vulnerabilities-when-you-have-macros.html, accessed 2020