Electronic Preprint for Journal of Information Processing Vol.27

Regular Paper

Detecting Unseen Malicious VBA Macros with NLP Techniques

Mamoru Mimura[1,a)] Hiroya Miura[1]

Received: December 10, 2018, Accepted: March 5, 2019

Abstract: In recent years, the number of targeted email attacks which use Microsoft (MS) document files has been increasing. In particular, malicious VBA (Visual Basic for Applications) macros are often contained in the MS document files. Some researchers proposed methods to detect malicious MS document files. However, there are a few methods to analyze malicious macros themselves. This paper proposes a method to detect unseen malicious macros with the words extracted from the source code. Malicious macros tend to contain typical functions to download or execute the main body, and obfuscated strings such as encoded or divided characters. Our method represents feature vectors from the corpus with several NLP (Natural Language Processing) techniques. Our method then trains the extracted feature vectors and labels with basic classifiers, and the trained classifiers predict the labels from unseen macros. Experimental results show that our method can detect 89% of new malware families. The best F-measure achieves 0.93.

Keywords: VBA macro, machine learning, natural language processing technique, bag-of-words, Doc2vec, TFIDF

1. Introduction

In recent years, email has become one of the most popular communication tools. This situation has led to targeted email attacks becoming a big threat to society. A targeted email attack is a specific attack in which the attacker attempts to persuade a victim to run specific action. Depending on the specific action, there are two types of targeted email attacks. One is to open malicious links and download a malicious program, and the other is to open malicious attachments. Attackers attempt to earn credibility with their victims through an eloquent mail text. Moreover, the attackers convince victims to unknowingly download a malicious file or click-through to a malicious site. According to a report published by Sophos [1], malicious attachment files are used in most targeted email attacks. The report shows that 85% of the attached files are Microsoft Office (MS) document files. Furthermore, most of the MS document files contain malicious VBA (Visual Basic for Applications) macros. VBA is an implementation of Microsoft's programming language Visual Basic 6, and built into most Microsoft Office applications. Malicious macros have a long history. For example, the LOVELETTER worm, one of the most famous malicious macro infected more than 45 million computers, and some organizations suffered serious damage in 2000. Subsequently, malicious macros gradually faded out. They grew in popularity again with the rise of targeted email attacks. Macros are a powerful tool to automate common tasks in MS document files. However, malicious macros use this functionality to infect the computer. In targeted attacks, attackers often use unseen malicious macros which are not detected by anti-virus programs with the latest definition. In general, anti-virus programs require virus pattern files, and the pattern files have to be updated. Most attackers, however, obfuscate programs to evade detection. Therefore, it is difficult to detect unseen malicious macros that contain new malware families.

To detect unseen malicious macros, some previous methods can be applied. For instance, Nissim et al. [2] analyzed the structure of docx files, and proposed a method to detect malicious docx files. These previous methods detect malicious MS document files. These methods, however, do not discriminate between malicious macros and benign macros. Hence, if a MS document file contains benign macros, these methods might detect the benign macros as malicious ones. If the malicious MS document file is camouflaged with the structure of a benign MS document file, the attacker can probably evade the detection method. Detecting malicious macros themselves can overcome these weaknesses. However, there are a few methods to analyze malicious macros themselves [3], [4]. Hence, there is room for improvement on these methods.

This paper proposes a method to detect unseen malicious macros themselves. Malicious VBA macros tend to contain typical functions to download or execute the main body, and obfuscated strings such as encoded or divided characters. To investigate the source code, we focus on NLP (Natural Language Processing) techniques. NLP techniques are usually used to analyze natural languages as the name indicates. In this paper, we presume VBA macros are written in a natural language, and attempt to learn the difference between benign and malicious VBA macros with Doc2vec. Doc2vec is an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts. Then we input the extracted feature vectors with the label into supervised learning models to classify benign and malicious VBA macros. The key idea of this research is reading

[1] National Defense Academy, Yokosuka, Kanagawa 239–8686, Japan
[a)] mim@nda.ac.jp

VBA macros as a natural language. To the best of our knowledge, Doc2vec has never been applied to detecting malicious VBA macros. Doc2vec enables extracting feature vectors from VBA macros automatically. That is why we focus on Doc2vec.

Our method uses some NLP techniques to investigate the macro's source code. Our method divides the source code into words, and represents feature vectors from the corpus with several NLP techniques. Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TFIDF) are used to select important words for classification. TF is a simple method which weights the value corresponding to frequency of words in a corpus. TFIDF is a more sophisticated method which weights a representative word in a corpus. Bag-of-Words (BoW) and Doc2vec represent feature vectors from the corpuses. BoW is a basic method that represents vectors corresponding to the frequency of the words. Doc2vec is a more complicated model that represents vectors from the context of the documents. Our method then uses the extracted feature vectors and labels, and trains basic classifiers, Support Vector Machine (SVM), Random Forests (RF) and Multi Layer Perceptron (MLP). Finally, the trained classifiers predict the labels from unseen macros. Experimental results show that our method can detect 89% of new malware families. The best F-measure achieves 0.93.

This paper addresses 4 research questions as follows.

(1) Does our method detect unseen malware families?
(2) Does Doc2vec represent feature vectors effectively?
(3) What is the best combination of these NLP techniques and classifiers?
(4) Does TFIDF select important words to classify macros?

In order to address these questions, we conduct some experiments. Based on the results, this paper makes the following contributions:

(1) Propose a method to detect unseen malicious macros which contain new malware families [5].
(2) Doc2vec is effective in classifying malicious macros [6].
(3) Linear classifiers are effective for Doc2vec [6].
(4) Reducing words using Term Frequency is effective for classifying macros [6].

We will introduce the structure of this paper. Section 2 introduces related work and reveals the differences between this paper and other relevant study. Section 3 describes malicious VBA macros, and Section 4 presents some NLP techniques. Section 5 proposes the method, and Section 6 conducts experiments. Section 7 discusses the results, and finally, describes conclusion.

2. Related Work

In targeted email attacks, attackers use attachment files which contain malicious codes. Methods to detect these malicious files can be categorized into static analysis and dynamic analysis. Our method does not execute the MS document files or VBA macros. Therefore, we focus on static analysis in this section. The attachment files are mainly categorized into executable files and document files. Our method investigates document files. The document files are roughly categorized into MS document files and PDF files. VBA macros are embedded in MS document files. We will show the details in the followings.

2.1 MS Document File

Nissim et al. proposed a framework (ALDOCX) that classifies malicious docx files using various machine learning classifiers [2]. ALDOCX creates feature vectors from the path structure of docx files. Naser et al. proposed a method to detect malicious docx files [7]. The method parses the structure of docx files, and analyzes suspicious keywords. These methods do not support Compound File Binary (CFB) file format. Our method, however, supports MS document files which conform to Compound File Binary (CFB) file format and Office Open XML (OOXML) file format.

Otsubo et al. proposed a tool (O-checker) to detect malicious document files (e.g., rtf, doc, xls, pps, jtd, pdf) [8]. O-checker detects malicious document files which contain executable files, using deviation of file format specifications. Boldewin implemented a tool (OfficeMalScanner) to detect MS document files which contain malicious shellcodes or executable files [9]. The tool scans entirely malicious files, and detects features of strings of Windows API, shellcode patterns, and embedded OLE data. This tool scores each document corresponding to each of the features. If the scores are more than a threshold, this tool judges the file as malicious. Mimura et al. proposed a tool to deobfuscate embedded executable files in a malicious document file (e.g., doc, rtf, xls, pdf) and detect them [10]. These methods focused on embedded malicious executable files or shellcodes, and do not detect malicious macros. Our method on the other hand, detects malicious macros.

2.2 VBA macro

There are a few methods to detect malicious macros. Bearden et al. proposed a method of classifying MS Office files containing VBA macros as malicious or benign using the K-Nearest Neighbors machine learning algorithm, feature selection, and TFIDF where p-code opcode n-grams compose the file features [3]. This study achieved 96.3% file classification accuracy. However, the samples were only 40 malicious and 118 benign MS Office files. This paper provides more reliable results with thousands of distinct samples. Kim et al. focused on obfuscated source code and proposed a method to detect malicious macros with a machine learning technique [4]. This method extracts feature vectors from obfuscated source code. Therefore, this method might not detect malicious VBA macros which are not obfuscated. Our method uses not only features in obfuscated macros, but also other features.

2.3 PDF File

Some researchers deal with the detection of malicious PDF files. For instance, Igino Corona et al. proposed a method to classify malicious files according to the frequency of suspicious reference APIs [11]. Liu et al. proposed a method which analyzed obfuscated scripts to classify malicious PDF files [12]. This method uses the characteristics of obfuscation, which is common to our method. These methods classify malicious PDF files. Our method investigates MS document files and detects malicious macros.

Table 1 Typical functions in malicious macros.

Downloader	Dropper
`CreateObject` function	`CustomProperties` collection
`Shell` function	
`SendKey` statement	
`Declare` statement	

Table 2 Typical obfuscation methods.

#	summary
1	Replace statement name, etc.
2	Encode and decode with ASCII code
3	Use XOR
4	Split characters
5	Use reflection functions

3. Malicious VBA Macro

3.1 Behavior

This section describes the behavior of malicious macros, and reveals their features. Attackers use a slick text of the type that the victim expects, and induces the victim to open an attachment. When the victim opens the attachment and activates the macro, the macro compromises the computer. There are two types of malicious macros, Downloader and Dropper.

Downloader is a malicious macro which forces the victims computer to download the main body. When Downloader connects to the server, it tends to use external applications. Finally, the computer downloads and installs the main body from the server.

In contrast, Dropper contains the main body in itself. When a victim opens the attachment of a phishing email, Dropper extracts the code and executes it as an executable file. The difference between Dropper and Downloader is that Dropper contains the main body in itself. Therefore, Dropper can infect victims without communicating with external resources.

Malicious macros tend to contain functions to download or extract the main body in the source code. Hence, our method attempts to detect these features.

3.2 Typical Function

To detect these features, we focus on typical functions described in the source code. **Table 1** shows the typical functions which are frequently described in Downloader and Dropper. `CreateObject` function returns a temporary object which is part of an external application function. For example, if `CreateObject` function parses "InternetExplorer.Application" as an argument, the function accesses Internet Explorer. `Shell` function enables to execute an argument as a file name. `SendKey` statement and `Declare` statement also appear in the Downloader. `SendKey` statement sends keystrokes to the active window as if typed at the keyboard. Attackers use `SendKey` statement with `Shell` function to execute any commands. `Declare` statement is used to declare references to external procedures in a dynamic-link library. The `Declare` statement allows accessing a variety of functions. `CustomProperties` represents additional information, and the information can be used as metadata. Attackers frequently use `CustomProperties` collection to conceal malicious binary code.

3.3 Obfuscation

Most malicious macros are obfuscated to prevent analysis. Therefore, capturing obfuscated strings is an effective method for detecting malicious macros. We will show some obfuscation techniques of the source code.

Table 2 shows typical obfuscation techniques in malicious macros. Method 1 replaces class names, function names et al. with random strings. The random strings tend to be more than 20 characters. Method 2 encodes and decodes strings with ASCII code. VBA macros provide AscB function and ChrB function. AscB function encodes characters to ASCII codes. ChrB function decodes ASCII codes to characters. Method 3 encodes and decodes characters by XOR operation. Method 4 divides a string into characters. The divided characters are assigned to variables. By adding together those variables, the original string is restored. Method 5 uses reflection functions which execute the strings as instructions. These strings contain function names, class names and method names. Attackers often conceal malicious functions with these techniques.

4. NLP Technique

4.1 Bag-of-Words

BoW is a basic method to extract feature vectors from a document. BoW represents the frequency of a word in a document, and extracts matrix from documents. In this matrix, each row corresponds to each document, and each column corresponds to each unique word in documents. This method does not consider word order or meaning. In this method, the number of unique words corresponds to the dimension of matrix. Thus, the more number of unique words increases, the more the number of matrix dimensions increases. Therefore, methods to adjust the number of dimensions are required. To adjust the number of dimensions, important words have to be selected.

4.2 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency ($TFIDF$) is one of the most popular methods for selecting important words. We will introduce how a $TFIDF$ value is calculated.

$$TFIDF_{i,j} = frequency_{i,j} \times \log_2 \frac{D}{document\_frequency_i}$$

The $frequency_{i,j}$ (TF) is the frequency of a word i in a document j. The $document\_frequency_i$ is the frequency of documents in which the word i appears. The IDF is the logarithm of a value in which D (the number of total documents) is divided by the $document\_frequency_i$. $TFIDF$ value is a value which is a product of TF and IDF. Finally, $TFIDF$ values are normalized. In this model, if a word appears rarely in an entire corpus and appears frequently in a document, the TFIDF value increases.

4.3 Doc2vec

Doc2vec [13] is an extension of Word2vec [14]. First, we will introduce Word2vec. Word2vec is a model that is used to represent word embeddings. Word2vec is a two-layer neural network that is trained to reconstruct the linguistic context of words. Word2vec has a hidden layer and an output layer. The input of

Word2vec is a large corpus of documents, and Word2vec represents the input in feature vectors. The number of dimensions of the feature vector is typically several hundred. Each unique word in the corpus is assigned a corresponding element of the feature vector. Word vectors are positioned in the vector space such that common contexts in the corpus are positioned in close proximity to one another in the space. This is based on the probability of words co-occurrence around a word. Word2vec has two algorithms, Continuous Bag-of-Words (CBoW) and Skip-gram. CBoW is an algorithm which predicts a centric word from surrounding words. Skip-gram is an algorithm which predicts surrounding words from a centric word. Word2vec enables obtaining similarity of words, and also predict equivalent words. Doc2vec has two algorithms, Distributed Memory (DM) and Distributed Bag-of-Words (DBoW). DM and DBoW are extensions of CBoW and Skip-gram respectively. Doc2vec enables to obtain similarity of documents, and also extract feature vectors from documents.

5. Proposed Method

5.1 Outline

We proposes a method to detect unseen malicious macros with NLP techniques. Our method requires a language model and a classifier to detect malicious macros. **Figure 1** shows an outline of the proposed method.

In the training phase, our method requires both malicious and benign samples with the labels. Step 1 extracts words from labeled macros in MS document files. Step 2 selects important words and constructs a language model with the corpus. Then the extracts words are converted into feature vectors with the language model. Step 3 trains classifiers with the extracted feature vectors and labels.

In the test phase, our method investigates unlabeled samples. Step 1 extracts words from unknown macros in MS document files. Step 2 converts extracted words into feature vectors with the language model. Step 3 classifies the extracted feature vectors with the trained classifiers, and the predicted labels are obtained.

5.2 Extract Word

Our method extracts macros from MS document files with Olevba [15]. Olevba is open source software that can extract macros from MS document files. Then, our method divides the source code into words. Our method uses some special characters as the delimiter. **Table 3** shows the special characters.

Thereafter, our method replaces some patterns. **Table 4** shows the patterns.

These patterns frequently appear in malicious macros. Because most malicious macros are obfuscated with these methods [4], [6]. If our method does not replace these patterns, each word is handled as each feature respectively. These words, however, have a common context or meaning. Our method replaces these patterns into single words to improve accuracy.

5.3 Language Model

In the training phase, our method selects important words based on the TF or TFIDF values. Then, our method constructs

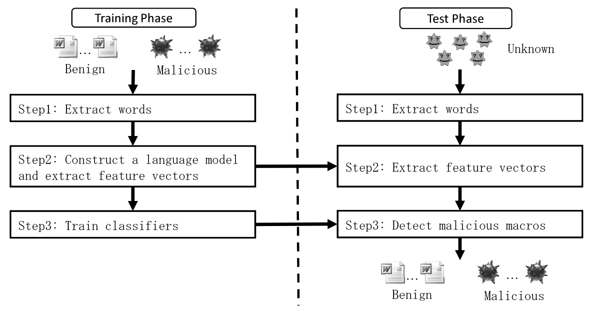

Fig. 1 An outline of the proposed method.

Table 3 Special characters as the delimiter.

special character	name	special character	name
”	double quote	+	plus
’	single quote	/	slash
{}	square bracket	&	and
()	round bracket	%	percentage
,	comma	¥	yen sign
.	period	$	dollar sign
∗	asterisk	#	sharp
-	hyphen	@	at mark

Table 4 The patterns to replace.

method	pattern	replaced word
1	Hexadecimal 1 (e.g., 0xXX)	0xhex
2	Hexadecimal 2 (e.g., &HXX)	andhex
3	Asc, AscB, AscW	asc
4	A string of 20 or more characters	longchr
5	A number of 20 digits or more	longnum
6	Element of array	elementofarray

a language model with the selected words. To construct a language model, our method uses BoW and Doc2vec. Thereafter, our method converts the words into feature vectors with the language model. In the test phase, our method uses the constructed language model, and converts the words extracted from unknown macros. Our method uses gensim-2.0.0 [16] to implement BoW and Doc2vec. Gensim has many functions related to natural language processing techniques. The Doc2vec model is trained 30 epochs with the DBoW algorithm. The number of dimensions is 100. These parameters are determined through a trial and error process.

5.4 Classifier

Our method uses the extracted feature vectors and labels, and trains the classifiers, Support Vector Machine (SVM), Random Forests (RF), and Multi Layer Perceptron (MLP). These classifiers are fundamental and often used for this field. Our method uses scikit-learn-0.18.1 [17] to implement SVM, RF, and MLP. Scikit learn is a machine learning library and has many classification algorithms. These classifiers use default values for all parameters.

6. Experiment

6.1 Environment

This section conducts experiments to evaluate our method. **Table 5** shows the environment. We implemented our method with Python 2.7 in this environment.

6.2 Dataset

To evaluate our method, we use actual malicious and benign

Table 5 Environment.

CPU	IntelCorei7 (3.30 GHz)
memory	32 GB
OS	Windows8.1Pro

Table 6 The numbers of samples.

2015's samples (Training)		2016's samples (Test)	
benign	malicious	benign	malicious
622	515	1,200	641

Table 7 The rates of malware families in 2015's samples.

	family	rate
1	O97M/Donoff	78.0%
2	O97M/Adnel	5.2%
3	O97M/Bartallex	4.3%
4	W97M/Adnel	3.3%
5	X97M/Donoff	2.8%
6	O97M/Madeba	0.9%
7	O97M/Farheyt	0.9%
8	None	0.9%
9	O97M/Daoyap	0.9%
10	W97M/Bartallex	0.6%

Table 8 The rates of malware families in 2016's samples.

	family	rate
1	O97M/Donoff	65.4%
2	**New malware families**	14.6%
3	None	7.1%
4	O97M/Madeba	5.3%
5	W97M/Thus	3.4%
6	W97M/Marker	1.7%
7	O97M/Farheyt	1.3%
8	O97M/Macrobe	0.4%
9	W97M/Adnel	0.2%
10	O97M/Bartallex	0.2%

macros. **Table 6** shows the numbers of samples.

This dataset was collected and provided by Virus Total [18]. We selected all VBA macros whose file extensions were doc, docx, xls, xlsx, ppt, and pptx. These samples were uploaded to Virus Total between 2015 and 2016 for the first time. The malicious samples are judged malicious by a rate of more than 50% anti-virus vendors. The benign samples are judged benign by the whole anti-virus vendors. We investigated the rates in September 2018. This means that anti-virus vendors have plenty of time for analyzing. Therefore, we assume the rates are partially reliable. However, some malicious VBA macros for APT attacks might not be shared with all anti-virus vendors. Hence, we chose these thresholds for sample selection. There is no overlap in these specimens. We use 2015's samples as training data and 2016's samples as test data in the following experiments. In the following experiments, we assume that the present time is the end of 2015. In the end of 2015, 2016's samples were unseen samples. At that time, many anti-virus programs with the latest definition probably could not detect these samples. Because, anti-virus programs need samples to update the definition, and these samples were uploaded during 2016 for the first time. Subsequently, these samples had been analyzed and finally labeled.

Table 7 and **Table 8** illustrate the malware families and rates in the datasets. These malware families are defined by Microsoft Defender [19]. The representative malware family is `O97M/Donoff` in the both datasets. In comparison with 2015's samples, 2016's samples contain 14.6% of new malware families.

Table 9 Confusion Matrix.

		actual value	
		true	false
predicted result	positive	*TP*	*FP*
	false	*FN*	*TN*

6.3 Evaluation Measure

To evaluate accuracy, this paper uses Precision, Recall, and F-measure as metrics. These metrics are defined as follows.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - measure = \frac{2Recall \times Precision}{Recall + Precision}$$

Table 9 shows the confusion matrix.

F-measure is useful metrics and considers both the precision and recall. Since this paper does not investigate the details of detection rates deeply, this paper focuses on F-measure.

6.4 Experiment

To evaluate our method, we conduct the four following experiments. Each experiment corresponds to each research question described in Section 1.

Experiment 1

After extracting words from both benign and malicious macros, our method selects the important words. To select these words, our method has 3 options as follows.

- Select words from malicious macros
- Select words from benign macros
- Select words from both macros

Note that the selected words are extracted from both benign and malicious macros in any options. In this experiment, we attempt these 3 options with BoW and SVM. To adjust the number of dimensions, important words are selected with TF. The purpose of this experiment is investigating the most effective method to extract words and construct a corpus. In the following experiments, our method uses the best method to construct a corpus.

Experiment 2

Our method replaces some patterns into single words. To evaluate the effectiveness, our method has 2 options as follows.

- Replace some patterns into single words (Replaced)
- Do not replace (Unreplaced)

In this experiment, we attempt these methods with BoW and SVM. To adjust the number of dimensions, important words are selected with TF. The purpose of this experiment is evaluating the effectiveness of the replacement process.

Experiment 3

This experiment compares the combinations of the language models and classifiers. In this experiment, we attempt BoW and Doc2vec as the language models, SVM, RF, and MLP as the classifiers. To adjust the number of dimensions, important words are selected with TF. The purpose of this experiment is investigating the best combination of the language models and classifiers. In the following experiment,

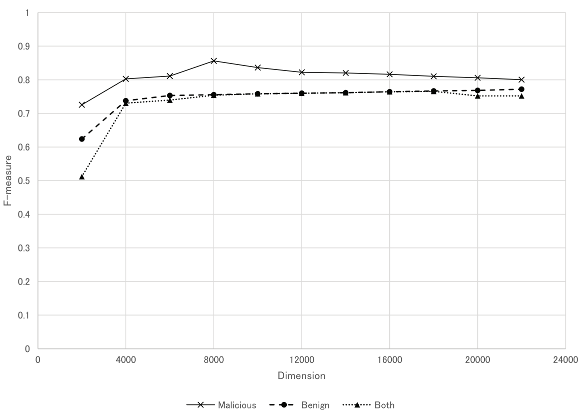

Fig. 2 The classification accuracy of the methods with BoW and SVM.

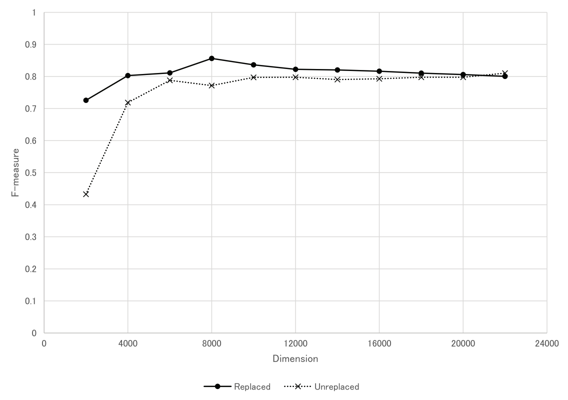

Fig. 3 The classification accuracy of the replaced method and unreplaced method with BoW and SVM.

our method uses the best combination.

Experiment 4

The final experiment compares the methods to select important words. TF and TFIDF are used to adjust the number of dimensions. The purpose of this experiment is investigating the most effective method to select important words.

6.5 Result

Experiment 1

Figure 2 shows the classification accuracy of the 3 methods with BoW and SVM. The horizontal axis corresponds to the dimensions, and the vertical axis corresponds to the F-measure.

As a result, extracting words from malicious macros is effective. Therefore, our method uses this method to construct a corpus in the following experiments.

Experiment 2

Figure 3 shows the classification accuracy of the replaced method and unreplaced method with BoW and SVM. The horizontal axis corresponds to the dimensions, and the vertical axis corresponds to the F-measure.

As a result, replacing these patterns into single words is effective. Therefore, our method replaces these patterns into single words in the following experiments.

Experiment 3

Figure 4, **Fig. 5**, and **Fig. 6** show the classification accuracy of the combinations of the language models and classifiers. The horizontal axis corresponds to the dimensions, and the

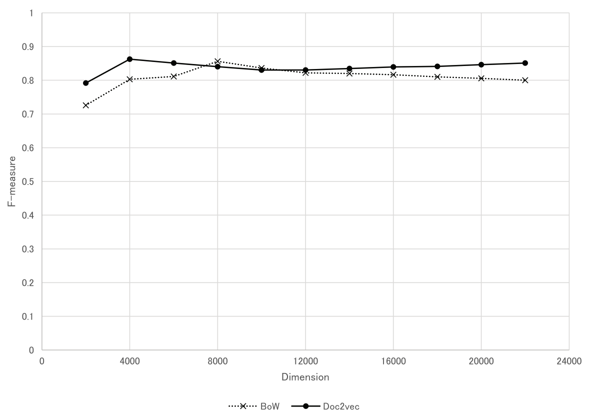

Fig. 4 The classification accuracy of the SVM with BoW and Doc2vec.

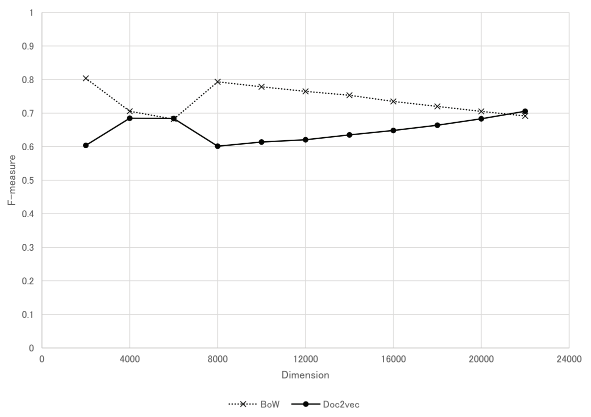

Fig. 5 The classification accuracy of the RF with BoW and Doc2vec.

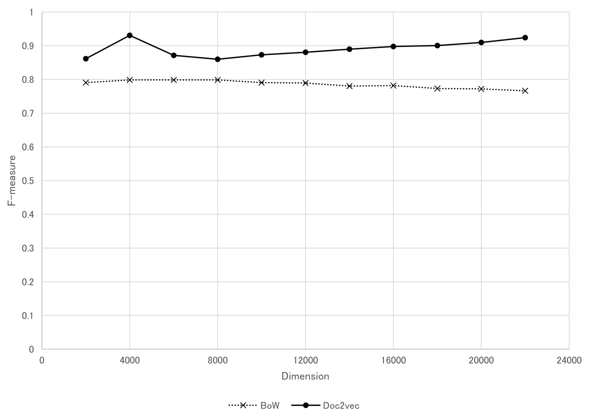

Fig. 6 The classification accuracy of the MLP with BoW and Doc2vec.

vertical axis corresponds to the F-measure.

Overall, the F-measure of Doc2vec is higher than BoW in Fig. 4 and Fig. 6. In contrast, the F-measure of BoW is higher than Doc2vec in Fig. 5. In Fig. 6, the best F-measure achieves 0.93. Moreover, the F-measure with MLP and Doc2vec is stable. The combination of SVM and Doc2vec is also stable and quite good. Therefore, we conclude the best combination is MLP and Doc2vec.

Experiment 4

Figure 7 shows the classification accuracy of the methods with TF and TFIDF. The horizontal axis corresponds to the dimensions, and the vertical axis corresponds to the F-measure.

As a result, TF is more effective than TFIDF. The best F-

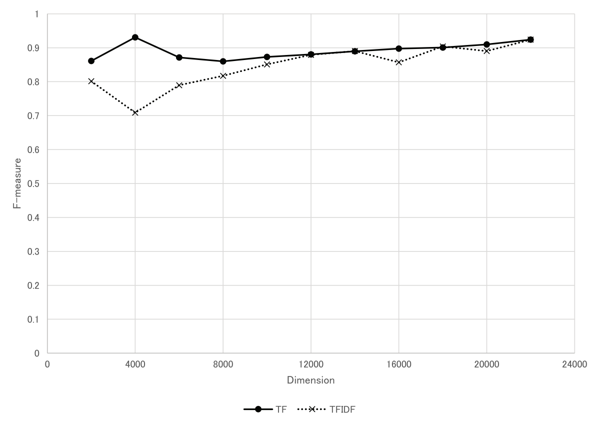

Fig. 7 The classification accuracy of the methods with TF and TFIDF.

measure achieves 0.93.

7. Discussion

7.1 Detecting Unseen Malware Families

We investigated new malware families in the 2016's samples. **Table 10** shows the families.

Our method detected 89% of the new malware families. Therefore, our method can detect unseen malicious macros which contain new malware families. The main functions of malicious macros are downloading and executing the main body. New malware samples have to contain these functions to some extent. Because, they require the main body which has many sophisticated functions. Therefore, our method can effectively detect new malware samples.

Our method could not detect some malicious macros. These malicious macros are not obfuscated. Hence, one possible reason is that these macros do not contain the typical patterns described in Section 3. These macros contain suspicious SQL commands and URLs. These suspicious words do not frequently occur in benign macros. If we replace these suspicious words into single words, the detection rate might be improved.

7.2 Frequent Words in Malicious Macros

In the first experiment, extracting words from malicious macros was the most effective. To reveal the reason, **Table 11** shows the frequent words in malicious macros. The ratio is calculated by dividing "the number of samples which contain the word" by "the number of samples".

The left half of the table contains frequent words in malicious macros. The right half corresponds to the word frequency in benign macros. Malicious macros include many of these frequent words. In contrast, benign macros rarely include these words except some words. Therefore, classifiers can easily discriminate between malicious and benign macros. This also explains why TFIDF was less effective than TF to classify macros. TFIDF values increase, if a word appears rarely in an entire corpus. These frequent words, however, frequently appear in malicious macros. Hence, these TFIDF values decrease. That is why TFIDF was less effective than TF. Furthermore, these frequent words contain some replaced words such as "elementofarray" or "longchr". This might be one possible reason that replacing some patterns into single words was effective.

Table 10 The new malware families in the 2016's samples.

	family		family
1	JS/Swabfex	15	W97M/Nsi
2	O97M/Zinunlate	16	X97M/ShellHide
3	O97M/Vibro	17	W97M/Qncwan
4	X97M/Mailcab	18	W97M/Groov
5	W97M/Avosim	19	W97M/Agent
6	X97M/Laroux	20	Win32/Bitrep
7	O97M/Powmet	21	W97M/Walker
8	XM/Laroux	22	W97M/Broxoff
9	O97M/Pyordonofz	23	Win32/Occamy
10	W97M/Xaler	24	Win32/Skeeyah
11	O97M/Emulasev	25	O97M/Prikormka
12	O97M/Bancarobe	26	W97M/Ursnif
13	Gen	27	Win32/Tiggre
14	O97M/DarkSnow	28	O97M/Pollwer

Table 11 Frequent words in malicious macros.

word	ratio in malicious	ratio in benign
elementofarray	99.0%	43.0%
andchr	93.9%	28.0%
next	90.9%	27.9%
function	85.1%	18.3%
string	83.3%	25.7%
len	79.0%	14.7%
public	77.5%	17.7%
longchr	73.7%	19.7%
createobject	73.0%	6.6%
error	73.0%	20.7%
byte	56.1%	1.5%
callbyname	51.3%	0.1%

7.3 The Best Combination of the NLP Techniques and Classifiers

In the third experiment, Doc2vec was more effective than BoW for classifying macros. This result might have depended on the word order or meaning. The F-measure with MLP and Doc2vec was stable, and the best F-measure achieved 0.93. The combination of SVM and Doc2vec was also stable and quite good. MLP and SVM have something in common. These classifiers perform quick pattern classification by linear separation. Therefore, we conclude that Doc2vec and linear classifiers are effective for classifying macros.

In the first and second experiments, we used BoW and SVM, which are the most basic and fundamental algorithms. We did not evaluate with RF, MLP, and Doc2vec. Hence, there is some possibility that these combinations might achieve better results. SVM, however, obtained the most stable results. Therefore, it appears that the rough results make little difference.

7.4 Comparison

The purpose of our method is detecting unseen malicious VBA macros. In practical use, many methods which contain our method can only use previous samples for training, and the test samples should not be the previous samples. If test samples contain previous samples, it is not possible to evaluate the performance appropriately. Therefore, appropriate experimental conditions with enough samples are required.

Bearden et al. proposed a method to detect malicious VBA macros with machine learning and NLP techniques [3]. Their method uses traditional machine learning and NLP techniques, and achieved 96.3% classification accuracy. However, the samples were only 40 malicious and 118 benign MS Office files. To evaluate their method in practical use, more samples are required.

Kim et al. proposed a method to detect malicious VBA macros with feature vectors from obfuscated source code [4]. Their method might not be able to detect non-obfuscated malicious VBA macros. Our method uses not only features in obfuscated macros, but also other features. Moreover, they conducted cross-validation with thousands of samples to evaluate their method. The details of the samples are not described in their paper. However, in reality, their method can only use previous samples for training. Hence, cross-validation is not appropriate in this case. Their method might not detect unseen malicious macros which contain new malware families. We evaluated our method with thousands of samples, and used only previous samples for training. We described details of our samples, and indicated that our method could detect unseen malicious macros which contain new malware families.

8. Conclusion

In this paper, we propose a method to detect unseen malicious macros themselves. To investigate the source code, we focus on NLP techniques. Our method divides the source code into words, and extracts feature vectors from the corpus with BoW and Doc2vec. Our method selects important words with TF and TFIDF to improve accuracy. Then, our method uses basic classifiers to detect unseen macros. Experimental results show that our method can detect 89% of new malware families, and the best F-measure achieves 0.93. Doc2vec represents feature vectors effectively, and the best combination of NLP techniques and classifiers is Doc2vec and MLP. Contrary to our expectations, TF is more effective than TFIDF in classifying macros.

In this paper, we used both malicious and benign samples obtained from Virus Total. We assumed these samples represented all VBA macros on the Internet. We selected the whole VBA macros whose file extensions were doc, docx, xls, xlsx, ppt, and pptx. Hence, we believe these malicious samples mostly represent the population of malware samples. More benign samples, however, might have to be collected to represent the population. For future work, we should evaluate our method with other samples. To derive more reliable results, samples should be obtained from other sources. More latest malware samples should be investigated. However, as we mentioned previously, these latest samples should be analyzed on a long-term basis. It seems to take more time to label precisely. Developing a practical detection system is another task for future work.

References

[1] Wolf in sheep's clothing: A SophosLabs investigation into delivering malware via VBA, available from ⟨https://nakedsecurity.sophos.com/2017/05/31/wolf-in-sheeps-clothing-a-sophoslabs-investigation-into-delivering-malware-via-vba/⟩.

[2] Nissim, N., Cohen, A. and Elovici, Y.: 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).

[3] Bearden, R. and Lo, D.C.-T.: Automated microsoft office macro malware detection using machine learning, *2017 IEEE International Conference on Big Data*, *BigData 2017*, pp.4448–4452, IEEE (2017) (online), available from ⟨http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8241556⟩.

[4] Kim, S., Hong, S., Oh, J. and Lee, H.: Obfuscated VBA Macro Detection Using Machine Learning, *DSN*, pp.490–501, IEEE Computer Society (2018) (online), available from ⟨http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8415926⟩.

[5] Miura, H., Mimura, M. and Tanaka, H.: Discovering New Malware Families Using a Linguistic-Based Macros Detection Method, *2018 Sixth International Symposium on Computing and Networking Workshops* (*CANDARW*), pp.431–437 (online), DOI: 10.1109/CANDARW.2018.00085 (2018).

[6] Miura, H., Mimura, M. and Tanaka, H.: Macros Finder: Do You Remember LOVELETTER?, *Proc. Information Security Practice and Experience - 14th International Conference*, *ISPEC 2018*, pp.3–18 (online), DOI: 10.1007/978-3-319-99807-7_1 (2018).

[7] Naser, A., Hjouj Btoush, M. and Hadi, A.: Analyzing and Detecting Malicious Content: DOCX Files, *International Journal of Computer Science and Information Security* (*IJCSIS*), Vol.14, pp.404–412 (2016).

[8] Otsubo, Y., Mimura, M. and Tanaka, H.: O-checker: Detection of Malicious Documents through Deviation from File Format Specifications, Black Hat USA (2016).

[9] Boldewin, F.: Analyzing MSOffice malware with OfficeMalScanner, *30th July* (2009).

[10] Mimura, M., Otsubo, Y. and Tanaka, H.: Evaluation of a Brute Forcing Tool that Extracts the RAT from a Malicious Document File, *AsiaJ-CIS*, IEEE Computer Society, pp.147–154 (2016) (online), available from ⟨http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7781470⟩.

[11] Corona, I., Maiorca, D., Ariu, D. and Giacinto, G.: Lux0R: Detection of Malicious PDF-embedded JavaScript code through Discriminant Analysis of API References, *Proc. 2014 Workshop on Artificial Intelligent and Security Workshop*, *AISec 2014* (2014), Dimitrakakis, C., Mitrokotsa, A., Rubinstein, B.I.P. and Ahn, G.-J. (Eds.), pp.47–57, ACM (2014) (online), available from ⟨http://dl.acm.org/citation.cfm?id=2666652⟩.

[12] Liu, D., Wang, H. and Stavrou, A.: Detecting Malicious Javascript in PDF through Document Instrumentation, *DSN*, pp.100–111, IEEE Computer Society (2014) (online), available from ⟨http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6900116; http://www.computer.org/csdl/proceedings/dsn/2014/2233/00/index.html⟩.

[13] Le, Q.V. and Mikolov, T.: Distributed Representations of Sentences and Documents, *Proc. 31th International Conference on Machine Learning*, *ICML 2014*, pp.1188–1196 (2014) (online), available from ⟨http://jmlr.org/proceedings/papers/v32/le14.html⟩.

[14] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J.: Distributed Representations of Words and Phrases and their Compositionality, *Advances in Neural Information Processing Systems 26: Proc. 27th Annual Conference on Neural Information Processing Systems 2013*, pp.3111–3119 (2013) (online), available from ⟨http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality⟩.

[15] olevba, available from ⟨https://github.com/decalage2/oletools/wiki/olevba⟩.

[16] gensim topic modelling for humans, available from ⟨https://radimrehurek.com/gensim/⟩.

[17] scikit-learn Machine Learning in Python, available from ⟨https://scikit-learn.org/⟩.

[18] Virus Total, available from ⟨https://www.virustotal.com/⟩.

[19] Windows Defender Antivirus, available from ⟨https://www.microsoft.com/en-us/windows/windows-defender/⟩.

Mamoru Mimura received his B.E. and M.E. in Engineering from National Defense Academy of Japan, in 2001 and 2008 respectively. He received his Ph.D. in Informatics from the Institute of Information Security in 2011 and M.B.A. from Hosei University in 2014. During 2001–2017, he was a member of the Japanese Maritime Self Defense Forces. During 2011–2013, he was with the National Information Security Center. Since 2014, he has been a researcher in the Institute of Information Security. Since 2015, he has been with the National center of Incident readiness and Strategy for Cybersecurity. Currently, he is an Associate Professor in the Department of C.S., National Defense Academy of Japan.

Hiroya Miura received his B.E. and M.E. in Engineering from National Defense Academy of Japan, in 2013 and 2019 respectively. Currently, he is a member of the Japanese Ground Self Defense Forces.