Electronic Preprint for Journal of Information Processing Vol.28

Recommended Paper

Using LSI to Detect Unknown Malicious VBA Macros

Mamoru Mimura[1,a) Taro Ohminami[2]

Received: November 25, 2019, Accepted: June 1, 2020

Abstract: Targeted email attacks are one of the main threats to organizations of all sizes and fields. In targeted email attacks, malicious VBA (Visual Basic for Applications) macros are often embedded into the attachment files to compromise the target computers. These malicious VBA macros are obfuscated in several ways to deceive anti-virus programs. Therefore there are limitations on applying pattern-based detection to detecting these unknown malicious VBA macros. To detect unknown malicious VBA macros, some methods with machine learning techniques are applicable. One method extracts words from the source code, and constructs a language model to represent VBA macros for machine learning techniques. This method constructs a language model from all the extracted words which include trivial words. Hence, there seems still room for improvement of this model. To construct an efficient language model, this paper focuses on LSI (Latent Semantic Indexing). LSI is a fundamental technique in topic modeling and calculates similarity of documents. Our method extracts words from the source code and converts them into feature vectors with several natural language processing techniques. Our method trains a classifier with benign and malicious VBA macros and detects unknown malicious VBA macros. Several thousands of samples for evaluation are obtained from Virus Total. The experimental results show that our method could detect unknown malicious VBA macros more efficiently, and reveal the advantages and disadvantages of each language model.

Keywords: VBA macro, SVM, NLP, Bag-of-Words, TFIDF, LSI

1. Introduction

Targeted email attacks are one of the main threats to organizations of all sizes in every field. In targeted email attacks, malicious VBA (Visual Basic for Applications) macros are often embedded into the attachment files to compromise the target computers. According to a report, MS (Microsoft) document files account for a large percentage of the attachments in targeted email attacks and most MS document files contain malicious VBA macros [*1]. VBA is the programming language of Office programs and allows automating tasks in MS document files. These malicious VBA macros are obfuscated in several ways to deceive anti-virus programs. Some attackers confirm in advance that their VBA macro will not be detected even by anti-virus programs having the latest definitions. In fact, even main anti-virus programs with the latest definitions cannot always detect new malware samples [1], [2]. Hence, pattern-based detection has limits when detecting these new malicious VBA macros. Moreover, malicious VBA macros do not require vulnerabilities to compromise a computer.

To detect new malicious VBA macros, some methods with machine learning techniques have been proposed [3], [4]. Due to limited evaluation methods, the practical performance and long-term effect are still open to discussion. One method extracts words from the source code and constructs a language model to represent VBA macros for machine learning techniques [5], [6], [7]. Thereafter, these represented feature vectors are classified by trained models such as SVM (Support Vector Machine). This method constructs a language model from all the extracted words, which include trivial words. For instance, common words included in most VBA macros are not useful for classification. Conversely, very rare words that seldom appear in VBA macros might be not useful. Hence, there seems still room for improvement on this model.

Our method assumes VBA macros are written by a natural language. To construct a more efficient language model, our method uses LSI (Latent Semantic Indexing). LSI is a natural language processing technique for extracting semantics of words in documents. This model assumes that words that are close in meaning will occur in similar pieces of documents, and reduces the number of words while preserving the similarity structure by SVD (Singular Value Decomposition). Our method extracts words from the source code and converts them into feature vectors with NLP (Natural Language Processing) techniques such as LSI. Our method trains a model with benign and malicious VBA macros, and detects unknown malicious VBA macros with the trained model. Our method requires benign and malicious samples for training. We obtained several thousands benign and malicious samples from Virus Total [*2], which is one of the most popular sites that share malware samples. To evaluate the practical performance and long-term effect against new malicious VBA

[1] National Defense Academy, Yokosuka, Kanagawa 239–8686, Japan
[2] Japan Air Self-Defense Force, Shinjuku, Tokyo 162–8801, Japan
[a)] mim@nda.ac.jp

The preliminary version of this paper was published at the 14th International Workshop on Security (IWSEC 2019), August 2019. The paper was recommended to be submitted to Journal of Information Processing (JIP) by the Program Co-Chairs of IWSEC 2019.

[*1] https://nakedsecurity.sophos.com/2017/05/31/wolf-in-sheeps-clothing-a-sophoslabs-investigation-into-delivering-malware-via-vba/

[*2] https://www.virustotal.com/

macros, the time series of samples is very important. The datasets for evaluation are constructed, considering the time series. The experimental results show that our method could detect unknown malicious VBA macros containing new malware families, and reveal the advantages and disadvantages compared with each language model.

Previous studies do not reveal the effectiveness of LSI in VBA macros nor evaluate the long-term effect of their methods. Even though the required time is an essential element in evaluating practical performance, these studies did not pay attention to required time either. Our previous study [8] did not evaluate the long-term effect by time series analysis with each language model. This study will reveal the advantages and disadvantages of each language model. This paper provides the following contributions:

(1) Proposes a more accurate and efficient method to detect unknown malicious VBA macros with LSI [8].
(2) Evaluates the long-term effect by time series analysis with each language model.
(3) Reveals the advantages and disadvantages of each language model.

The structure of this paper is shown below. Section 2 describes related studies. Section 3 describes malicious VBA macros, and Section 4 provides NLP techniques. Section 5 presents our method, and Section 6 demonstrates the performance. Finally, we discuss the results and conclude this paper.

2. Related Work

Our method examines MS document files by applying NLP techniques.

2.1 MS Document File

Several methods without dynamic analysis are proposed to examine MS document files. OfficeMalScanner is a basic tool to detect malicious MS document files [9]. This tool scans the entire MS document file for generic shellcode patterns or embedded objects, and the malicious index rating can be used for automated analysis as threshold values. We proposed a method to extract the executable files embedded in a document file [1]. This method can detect new malicious document files with known obfuscation methods. Otsubo et al. developed a tool to detect anomaly file structures of document files containing executable files [10]. They examined hundreds of malicious document files and found document files not strictly conforming to the file format.

A similar idea has been extended to XML-based office documents. Cohen et al. presented a novel structural feature extraction method for XML-based Office documents [2]. This method extracts discriminative features from malicious documents based on their structure, and detects malicious document files with machine learning algorithms. Nissim et al. created a detection model that detects new malicious docx files [11]. In this model, detection relies upon their structural feature extraction methodology [2].

Another approach is a visualization based method for malware detection [12]. In this approach, the target file is converted into an image to examine. Deep learning methods have achieved outstanding performance in image classification. Yakura et al. used CNN (Convolutional Neural Network) with Attention Mechanism to analyze imaged binary samples [13], [14]. CNN was also used to detect shellcode in imaged document files [15].

These methods can be applied to document files and detect new malicious document files. Some methods might detect malicious document files which contain VBA macros. These methods however do not examine the contents of VBA macros. Hence, malicious VBA macros in a normal document file might not be detected. Thus, there were few methods to detect malicious VBA macros themselves. One possible reason is that malicious VBA macros are relatively few in number compared to malicious document files.

2.2 VBA Macro

There are several methods for examining the contents of VBA macros. Bearden et al. proposed a method of classifying MS office files containing macros as malicious or benign using the K-Nearest Neighbors machine learning algorithm [3]. This method extracts important features from p-code opcode (translated VBA macro code) with TFIDF (Term Frequency-Inverse Document Frequency). Our method uses raw VBA macro code and constructs an LSI model from the TFIDF scores. Kim et al. focused on the obfuscation techniques and proposed an obfuscated macro code detection method using machine learning classifiers [4]. To train these classifiers, their method uses 15 discriminant static features, taking into account the characteristics. Because cross-validation is not valid for time series models, the practical performance and long-term effect are still open to discussion.

Miura et al. proposed a method to detect new malicious VBA macros with Doc2vec [5], [6], [7]. Doc2vec learns fixed-length feature representations from variable-length pieces of texts, such as documents [16]. This method extracts words from the source code, and constructs a Doc2vec model to represent VBA macros for classifiers. This method, however, constructs a language model from all the extracted words. Hence, this model might contain unnecessary words to classify.

2.3 NLP-based Detection

In the network security field, several studies focus on NLP techniques. A Doc2vec model was used to examine proxy logs for intrusion detection [17], [18]. This method was improved to mitigate the class imbalance problem [19], [20]. A similar technique was extended to examine network traffic [21], [22], or JavaScript code [23]. Ndichu et al. used Abstract Syntax Tree (AST) for code structure representation and Doc2vec for feature learning to detect malicious JavaScript code [24]. Ito et al. proposed a method to detect malicious executable files with NLP techniques [25]. Wang et al. proposed a source code-based analysis system that detects vulnerabilities with NLP techniques [26]. These methods use NLP techniques to examine proxy logs, network packets, JavaScript code and so on. Our method uses NLP techniques to examine the contents of VBA macros.

3. Malicious VBA Macro

3.1 Behavior

Malicious VBA macros mainly contained in MS document files conform to CFB (Compound File Binary) file format or OOXML (Office Open XML) file format. CFB file format is the binary file format used by Microsoft Office 2003 and earlier. OOXML stores a document as a collection of separate files and folders in a compressed zip package, and is used by Microsoft Word 2007 and later versions. Many extensions conforming to both file formats support VBA macros. VBA is a programming language running with Office programs and provides useful functions. VBA macros are a series of commands that can be run automatically to perform a task. The following sample code shows how to run a VBA macro automatically.

```
1: Sub Auto_Open()
2: Msgbox "Hello World!"
3: End Sub
```

This simple example shows a welcome message with specific text. Thus, attackers abuse these useful functions to compromise other computers. They code a malicious VBA macro into an MS document file, and send it to the target by e-mail. Malicious VBA macros mainly could be used to drop malware and download malware. The former is called dropper and the latter is called downloader. Once a malicious document file is opened, only a single click is required for the malicious VBA macro to activate.

Dropper contains and extracts the main body from itself. The main body is encoded or obfuscated by various methods. Thus, dropper enables persistent access to the computer without Internet connection. Downloader downloads the main body from the Internet as its name suggests. Thus, downloader requires internet connection to gain persistent access to the computer. Downloader does not contain the main body. Hence, the size tends to be less than dropper's.

As described in this section, malicious VBA macros tend to contain functions to download or extract the main body in the source code. Traditional approaches attempted to represent these features manually. In contrast, our method attempts to detect these features automatically with NLP techniques.

3.2 Obfuscation

Malicious VBA macros are often obfuscated by various methods such as Base64. The typical obfuscation techniques are summarized in **Table 1**.

The Encode Strings technique is converting parameters with reversible algorithms. Several functions are used for encoding strings. For instance, some functions such as Asc(), Hex(), and Chr() change characters to the number of the ASCII code. These functions convert strings into numerous numerical characters. Therefore, malicious VBA macros tend to contain these functions and formatted numerical characters.

Table 1 Typical obfuscation techniques.

	method	example
1	encode strings	convert to ASCII code
2	replace strings	replace with random strings
3	split strings	divide strings into characters

The Replace Strings technique is replacing strings with random strings. Several functions are used for replacing strings. For instance, some functions such as Replace(), Right(), or Left() are used to replace strings to other random strings. These functions mainly convert function names and variable names into random strings. Therefore, malicious VBA macros tend to contain these functions and random strings.

The Split Strings technique divides the strings into other characters. This technique is effective for hiding from signature-based anti-virus programs. The divided characters are restored to their original strings by join operators such as "and" or "plus".

As described in this section, malicious VBA macros tend to contain these functions and characteristic strings. Kim et al. extracted these features manually to use machine learning classifiers [4]. Our method does not extract these fixed features. Our method assumes that NLP techniques will automatically extract these features.

4. NLP Technique

4.1 Bag-of-Words

BoW (Bag-of-Words) is a basic method to extract feature vectors from documents. BoW represents the frequency of a word in a document and extracts a 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 a matrix. Therefore, the more the number of unique words increases, the greater the increase in the number of dimensions. 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

TFIDF (Term Frequency-Inverse Document Frequency) is one of the most popular methods for defining word importance. TFIDF value is calculated as follows.

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

The $frequency_{i,j}$ 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 TF is the $frequency_{i,j}$. 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 the 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 Latent Semantic Indexing

BoW converts documents into vectors. From these vectors, TFIDF produces vectors with the same number of rows and columns, which represent word importance. To reduce the dimension of the vectors, LSI (Latent Semantic Indexing) or LSA (Latent Semantic Analysis) is often used. The core idea is to take a matrix of documents and words, and decompose it into a separate document-topic matrix and a topic-word matrix. LSI is used for classifying documents, data clustering and calculating similarity of documents. Given m documents and n words in our vocabulary, we can construct an $m \times n$ matrix T in which each row represents a document and each column represents a word. T is described as follows.

$$T = \begin{bmatrix} tf_{11} & \cdots & tf_{1n} \\ \vdots & \ddots & \vdots \\ tf_{m1} & \cdots & tf_{mn} \end{bmatrix}$$

In this matrix, a row will be a vector corresponding to a word giving its relation to each document. Likewise, a column in this matrix will be a vector corresponding to a document giving its relation to each word. In the simplest version of LSI, each entry can simply be a raw count of the number of times the j-th word appeared in the i-th document. In practice, however, raw counts do not work particularly well because they do not account for the significance of each word in the document. In all likelihood, T is sparse and redundant across its many dimensions. As a result, to find the few latent topics that capture the relationships among the words and documents, we want to perform dimensionality reduction on T. Dimensionality reduction can be performed using truncated SVD (Singular Value Decomposition). From the theory of linear algebra, there exists a decomposition of T such that D and W are orthogonal matrices and V is a diagonal matrix. Matrix T' is calculated from T by SVD using TFIDF as follows.

$$T' = DVW \approx \begin{bmatrix} d_1 \\ \vdots \\ d_r \end{bmatrix} \begin{bmatrix} v_1 & & 0 \\ & \ddots & \\ 0 & & v_r \end{bmatrix} \begin{bmatrix} w_1 & \cdots & w_r \end{bmatrix}$$

An element of V corresponds to each topic of all the documents. A row of D corresponds to each document. A column of W corresponds to each word. Thus, SVD reduces dimensionality by selecting only the r largest singular values, and only retaining the first r columns or rows of D and W. To select the largest values, we can use TFIDF scores. In this case, r is a hyper parameter we can select and adjust to reflect the number of topics. In our method, this parameter is used to reduce dimensionality while keeping the significance of each word in the document. This model efficiently represents word frequency of a document. However, this model does not consider the work meaning or context.

4.4 Doc2vec

Word2vec was created to represent word meaning or context [27]. Word2vec consists of shallow neural networks which are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of documents and produces a vector space of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words which share common contexts in the corpus are located in close proximity to one another in the space. Word2vec is a method to represent the word with a meaning or context. Paragraph Vector is the extension of Word2vec to represent a document [28]. Doc2vec is an implementation of the Paragraph Vector. The only change is replacing a word into a document ID. This model represents a document with word meaning or context. Words, however, could have different meanings in different contexts. Hence, vectors of two documents which contain the same word in two distinct senses need to account for this distinction.

5. Proposed Method

5.1 Outline

This paper proposes a method to detect unknown malicious VBA macros with LSI. Our method requires benign and malicious VBA macros as training data. The outline of our method is shown in **Fig. 1**.

In the training phase, our method extracts words from their source code, and constructs an LSI model. Thereafter, our method extracts feature vectors, and trains a classifier with the feature vectors and labels. In the test phase, our method extracts feature vectors from unknown samples with the LSI model, and detects malicious VBA macros with the trained classifier.

5.2 Training Phase

In the training phase, our method requires benign and malicious VBA macros as training data. These samples are obtained from web pages such as Virus Total. First, our method extracts VBA macros from benign and malicious MS document files. Thereafter, our method divides their source code into words (①). Special characters shown in **Table 2** are used as the delimiter.

Second, our method constructs an LSI model from these words extracted from all the samples (②). The unique words are selected from these words to construct a BoW model. The BoW model is converted into a TFIDF model which represents word importance. The TFIDF model produces the vectors, and these vectors are reduced by SVD to construct an LSI model. The LSI

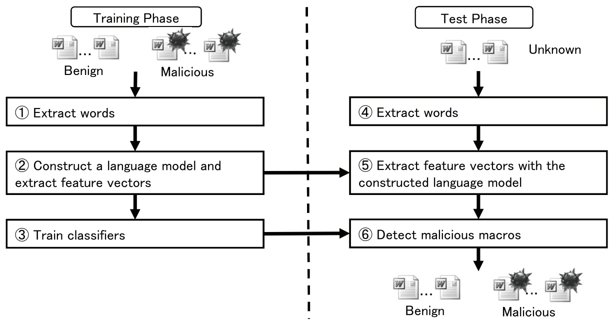
Training Phase
Benign
Malicious
① Extract words
② Construct a language model and extract feature vectors
③ Train classifiers
Test Phase
Unknown
④ Extract words
⑤ Extract feature vectors with the constructed language model
⑥ Detect malicious macros
Benign
Malicious

Fig. 1 Proposed method.

Table 2 Special characters as the delimiter.

symbol	pronounce	symbol	pronounce
”	double quote	\r	carriage return
‘	backquote	\f	form feed
<>	less / greater than	\v	vertical tab.
{}	brace	\t	horizontal tab.
()	parenthesis	\n	line feed
.	period	_	underline
,	comma	%	percent sign
:	colon	$	dollar sign
;	semicolon	/	slash
=	equals	!	exclamation mark
+	plus	?	question mark
-	dash	@	at sign
*	asterisk		

model converts benign and malicious VBA macros into feature vectors. The dimension of feature vectors is compressed into the number of topics. Finally, an SVM classifier is trained by these feature vectors with their labels (③). SVM is a supervised learning model that assigns new examples to one category or the other.

5.3 Test Phase

In the test phase, our method investigates unknown samples. These unknown samples are assumed as the attachments in targeted email attacks. First, our method extracts words from MS document files in the same way (④). Second, these words are converted into feature vectors by the LSI model (⑤) which was constructed in the training phase. Finally, the trained classifier investigates these feature vectors, and predicts the label (⑥).

6. Evaluation

6.1 Implementation

We implemented our method with Python2.7 in an environment as shown in **Table 3**. Our method uses olevba [*3] to extract VBA macros from MS document files. Olevba is a script to parse OLE and OpenXML files such as MS document files to detect VBA macros and extract their source code in clear text. Our method uses scikit-learn-0.19.2 [*4] to implement SVM. Scikit learn is a machine learning library and has many classification algorithms. The parameters are provided from a grid search which exhaustively generates candidates from a grid of parameter values. As a result, a linear kernel (C = 0.5) is chosen. We used gensim-3.4.0 [*5] to implement an LSI model. Gensim has many functions related to NLP techniques such as BoW or LSI. Moreover, we implemented the previous methods [5], [6], [7] with BoW and Doc2vec. In this paper, the same parameters are chosen to provide a fair comparison.

6.2 Dataset

To evaluate our method, actual VBA macros were obtained from Virus Total. These VBA macros contain both benign and malicious VBA macros. **Table 4** shows the number of samples.

We selected all MS document files containing VBA macros. Their file extensions are doc, docx, xls, xlsx, ppt, and pptx. These samples were uploaded to Virus Total between April 2015 and March 2018 for the first time. Each year in the table corresponds to the fiscal year from April to March. In general, unknown samples are investigated as soon as possible. Hence, we assume these samples appeared at the time. In targeted email attacks, anti-virus programs with the latest definitions cannot always detect new malware samples [1], [2]. This suggests that some anti-virus programs could not detect malicious samples correctly. Hence, we determined to use malicious samples, which are judged malicious by a rate of more than 50% of anti-virus vendors. The benign samples are judged benign by all anti-virus vendors. We compared the hash values and removed duplicated samples.

Table 5 shows the main malware families in each dataset.

Table 3 Environment.

CPU	IntelCorei7 (3.40 GHz)
Memory	16 GB
OS	Windows 10 Home

Table 4 The number of VBA macros obtained from Virus Total.

year	benign	malicious	total
2015	622	870	1,492
2016	1,200	1,150	2,350
2017	2,220	1,083	3,303
total	4,042	3,103	7,145

Table 5 Main malware families in each dataset.

Dataset	Family name
2015	TrojanDownloader:O97M/Donoff
	TrojanDownloader:O97M/Adnel
	TrojanDownloader:O97M/Bartallex
	TrojanDownloader:W97M/Adnel
	TrojanDownloader:W97M/Donoff
2016	TrojanDownloader:O97M/Donoff
	Virus:W97M/Thus.GB
	Trojan:Win32/Tiggre!rfn
	Virus:X97M/Metcol.A
	Trojan:Win32/Occamy.C
2017	TrojanDownloader:O97M/Donoff
	Trojan:O97M/Madeba.A!det
	TrojanDownloader:JS/Swabfex.P
	Virus:W97M/Thus.GB
	TrojanDownloader:O97M/Donoff.CD

Table 6 Confusion matrix.

		Actual value	
		True	False
Predicted result	Positive	*TP*	*FP*
	False	*FN*	*TN*

These names are defined by Windows Defender Antivirus [*6]. Thus, each dataset is adequately distributed and contains a wide variety of malware samples. Furthermore, the consecutive datasets contain new malware families, which are not defined yet.

6.3 Evaluation Metrics

To evaluate accuracy, we use Accuracy, Precision, Recall, and F1 score as metrics. These metrics are defined as follows.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

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

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

$$F1 = \frac{2Recall \times Precision}{Recall + Precision}$$

Table 6 shows the confusion matrix.

In this experiment, TP means detecting malicious VBA macros correctly.

6.4 Experimental Method

To reveal the performance and long-term effect against new VBA macros, the time series of samples is very important. The purpose of our method is detecting unknown 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 con-

[*3] https://github.com/decalage2/oletools/wiki/olevba
[*4] https://scikit-learn.org/
[*5] https://radimrehurek.com/gensim/
[*6] https://www.microsoft.com/en-us/windows/windows-defender/

Table 7 The combinations of the training data and test data.

	1	2	3
Training Data	2015	2015	2016
Test Data	2016	2017	2017

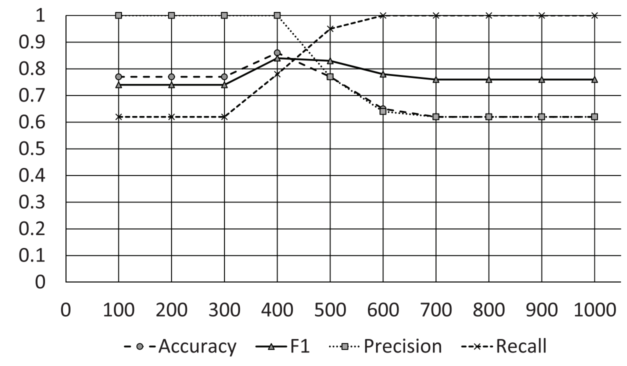

Fig. 2 Results of the preliminary experiments.

tain previous samples, it is not possible to evaluate the practical performance appropriately. Hence, cross-validation that randomly splits a dataset into a training and a testing set is not appropriate in this case [29]. Therefore, the datasets for evaluation are constructed considering the time series.

First, we conduct a preliminary experiment to find the optimum value of a hyper parameter. As described in the previous section, this hyper parameter is the number of topics, and is used to reduce dimensionality while retaining the significance of each word in the document. In this preliminary experiment, we conduct a 5-fold cross-validation with the 2015's dataset. We trace changes to the value of the dimension. Since this preliminary experiment, subsequent experiments will be conducted with the optimum value.

Next, we conduct 5-fold cross-validation with 2015's datasets to confirm the generalization performance.

Finally, we conduct time series analysis to reveal the performance and long-term effect. The combinations of the training data and test data are shown in **Table 7**.

The first combination is performed to reveal the stable performance. Our previous study did not consider the training bias of the language model [8]. In this experiment, we repeat the process 5 times to remove the training bias. This provides more reliable results. The second combination is performed to reveal the long-term performance. The third combination will reveal how update improves the performance.

6.5 Result

Figure 2 shows the result of the preliminary experiments.

The horizontal axis corresponds to the dimensions, and the vertical axis corresponds to each metric. The dimensions are adjusted by varying the parameter described in the previous section. As described in Fig. 2, the precision and recall are in a tradeoff relationship. The accuracy and F1 score are maximized at 400 dimensions. Thus, we determined the optimum value, and fix the number of topics to 400 in subsequent experiments.

Table 8 shows generalization performance of the 5-fold cross-validation with 2015's datasets.

LSI produced slightly better performance than other language

Table 8 Result of the 5-fold cross-validation.

metrics	BoW	LSI	Doc2vec
Accuracy	0.94	0.99	0.98
Precision	0.98	0.99	0.98
Recall	0.91	0.99	0.99
F1	0.95	0.99	0.98

Table 9 Average scores of the time series analysis.

metrics	BoW	LSI	Doc2vec
Accuracy	0.73	0.89	0.93
Precision	0.94	0.99	0.97
Recall	0.50	0.80	0.89
F1	0.66	0.88	0.93

Table 10 Average required time of the time series analysis.

	BoW	LSI	Doc2vec
construction	-	3.8 s	21.3 s
training	1.1 s	0.2 s	0.1 s
detection	0.1 s	0.1 s	0.1 s

Table 11 Average scores of the time series analysis (2015–2017).

metrics	BoW	LSI	Doc2vec
Accuracy	0.68	0.89	0.93
Precision	0.60	0.92	0.95
Recall	0.16	0.73	0.82
F1	0.23	0.82	0.88

Table 12 Average scores of the time series analysis (2016–2017).

metrics	BoW	LSI	Doc2vec
Accuracy	0.91	0.97	0.96
Precision	0.85	0.97	0.92
Recall	0.88	0.93	0.95
F1	0.87	0.95	0.94

models. All metrics achieved almost perfect performance. As we expected, BoW is less than other language models even in cross-validation. Because we reduced the dimension by the frequency for the sake of fair comparison. In a BoW model, the dimension indicates the word appearance, which is the critical and only element for detection. Therefore, we conclude that the generalization performance of our method is better.

Table 9 shows performance of the time series analysis.

Due to practical restriction, the performance is generally reduced from the 5-fold cross-validation. Regarding accuracy and F1 score, Doc2vec produced the best performance among the language models. The best F1 score achieves 0.93. LSI provides the second best performance.

Table 10 shows required time of the time series analysis.

The construction indicates time for the language model construction. The training and detection indicate time for the SVM classifier. Doc2vec requires more time than other language models for training which includes construction time. All language models could inspect 2,350 samples within 0.1 second.

Table 11 and **Table 12** show performance of the time series analysis with each combination.

Even though the training model was constructed over a year ago, the LSI and Doc2vec models maintain satisfactory performance. The best F1 score achieves 0.88 in the Doc2vec model. In contrast, the BoW model could no longer classify VBA macros.

Since the update, all models improved their performance significantly. The best F1 score achieves 0.95 in the LSI model. Thus, time aging slightly reduces the performance in the LSI and

Table 13 Each detection rate of known and unknown malware families.

Dataset	2016		2017	
Total number	1150		1083	
Each cline2-5 Number	Known 850	Unknown 300	Known 542	Unknown 541
Detection Count	779	272	466	446
Detection Rate (%)	91.6	90.7	82.3	86.1

Table 14 Number of unique words in the samples detected incorrectly.

Sample	Number of unique words included in only benign	Number of unique words included in only malicious
FN	4118	3195
FP	2384	1955

Doc2vec models. This can be improved by updating the training models. In the BoW model, time aging dramatically reduces the performance.

7. Discussion

7.1 Accuracy

In actual use, many methods using machine learning techniques cannot use the following samples for training. These methods should be evaluated with the previous samples. In the time series analysis, our method used only previous samples for training. The best F1 score achieves almost 0.95. Thus, our method is effective on new VBA macros. Despite the training samples that were discovered over a year ago, the F1 score maintains almost 0.82. Hence, our method is accurate, and performance is not excessively reduced due to aging.

Next, **Table 13** shows the detection rate of known and unknown malware families.

All of the detection rates are in almost the same range. Therefore, our method is effective not only for known families but also unknown malware families.

Several samples were detected incorrectly by our method. We analyzed these samples, and counted the unique words included in only benign and malicious files. **Table 14** shows the number of unique words in the samples detected incorrectly. The overlooked malicious samples (FN) tend to contain more words included in only benign files than other samples. Therefore, these malicious samples were classified as benign. The detected benign samples (FP) also tend to contain words included in only benign files. However, the words included in only malicious files are relatively numerous. Hence, we conclude that is why these benign samples were classified as malicious. Excluding these unique words from the language model might decrease false negatives or false positives. To improve the accuracy, words which do not decrease accuracy should be selected from these words.

7.2 Topic Vector

We analyzed the contents of the topic vectors to reveal the effectiveness of LSI. Some examples of the words classified by LSI are shown in **Table 15**.

The topics 1, 2, 3, and 6 consist of random strings which are often contained in obfuscated malicious VBA macros. Each topic has regularity such as alphabetical character, alphanumeric character, numerical character or the length. The topics 4 and 5 contain meaningful words. Some words are related with VBA macros. Thus, LSI classifies related words into each topic vector. This allows reducing the dimensions without lowering classification accuracy. Furthermore, the topic vectors represent the essence. Hence, LSI is effective for classifying VBA macros. Thus, LSI represents VBA macros efficiently, and that is why the LSI model is effective for detection.

Table 15 Some examples of the words classified by the topic vector.

Topic	Parts of the contents
1	epcrazkwlscpbqm, ipathwdinj, ylabbu, ynvneqrhdqazxz vxqhus, ubepnlsziosn
2	wlijflsdkj, ghrj32, 2k3h, sdlkjfwhfe, rkj23, njqkwndjwqd
3	qs8juqb1am, qa94, zyprern, e5iqj, twkk, zytologischen
4	control, checkbox, range, click, cells, sheets
5	range, selection, select, activewindow, long, macro
6	lirdifqhvefomgkmsysaqvqrpufmtkkkzqskk ujphtxxljdvsoljplwzpklvf fcyityvgxyuvedjncjpiqgmhiglvowacmdhjbadhsocwne smgytipxywbmgnqedtxarqgyiqsnquisnbobbxwzgy

7.3 Advantages and disadvantages

According to the experimental results, LSI and Doc2vec produced better performance than BoW. Doc2vec maintained better performance than LSI in the long term. Since the update, LSI improved the performance significantly. The best F1 score achieves 0.95 in the LSI model. This suggests that LSI requires many training samples to maintain good performance. In contrast, Doc2vec requires fewer training samples. However, Doc2vec requires more time than other language models for training. In actual use, there are no problems with a longer training time. Hence, the best language model depends on the situation. If we have enough training samples and can update the model frequently, we should use LSI. Doc2vec should be the alternative model to compensate for fewer samples.

7.4 Comparison

Based on the experimental result, this section produces the quantitative comparison. We focus on the evaluation methods of the previous studies to detect malicious VBA macros.

Bearden et al. conducted 10-fold cross-validation with several dozen samples to evaluate their method [3]. The number of malicious VBA macros is only 40. Kim et al. conducted 10-fold cross-validation with several thousand samples to evaluate their method [4]. Nevertheless, they used 90% of the samples for training, and the detection rate was no more than 0.915. Moreover, they did not describe the details of malware samples. As we described before, cross-validation is not appropriate for evaluating performance for detecting new malicious VBA macros because the training samples only contained the previous samples. Therefore, it is not clear whether their method can detect new malicious VBA macros in actual use. We used only previous samples for training from more than ten thousand samples. The training samples account for almost 25 percent of our samples. As a result, our method could detect new malicious VBA macros. Furthermore, this paper describes the details of malware samples.

Miura et al. evaluated their method with several thousand samples for 2 years [5], [6], [7]. The best F1 score achieved 0.93. However, they did not evaluate the long-term effect of their methods. We obtained six thousands of samples for 3 years from Virus Total. We categorized these samples into each year, and evaluated the long-term effect by time series analysis. Moreover, the best F1 score achieved almost 0.95. Previous studies also did not evaluate the time required by their methods. Required time is an essential element for evaluating actual performance. Our method requires only one-third to one-quarter of the time of the previous method. Furthermore, we compared each detection rate of known and unknown malware families. This revealed that our method was effective not only in detecting known families but also unknown malware families.

7.5 Ethics

Our method requires benign and malicious samples which can be collected from the Internet. Our method is light-weight and easy to implement. We used malware samples obtained from a commercial web site. We indicated clear conditions for selecting malware samples from the web site. The details of samples were described and investigated. Hence, our method is reproducible and has high transparency.

7.6 Limitations

In this paper, our method used an SVM classifier to detect malicious VBA macros. Other classifiers could be used for classification. This paper focused on the language models, and revealed their advantages and disadvantages. To evaluate the feature of the language models, the same classifier should have been used. SVM provided better accuracy for VBA macro detection [7]. Hence, we decided to use an SVM classifier.

Another limitation is caused by the datasets. In this experiment, we used balanced datasets extracted from macro samples. Ideally, the datasets should represent the population of all macros in the world. The actual ratio of benign and malicious samples seems highly imbalanced. As we mentioned previously, our datasets are more practical than previous studies. To evaluate the actual performance, more practical datasets are required. However, this issue is complicated and beyond the scope of this paper.

8. Conclusion

In this paper, we propose a more accurate and efficient method for detecting new malicious VBA macros with LSI. Our method extracts words from the source code and converts them into feature vectors with an LSI model. Previous works do not reveal the effectiveness of LSI in VBA macros nor evaluate the long-term effect of their methods. The experimental results reveal the effectiveness of LSI in VBA macros. Furthermore, we evaluate the long-term effect by time series analysis with actual VBA macros. The best F1 score achieves almost 0.95. Hence, our method is more accurate and can detect new malware families.

Our method is light-weight and investigates VBA macros without requiring much time. Therefore, one topic for future work is implementing a real-time detection system. We can implement our method on a mail server or proxy server for investigating files in real time. Another future work is evaluating the actual performance on more practical datasets.

References

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

[2] Cohen, A., Nissim, N., Rokach, L. and Elovici, Y.: SFEM: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods, *Expert Syst. Appl.*, Vol.63, pp.324–343 (2016).

[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*, Nie, J.-Y., Obradovic, Z., Suzumura, T., Ghosh, R., Nambiar, R., Wang, C., Zang, H., Baeza-Yates, R.A., Hu, X., Kepner, J., Cuzzocrea, A., Tang, J. and Toyoda, M. (Eds.), 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.: 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).

[6] 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).

[7] Mimura, M. and Miura, H.: Detecting Unseen Malicious VBA Macros with NLP Techniques, *JIP*, Vol.27, pp.555–563 (online), DOI: 10.2197/ipsjjip.27.555 (2019).

[8] Mimura, M. and Ohminami, T.: Towards Efficient Detection of Malicious VBA Macros with LSI, *Proc. Advances in Information and Computer Security - 14th International Workshop on Security, IWSEC 2019*, Attrapadung, N. and Yagi, T. (Eds.), Lecture Notes in Computer Science, Vol.11689, pp.168–185, Springer (2019).

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

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

[11] 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).

[12] Kancherla, K. and Mukkamala, S.: Image visualization based malware detection, *2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)*, pp.40–44 (online), DOI: 10.1109/CICYBS.2013.6597204 (2013).

[13] Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y. and Sakuma, J.: Malware Analysis of Imaged Binary Samples by Convolutional Neural Network with Attention Mechanism, *Proc. 8th ACM Conference on Data and Application Security and Privacy, CODASPY 2018*, pp.127–134 (online), DOI: 10.1145/3176258.3176335 (2018).

[14] Yakura, H., Shinozaki, S., Nishimura, R., Oyama, Y. and Sakuma, J.: Neural malware analysis with attention mechanism, *Computers & Security*, Vol.87, p.101592 (online), DOI: 10.1016/j.cose.2019.101592 (2019).

[15] Mimura, M., Otsubo, Y., Tanaka, H. and Goto, A.: Is Emulating "Binary Grep in Eyes" Possible with Machine Learning?, *CANDAR*, pp.337–343, IEEE Computer Society (2017) (online), available from ⟨http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8338657⟩.

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

[17] Mimura, M. and Tanaka, H.: Heavy Log Reader: Learning the Context of Cyber Attacks Automatically with Paragraph Vector, *Proc. Information Systems Security - 13th International Conference, ICISS 2017*, pp.146–163 (online), DOI: 10.1007/978-3-319-72598-7_9 (2017).

[18] Mimura, M. and Tanaka, H.: Leaving All Proxy Server Logs to Paragraph Vector, *Journal of Information Processing*, Vol.26, pp.804–812 (online), DOI: 10.2197/ipsjjip.26.804 (2018).

[19] Mimura, M. and Tanaka, H.: A Linguistic Approach Towards Intrusion Detection in Actual Proxy Logs, *Proc. 20th International Conference, ICICS 2018*, pp.708–718 (online), DOI: 10.1007/978-3-030-

01950-1_42 (2018).

[20] Mimura, M.: Adjusting lexical features of actual proxy logs for intrusion detection, *Journal of Information Security and Applications*, Vol.50, p.102408 (online), DOI: 10.1016/j.jisa.2019.102408 (2020).

[21] Mimura, M. and Tanaka, H.: Reading Network Packets as a Natural Language for Intrusion Detection, *Information Security and Cryptology - ICISC 2017 - 20th International Conference, Seoul, South Korea, November 29 - December 1, 2017, Revised Selected Papers*, pp.339–350 (online), DOI: 10.1007/978-3-319-78556-1_19 (2017).

[22] Mimura, M.: An Attempt to Read Network Traffic with Doc2vec, *Journal of Information Processing*, Vol.27, pp.711–719 (online), DOI: 10.2197/ipsjjip.27.711 (2019).

[23] Mimura, M. and Suga, Y.: Filtering Malicious JavaScript Code with Doc2Vec on an Imbalanced Dataset, *2019 14th Asia Joint Conference on Information Security* (*AsiaJCIS*), pp.24–31 (online), DOI: 10.1109/AsiaJCIS.2019.000-9 (2019).

[24] Ndichu, S., Kim, S., Ozawa, S., Misu, T. and Makishima, K.: A machine learning approach to detection of JavaScript-based attacks using AST features and paragraph vectors, *Applied Soft Computing*, Vol.84, p.105721 (online), DOI: 10.1016/j.asoc.2019.105721 (2019).

[25] Ito, R. and Mimura, M.: Detecting Unknown Malware from ASCII Strings with Natural Language Processing Techniques, *2019 14th Asia Joint Conference on Information Security* (*AsiaJCIS*), pp.1–8 (online), DOI: 10.1109/AsiaJCIS.2019.00-12 (2019).

[26] Wang, J., Ma, S., Zhang, Y., Li, J., Ma, Z., Mai, L., Chen, T. and Gu, D.: NLP-EYE: Detecting Memory Corruptions via Semantic-Aware Memory Operation Function Identification, *22nd International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2019*, pp.309–321 (2019) (online), available from ⟨https://www.usenix.org/conference/raid2019/presentation/wang-0⟩.

[27] Mikolov, T., Chen, K., Corrado, G. and Dean, J.: Efficient Estimation of Word Representations in Vector Space, *CoRR*, Vol.abs/1301.3781, pp.1–12 (2013) (online), available from ⟨http://arxiv.org/abs/1301.3781⟩.

[28] Le, Q. and Mikolov, T.: Distributed representations of sentences and documents, *International Conference on Machine Learning*, pp.1188–1196 (2014).

[29] Marchal, S. and Asokan, N.: On Designing and Evaluating Phishing Webpage Detection Techniques for the Real World, *11th USENIX Workshop on Cyber Security Experimentation and Test, CSET 2018*, Collberg, C.S. and Peterson, P.A.H. (Eds.), USENIX Association (2018).

Editor's Recommendation

This paper achieves to detect unknown malicious VBA macros by their machine learning scheme using LSI (Latent Semantic Indexing). This is the first attempt to detect unknown malicious VBA macros with LSI. In addition, several thousands of samples for evaluation are gathered. The evaluation method will be useful for other researches. The paper gives insights to readers in this research field and thus is selected as a recommended paper.

(Program Co-Chairs of IWSEC 2019, Takeshi Yagi)

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 Japan Maritime Self Defense Force. 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 Computer Science, National Defense Academy of Japan.

Taro Ohminami received his B.E. in Engineering from National Defense Academy of Japan in 2019. Currently, he is a member of the Japan Air Self-Defense Force.