

Received October 27, 2020, accepted November 7, 2020, date of publication November 11, 2020, date of current version November 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3037330

An Improved Method of Detecting Macro Malware on an Imbalanced Dataset

MAMORU MIMURA 

National Defense Academy, Kanagawa 239-8686, Japan


e-mail: mim@nda.ac.jp

ABSTRACT In spear-phishing attacks, macro malware written in VBA (Visual Basic for Applications) is often used to compromise the target computers. Macro malware is often obfuscated in several ways to evade detection. To detect new macro malware, several methods with machine learning techniques have been proposed. While many methods were evaluated with the inadequate or balanced dataset with the same number of benign and malicious samples, practical performance is still open to discussion. In reality, the population of VBA macros consists of wide variety of samples. To evaluate practical performance, an imbalanced dataset which contains many benign samples is required. In this paper, we propose an improved method of detecting macro malware on an imbalanced dataset. Our method uses 2 language models (Doc2vec and Latent Semantic Indexing (LSI)) and 4 popular classifiers. These language models are used to extract features and mitigate the class imbalance problem by selecting important features. We create an imbalanced dataset with more than 30,000 samples and evaluate the practical performance. The experimental result demonstrates that our method mitigates the class imbalance problem and could detect completely new malware regardless of the family type. The result also reveals that LSI is more robust than Doc2vec to the class imbalance problem.

INDEX TERMS Information security, intrusion detection, machine learning, VBA macros, office documents, macro malware, natural language processing, Doc2vec, latent semantic indexing.

I. INTRODUCTION

Spear-phishing attacks are one of main threats for organizations of all sizes and across every field. The use of malicious documents has increased rapidly along with a spectrum of attacks. They offer flexibility in document structure with numerous features for attackers to exploit [1]. While many studies focus on Portable Document Format (PDF) document files [2]–[8] or their JavaScript [9]–[11], this study focuses on Microsoft (MS) document files. In spear-phishing attacks, macro malware written in VBA (Visual Basic for Applications) is often used to compromise the target computers. While VBA enables to automate tasks in MS document files, attackers abuse its useful functions to execute malicious tasks. Malicious VBA macros are obfuscated to evade anti-virus programs with the latest definitions. In fact, main anti-virus programs barely detect new malware samples [12], [13]. In addition, there are few suitable code analysis tools for detecting macro malware. Thus, traditional pattern-based detection methods have a serious limitation in detecting new malware.

The associate editor coordinating the review of this manuscript and approving it for publication was Pedro R. M. Inácio .

To detect new macro malware, several methods with machine learning models have been proposed [14]–[19]. While many methods were evaluated with the inadequate or balanced dataset, practical performance is still open to discussion. For instance, some studies conducted cross validation with restricted samples [14], [17]. While cross validation reveals the generalization performance with limited samples, it does not indicate that their method is effective against new malware. Because the training samples may contain many similar samples. New malware could be completely new and different. To avoid this problem, the other studies consider the time series and confirmed that the training samples consist of earlier samples [18]–[23]. However, these methods are evaluated with a balanced dataset with the same number of benign and malicious samples. Thus, many previous studies do not consider the class imbalance problem and may not be robust on an imbalanced dataset. In addition, they did not reveal detection rates in malware families. In reality, the population of VBA macros consists of wide variety of samples and contains many benign samples. To evaluate practical performance, an imbalanced dataset which contains many benign samples is required [24].

In this paper, we propose an improved method of detecting macro malware on an imbalanced dataset. Our method uses 2 language models (Doc2vec and Latent Semantic Indexing (LSI)) and 4 popular classifiers. These language models are used to extract features. To improve the performance and mitigate the class imbalance problem, we apply a mitigation technique for proxy logs to macro malware [25]. This technique extracts important features based on word frequency. We create an imbalanced dataset with more than 30,000 samples and evaluate the practical performance. These samples are obtained from multiple sources to improve the reliability of result. The experimental result demonstrates that our method mitigates the class imbalance problem and could detect new malware families. The best f-measure achieves 0.99. This study evaluates practical performance of macro malware detection methods on an imbalanced dataset. In addition, we analyze the detail and reveal detection rates in malware families. This study also reveals robustness of language models as a feature extraction method against an imbalanced dataset. The mitigation technique is particularly effective to Doc2vec rather than LSI. The result also shows that LSI is more robust than Doc2vec to a class imbalance problem.

This paper produces the following contributions:

- 1) Proposes an improved method of detecting macro malware with a mitigation technique for proxy logs.
- 2) Evaluates practical performance of macro malware detection methods on an imbalanced dataset.
- 3) Reveals detection rates in malware families.
- 4) The mitigation technique is particularly effective to Doc2vec rather than LSI.
- 5) LSI is more robust than Doc2vec to a class imbalance problem.

The structure of this paper is shown as follows. Section 2 describes related work. Section 3 describes malicious VBA macros and section 4 provides Natural Language Processing (NLP) techniques. Section 5 presents our method and section 6 demonstrates the performance. Finally, we discuss the results and conclude this paper.

II. RELATED WORK

A. MS DOCUMENT FILE

In regard to intrusion detection, one of the hot research areas is detection, visualization, and classification of malware [26]. Our method examines MS document files without executing files on a computer. Several methods without dynamic analysis are proposed to examine MS document files. Office-MalScanner is a basic tool to detect malicious MS document files [27]. This tool scans the entire MS document file for generic shellcode patterns or embedded objects. Chen *et al.* developed a malicious document detector with the similar techniques [28]. In malicious document files, executable files are frequently embedded. The executable files are used to detect malware, because benign document files usually do not contain executable files [12]. These executable files involve file structure of document files. Otsubo *et al.* developed a tool

to detect anomaly file structure of document files containing executable files [29]. They examined hundreds of these malicious document files and found these document files do not conform to legitimate file format strictly.

The 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 [13]. 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 [30]. This model is based on their structural feature extraction methodology [13].

Another approach is a visualization based method for malware detection [31]. In this approach, the target file is converted into an image to examine. Deep learning methods have brought outstanding performance on image classification. Convolutional Neural Network (CNN) is applied to detect malicious document files [32]. This method converts a document file into images and attempts to detect shellcode patterns with CNN. Yakura *et al.* used CNN with Attention Mechanism to analyze imaged binary samples [33], [34].

These methods can be applied to document files for detecting new malicious document files. Some methods may detect malicious document files which contain VBA macros. These methods, however, do not examine the contents of VBA macros.

B. VBA MACRO

In regard to 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 [14]. This method extracts important features from p-code opcode with Term Frequency-Inverse Document Frequency (TFIDF), which is a popular method to define word importance. P-code is a set of mnemonic instructions executed by the engine. Our method uses raw VBA macro code and does not have to translate the code into p-code opcode. Santos *et al.* demonstrated that the choice of characteristics intrinsic to the VBA code that forms a macro could become an effective method for the classification of malware [15]. They used four classification algorithms: Binary Decision Trees, Support Vector Machines, Random Forest and Neural Networks. Aboud *et al.* proposed a method for detecting VBA macro malware using five classifiers: K-nearest Neighbors, Decision Tree, Random Forest, and Gaussian Naive Bayes [16]. Their method uses statistical features such as the number of specific kinds of variables. Kim *et al.* focused on the obfuscation techniques and proposed an obfuscated macro code detection method using machine learning models [17]. Their method uses 15 static features of obfuscation patterns. Hence, their method cannot detect plain macro malware.

To overcome these problems, raw VBA macro code was divided into words and used as features with Natural Language Processing (NLP) techniques. These extracted words

Algorithm 1 A Sample of Malicious Code

```

1: Sub AutoOpen()
2: Dim xHttp: Set xHttp = CreateObject
   ("Microsoft.XMLHTTP")
3: Dim bStrm: Set bStrm = CreateObject
   ("Adodb.Stream")
4: xHttp.Open "GET", <URL>, False
5: xHttp.Send
6: With bStrm
7:   Type = 1 '//binary
8:   Open
9:   write xHttp.responseBody
10: .savetofile "malware.exe", 2
11: End With
12: Shell ("malware.exe")
13: End Sub

```

were used to construct a Doc2vec model for detecting new malicious VBA macros [18], [19]. A Doc2vec model is based on the context of words in the documents. Fake text vectors may be used to mitigate the class imbalance problem [23]. While this method improves the sensitivity of minority class, this method is not evaluated on imbalanced dataset. Moreover, this method is limited to a Doc2vec model. Another popular language model is LSI and used to represent VBA macros [21], [22]. An LSI model is based on the word frequency in the documents. These methods simply extract words from both benign and malicious VBA macros. Therefore, these methods may not be robust on an imbalanced dataset. To demonstrate this problem, an imbalanced dataset with many benign samples will be used.

III. MALICIOUS VBA MACRO**A. BEHAVIOR**

Malicious VBA macros are usually embedded into MS document files. Typical MS document files conform to Compound File Binary (CFB) file format or Office Open XML (OOXML) file format. CFB file format is a binary file format used by Microsoft Office 2003 and earlier. OOXML is a collection of separate files and folders in a compressed zip package, and used by Microsoft Word 2007 and later. Many extensions conform to both file formats support VBA macros. VBA is a programming language supported in Office programs, and provides many useful functions. VBA macros are a series of commands that can be executed automatically to perform a task. Algorithm 1 shows how to perform malicious activities.

This simple example downloads a malicious file from the URL and executes it. Thus, attackers abuse its useful functions to compromise other computers. They send a MS document file with malicious VBA macros to the target computers. Typical malicious VBA macros are used to drop or download. The former is called dropper and the latter is called downloader. Once a malicious document file is opened, only a single click is required to activate the malicious VBA macro.

Dropper contains the main body in itself. The main body is encoded or obfuscated by several techniques. Hence, dropper can obtain persistent access to the computer without Internet connection. In contrast, downloader downloads the main body from the Internet as its name suggests. Therefore, downloader requires Internet connection to obtain persistent access to the computer. Thus, malicious VBA macros tend to contain functions to download or extract the main body. These functions appear in the code and can be useful for detecting malicious VBA macros.

B. OBFUSCATION

Malicious VBA macros are frequently obfuscated by several techniques. The typical obfuscation techniques are divided into 3 approaches.

Encoding

converts parameters with reversible algorithms. Several functions are used for encoding strings. For example, some functions such as Asc(), Hex(), and Chr() vary characters to the number of the ASCII code. These functions convert strings into numerous numerical characters. Therefore, typical malicious VBA macros tend to contain these functions and formatted numerical characters.

Replacing

converts strings into random strings. Several functions are used for replacing strings. For example, some functions such as Replace(), Right(), or Left() replace strings to other random strings. These functions mainly convert function names and variable names into random strings, because these names can be defined arbitrarily. Therefore, typical malicious VBA macros tend to contain these functions or random strings.

Splitting

divides strings into other characters. This technique is effective to avoid anti-virus programs. The divided characters are restored to its original strings by join operators such as "and" or "plus".

Thus, malicious VBA macros tend to contain names of these functions and characteristic strings. These features can be useful for detecting malicious VBA macros.

IV. NLP TECHNIQUES**A. BAG-OF-WORDS**

Bag-of-Words (BoW) is a fundamental model to extract feature vectors from documents. BoW represents the frequency of a word in documents and produces 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 simply assigns a word its corresponding column. Therefore, this model does not consider word meaning or context. In this study, BoW is used as a baseline to evaluate the performance of our method.

B. Doc2vec

To represent word meaning or context, Word2vec was produced [35]. Word2vec are shallow neural networks trained to

reconstruct linguistic contexts of words. Word2vec requires a corpus of documents and produces a vector space of several hundred dimensions. In this space, each unique word in the corpus is assigned a corresponding vector. These word vectors are positioned in the vector space such that similar words are located in close proximity to one another. Word2vec represents a word in documents. Paragraph Vector is the extension of Word2vec to represent a document [36]. Doc2vec is an implementation of the Paragraph Vector. In our method, Doc2vec is used to represent the features of VBA macros.

C. TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY

Term Frequency-Inverse Document Frequency (TFIDF) is one of the most popular models to define word importance. TFIDF score is defined as follows:

$$TFIDF_{i,j} = TF_{i,j} \times \log_2 \frac{D}{DF_i}$$

The $TF_{i,j}$ is the frequency of a word i in a document j . The DF_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 DF_i . In this model, as a word appears rarely in an entire corpus and appears frequently in a document, the TFIDF score increases. In our method, the TFIDF scores are used to construct a LSI model.

D. LATENT SEMANTIC INDEXING

Latent Semantic Indexing (LSI) or Latent semantic analysis (LSA) is one of techniques in topic modeling. The core idea is to decompose a matrix of documents and words into a separate document-topic matrix and a topic-word matrix. In this model, a row indicates a vector corresponding to a word, giving its relation to each document. A column indicates a vector corresponding to a document, giving its relation to each word. In practice, raw counts do not work particularly well because they do not account for the significance of each word in the document. In other words, this matrix is sparse and redundant across its many dimensions. Therefore, dimensionality reduction can be performed using truncated Singular Value Decomposition (SVD). SVD reduces dimensionality by selecting only the r largest singular values, and only keeping the first r columns or rows. To select the largest values, usually the TFIDF scores are used. In our method, LSI is also used to represent the features of VBA macros.

V. PROPOSED METHOD

A. STRUCTURE

This paper proposes an improved method of detecting macro malware on an imbalanced dataset. Figure 1 shows the structure of the proposed method.

Our method is a combination of previous NLP-based detection models. To mitigate the class imbalance problem, our method selects the frequent words from benign and malicious VBA macros respectively. A previous study for proxy logs revealed that frequent words are effective to mitigate a class imbalance problem [25], [37]. Therefore, our method

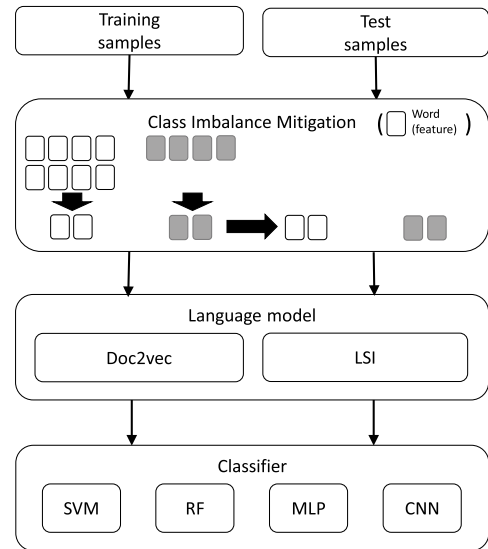


FIGURE 1. Structure of the proposed method.

applies this method for mitigation. For feature extraction, previous method used a Doc2vec model [18], [19], [23] or an LSI model [21], [22]. To compare the performance of these methods, our method uses both Doc2vec and LSI models for feature extraction. In addition, our method uses 4 popular classifiers.

B. TRAINING PHASE

The training procedure of our method is shown in Algorithm 2.

In the training phase, our method extracts words from benign and malicious VBA macros. These samples are labeled and obtained from web pages such as Virus Total.¹ Firstly, our method extracts VBA macros from MS document files. The source code is divided into words (line 1 to 7). A space character and special characters are used as the delimiter. For example, the words (Sub, Auto, Open, MsgBox, Hello, World, End, Sub) are extracted from Algorithm 3.

Secondly, our method selects the frequent words from benign and malicious VBA macros respectively (line 8 to 10). Thirdly, our method constructs the Doc2vec and LSI models from these words (line 11 to 14). The Doc2vec and LSI models convert benign and malicious VBA macros into feature vectors (line 15 to 21). Finally, classifiers are trained by these feature vectors with their labels (line 22 to 23).

C. TEST PHASE

The test procedure of our method is shown in Algorithm 4.

In the test phase, our method examines unknown samples. Firstly, our method extracts words from MS document files in the same way (line 1 to 4). Secondly, these words are converted into feature vectors by the Doc2vec and LSI models (line 5 to 8) which were constructed in the training phase. Finally, the trained classifiers predict the label from these feature vectors (line 9 to 12).

¹<https://www.virustotal.com/>

Algorithm 2 Training Procedure

```

1: /* Extract VBA macros */
2: for all malicious VBA macros  $m$  do
3:    $mw \leftarrow$  extract words from  $m$ 
4: end for
5: for all benign VBA macros  $b$  do
6:    $bw \leftarrow$  extract words from  $b$ 
7: end for
8: /* Selects frequent words */
9:  $smw \leftarrow$  Select frequent words from  $mw$ 
10:  $sbw \leftarrow$  Select frequent words from  $bw$ 
11: /* Construct a language models */
12: construct a Doc2vec model from  $smw, sbw$ 
13: construct a TFIDF model from  $smw, sbw$ 
14: construct a LSI model from the TFIDF
15: /* Convert words into vectors */
16: for all malicious words  $mw$  do
17:    $mv \leftarrow$  Call language models ( $smw$ )
18: end for
19: for all benign words  $bw$  do
20:    $bv \leftarrow$  Call language models ( $sbw$ )
21: end for
22: /* Train classifiers with the labeled vectors */
23: Call classifiers ( $mv, bv$ )
24: return

```

Algorithm 3 A Sample of Code

```

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

```

D. CLASSIFIER

Our method uses supervised learning models to predict unknown samples. We selected 4 classifiers: SVM (Support Vector Machine), RF (Random Forests), MLP (Multi-layer Perceptron), and CNN (Convolutional Neural Networks). These classifiers are popular in the various field and have different features.

E. IMPLEMENTATION

Our method was implemented with Python 3.6 in an environment as shown in Table 1. Olevba,² a script to parse OLE and OpenXML files is used to extract VBA macros from MS document files. The SVM and RF models are implemented with scikit-learn-0.21.2,³ a machine learning library which contains many classification algorithms. Chainer-6.0.0,⁴ a deep learning framework is used to implement the MLP and CNN models. The parameters are optimized by grid search, which exhaustively generates candidates from a grid of parameter values. Gensim-3.7.3⁵ is used to implement

²<https://github.com/decalage2/oletools/wiki/olevba/>

³<https://scikit-learn.org/>

⁴<https://docs.chainer.org/>

⁵<https://radimrehurek.com/gensim/>

Algorithm 4 Test Procedure

```

1: /* Extract VBA macros */
2: for all unknown VBA macros  $u$  do
3:    $uw \leftarrow$  extract words from  $u$ 
4: end for
5: /* Convert words into vectors */
6: for all unknown words  $uw$  do
7:    $uv \leftarrow$  Call language models ( $uw$ )
8: end for
9: /* Predict labels with the trained classifiers */
10: for all unknown vectors  $uv$  do
11:    $label \leftarrow$  Call classifiers ( $uv$ )
12: end for
13: return

```

TABLE 1. Environment.

CPU	IntelCorei9-7900X 3.3GHz
Memory	DDR4 128GB
GPU	GeForce GTX 1080 Ti 11G
OS	Windows 10 Home

TABLE 2. The number of samples obtained from Virus Total and Stack Overflow.

	2015		2016		2017	
	ben ign	mali cious	ben ign	mali cious	ben ign	mali cious
VT	561	862	1127	1150	2085	1049
SO	2533	-	10,215	-	11,702	-
total	3094	862	11,342	1150	13,787	1049

the LSI model. Gensim has many functions related to NLP techniques such as BoW, Doc2vec, or LSI.

VI. EVALUATION**A. DATASET**

To evaluate our method, actual VBA macros were obtained from Virus Total (VT)⁶ and Stack Overflow (SO).⁷ These actual VBA macros contain both benign and malicious VBA macros. We selected all MS document files containing VBA macros during 2015-2017 from Virus Total. Their file extensions include doc, docx, xls, xlsx, ppt, and pptx. These samples judged malicious by more than half of anti-virus vendors are labeled as malicious. In addition, we extracted benign macros from all questions in Stack Overflow using vba and excel tags. The benign samples are judged benign by all anti-virus vendors. We compared their hash values and removed duplicated samples. The VBA macros were extracted by olevba. Table 2 shows the number of samples.

Each year in the table indicates the period the samples were uploaded for the first time. Samples in each year are randomly distributed. Furthermore, we identified these samples with Windows Defender Antivirus.⁸ Table 3 shows the top 5 malware families in each dataset.

⁶<https://www.virustotal.com/>

⁷<https://stackoverflow.com/>

⁸<https://www.microsoft.com/en-us/windows/windows-defender/>

TABLE 3. Top 5 malware families in each dataset.

period	No.	family name
2015	1	TrojanDownloader:O97M/Donoff
	2	TrojanDownloader:O97M/Adnel
	3	TrojanDownloader:O97M/Bartallex
	4	TrojanDownloader:W97M/Adnel
	5	TrojanDownloader:W97M/Donoff
2016	1	TrojanDownloader:O97M/Donoff
	2	None
	3	Trojan:O97M/Madeba.A!det
	4	TrojanDownloader:JS/Swabfex.P
	5	Virus:W97M/Thus.GB
2017	1	TrojanDownloader:O97M/Donoff
	2	None
	3	Virus:W97M/Thus.GB
	4	Trojan:Win32/Tigre!rfn
	5	Virus:X97M/Metcol.A

TABLE 4. Confusion matrix.

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

“None” indicates that the samples are not defined by Windows Defender Antivirus, but detected by other anti-virus vendors. Thus, our dataset is distributed and contains a wide variety of malware samples. We are aware that the later samples contain completely new malware families. Note that these malware families are not contained in the earlier samples. As the trained model with the earlier samples could detect the later samples, it suggests that our method could detect completely new malware families. Thus, the timestamp of the dataset is very important for evaluation.

B. EVALUATION METRICS

To evaluate the performance, accuracy, precision, recall, and f-measure are used. These metrics are defined as follows:

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

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

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

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

Table 4 shows the confusion matrix.

In our experiment, TP indicates detecting malicious VBA macros correctly. Our dataset consists of 3061 positive and 28,223 negative samples, which is highly imbalanced. While the minority class accounts for only 10%, accuracy may not be appropriate to evaluate the performance. Hence, we also provide precision, recall, and f-measure of positive class.

C. EXPERIMENTAL METHOD

Initially, we conduct 5-fold cross validation with samples in 2015 to seek the optimum parameters and evaluate generalization performance. Next, we conduct time series analysis to reveal practical performance. Samples in 2015 for training and samples in 2016 for testing are used to compare with previous methods [18], [19], [21]–[23]. These previous methods

TABLE 5. Main optimized parameters of each classifier.

classifier	paramter
SVM	kernel:linear, C:0.5 probability:True
RF	criterion:gini max features:10 min samples split:3 min samples leaf:1 number of estimators:200
MLP	3 fully connected layers, epoch:30 optimizer: Adam activation function: ReLU hidden layer: 500 units
CNN	2 convolution layers, epoch:30 optimizer: Adam activation function: ReLU hidden layer: 16 and 64 chanel filter size: 3

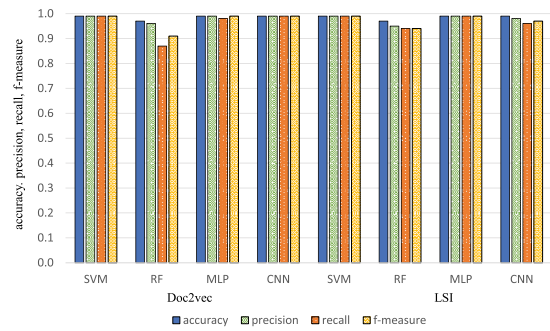


FIGURE 2. Generalization performance of the 5-fold cross validation on samples in 2015.

do not mitigate the class imbalance problem. To evaluate persistency, the testing samples are varied to ones in 2017. Finally, the training samples are varied to ones in 2016.

D. RESULT

We optimized parameters by grid search. The dimension of LSI is adjusted to 400. The main optimized parameters of each classifier are shown in Table 5.

The other parameters are set to default values.

Figure 2 shows generalization performance of the 5-fold cross validation on samples in 2015.

Overall, each performance achieves at least 0.96 except RF. Even RF performs at least 0.87. The precision has produced better results than recall. One reason for this is that the dataset contains many negative (benign) samples. Therefore, we conclude that the generalization performance is appropriate.

Figure 3 shows the comparison of the time series analysis with previous methods. As the previous methods use an SVM classifier, we use the same classifier in this experiment.

In comparison, our methods produced better performance than previous methods without mitigation. In particular, the Doc2vec with mitigation produced the best performance. The best f-measure achieves 0.99. In the previous methods, it is remarkable that Doc2vec performs less accurate than BoW. The low recall indicates many false negatives, in other

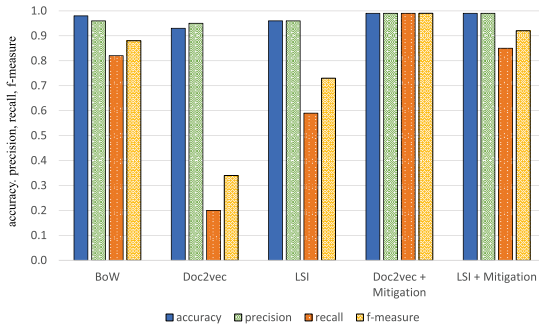


FIGURE 3. Comparison with previous methods.

TABLE 6. Required time of time series analysis.

model	BoW	Doc2vec	LSI
feature extraction	-	44.8s	11.2s
training	1.0s	1.8s	2.2s
test	0.4s	0.8s	0.7s
model	-	Doc2vec + mitigation	LSI + mitigation
feature extraction	-	33.0s	4.1s
training	-	1.1s	1.3s
test	-	0.1s	0.5s

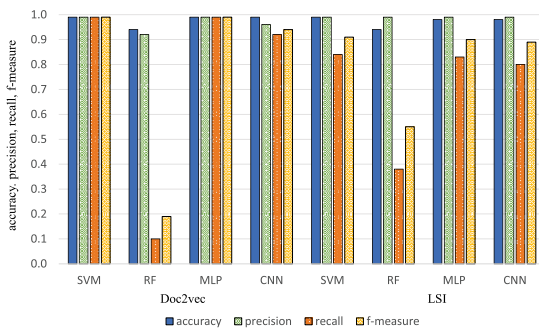


FIGURE 4. Practical performance of all combinations of the classifiers and language models.

words overlooking malicious VBA macros. Table 6 shows the required time for this experiment.

In comparison, our methods require less time than previous methods. This is because our methods reduce the unique words to construct a language model. The previous method with Doc2vec requires more time to construct a language model. However, all methods do not demand much time for detecting. Thus, our method requires only half a second for examining 12,565 samples in 2016. Therefore, our method is more effective than previous methods on the imbalanced dataset.

Figure 4 shows the practical performance of all combinations of the classifiers and language models.

In comparison, SVM with both language models produced the best performance. Therefore, we provide the performance of an SVM classifier in the remaining procedures. Table 7 shows the required time of all classifiers.

In this dataset, SVM is the fastest. Neural networks such as MLP or CNN require more training time than other models. A GPU reduces the training time dramatically. From the result, the time complexity seems to depend on each classifier.

TABLE 7. Required time of all classifiers.

model	RF	SVM	MLP	MLP (GPU)	CNN	CNN (GPU)
training	4.2s	1.1s	30.3s	11.4s	5m43s	13.3s
test	0.3s	0.1s	0.6s	0.1s	7.4s	0.2s

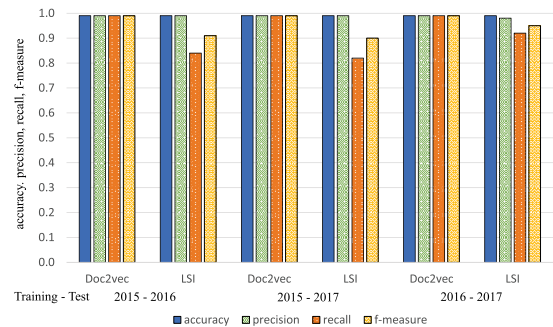


FIGURE 5. Persistency of the Doc2vec and LSI.

TABLE 8. Detection rates of top 10 unknown families in samples in 2016.

No.	family name	DR
1	TrojanDownloader:JS/Swabfex.P	50/50
2	Virus:W97M/Thus.GB	30/30
3	TrojanDownloader:O97M/Donoff.CD	19/20
4	TrojanDownloader:O97M/Donoff!rfn	19/20
5	TrojanDownloader:O97M/Zinunlate.A	19/19
6	TrojanDropper:O97M/Vibro.A	17/17
7	TrojanDownloader:O97M/Donoff!map	15/16
8	Virus:W97M/Marker.BR	13/13
9	TrojanDropper:O97M/Donoff	12/13
10	Virus:X97M/Mailcab.A	10/10

5 shows the persistency of the Doc2vec and LSI.

Regarding to Doc2vec, the original performance remains flat. The best f-measure achieves 0.99. In regard to LSI, the original performance achieves 0.91. Varying testing samples to ones in 2017, the performance is slightly reduced due to aging. Varying training samples to ones in 2016, the performance is improved. The best f-measure achieves 0.95. Thus, retraining the model with new samples is effective to improve the accuracy. As a result, the most effective combination is the SVM and Doc2vec. The training samples were discovered over a year ago, nevertheless the f-measure maintains flat even on an imbalanced dataset.

VII. DISCUSSION

A. ACCURACY

We analyzed detection rates of new malware families. Table 8 shows the detection rates of top 10 new malware families in samples in 2016.

These families contain both downloaders and droppers. Note that these new malware families are not included in the training dataset. Thus, our method could detect completely new malware families regardless of the family type.

B. COMPARISON

Table 9 shows comparison with other methods.

Previous methods are evaluated by 10-fold cross validation with several dozens of samples [14] or thousands of samples [17]. This means they used 90% of samples for training

TABLE 9. Comparison of detection rates with other methods.

method	sample	CV	time series	imbalanced and time series
[14]	158	0.99	-	-
[16]	1011	0.99	-	-
[17]	2537	0.915	-	-
[18]–[20]	2978	-	0.89	0.29
[21], [22]	7145	0.99	0.89	0.75
[23]	3842	0.99	0.91	-
Our method	31,284	0.99	-	0.99

without timestamps. While another study used samples for testing which were not present in the training samples, they used only 83 benign samples for testing [16]. VBA macros are highly used in enterprise environments with benign purposes [38]. Thus, these studies do not reveal practical performance. Moreover, they did not reveal the malware families.

Other previous studies considered time series of the dataset and selected training samples from earlier samples [18]–[22]. The detection rate was 0.89. These methods are enhanced with fake vectors and achieved 0.91 [23]. However, these methods are not evaluated on an imbalanced dataset. Moreover, they did not reveal detection rates in malware families.

We conducted 5-fold cross validation with more than 30,000 samples and the detection rate was almost perfect. Note that the condition is more severe than previous studies [14], [17]. In addition, we constructed an imbalanced dataset to evaluate practical performance. The experimental result demonstrated that previous methods are not robust to an imbalanced dataset. In contrast, our method could detect completely new malware families regardless of the family type.

C. FINDINGS

In the comparison of the time series analysis, our method clearly produced better performance than previous methods without mitigation. Inexplicably, Doc2vec performed less accurate than BoW in the previous methods. On a balanced dataset, Doc2vec performed good accuracy [21], [22]. It can thus be suggested that Doc2vec is not robust on an imbalanced dataset. This is consistent with previous studies [25], [37]. Our study provides additional support for this finding.

In contrast, LSI performed more accurate than the other previous methods. LSI also performed with good accuracy on a balanced dataset [21], [22]. This result would seem to suggest that LSI is more robust than other previous methods. As far as we are aware, this is the first time that LSI is robust on an imbalanced dataset.

With the mitigation technique, Doc2vec was drastically enhanced on an imbalanced dataset. LSI was also enhanced on an imbalanced dataset with the same technique. One reason for this difference is that LSI is based on word frequency. The mitigation technique is based on the same hypothesis. Hence, the effectiveness seems to be limited. In contrast, the mitigation technique is compatible with Doc2vec. Thus, the mitigation technique is particularly effective to Doc2vec.

D. LIMITATIONS

We are aware that our study may have two limitations. In this study, we created an imbalanced dataset from more than 30,000 samples. This dataset may not represent the typical samples. Regarding to the training data, this suggests room for improvement. If we could extract representative samples for the training data, the performance would be improved. Regarding the test data, this suggests more practical environment. Our method was evaluated with the maximum samples as far as we know. However, these samples do not completely represent all actual samples. Apparently, it is not feasible to obtain all actual samples. The best method to mitigate this is obtaining more samples. However, obtaining many labeled samples is a challenging task.

This study also evaluated the required time of our method. According to the result, the time complexity seems to depend on each classifier. To evaluate the time complexity, more samples are required. Hence, this limitation is also related to the number of samples. In addition, time complexity analysis of each classifier is a challenging task. Further data collection would be required to determine exactly how the practical performance and time complexity are.

VIII. CONCLUSION

In this paper, we propose an improved method of detecting macro malware on an imbalanced dataset. To mitigate the class imbalance problem, we apply a mitigation technique for proxy logs to macro malware. The experimental result with more than 30,000 samples demonstrates that previous methods are not effective on an imbalanced dataset. Our method could detect completely new malware regardless of the family type. The best f-measure achieves 0.99. The mitigation technique is particularly effective to Doc2vec. The result also reveals that LSI is more robust than Doc2vec.

Our method requires almost half a second for investigating more than 10,000 samples. Thus, one of our future work is to implement a real time detection system. We can implement our method on a mail server or proxy server to examine files in real time.

REFERENCES

- [1] P. Singh, S. Tapaswi, and S. Gupta, "Malware detection in PDF and office documents: A survey," *Inf. Secur. J., Global Perspective*, vol. 29, no. 3, pp. 134–153, May 2020.
- [2] C. Ulucenk, V. Varadharajan, V. Balakrishnan, and U. Tupakula, "Techniques for analysing PDF malware," in *Proc. 18th Asia-Pacific Softw. Eng. Conf. (APSEC)*, Ho Chi Minh, Vietnam, Dec. 2011, pp. 41–48.
- [3] D. Stevens, "Malicious PDF documents explained," *IEEE Secur. Privacy Mag.*, vol. 9, no. 1, pp. 80–82, Jan. 2011.
- [4] H. V. Nath and B. M. Mehtre, "Ensemble learning for detection of malicious content embedded in PDF documents," in *Proc. IEEE Int. Conf. Signal Process., Informat., Commun. Energy Syst. (SPICES)*, Feb. 2015, pp. 1–5.
- [5] M. Iwamoto, S. Oshima, and T. Nakashima, "A study of malicious PDF detection technique," in *Proc. 10th Int. Conf. Complex, Intell., Softw. Intensive Syst. (CISIS)*, Fukuoka, Japan, Jul. 2016, pp. 197–203.
- [6] S. H. T. Mavric and C. K. Yeo, "Online binary visualization for pdf documents," in *Proc. Int. Symp. Consum. Technol. (ISCT)*, May 2018, pp. 18–21.

- [7] N. Nissim, A. Cohen, J. Wu, A. Lanzi, L. Rokach, Y. Elovici, and L. Giles, "Sec-lib: Protecting scholarly digital libraries from infected papers using active machine learning framework," *IEEE Access*, vol. 7, pp. 110050–110073, 2019.
- [8] A. Corum, D. Jenkins, and J. Zheng, "Robust PDF malware detection with image visualization and processing techniques," in *Proc. 2nd Int. Conf. Data Intell. Secur. (ICDIS)*, South Padre Island, TX, USA, Jun. 2019, pp. 108–114.
- [9] F. Schmitt, J. Gassen, and E. Gerhards-Padilla, "PDF scrutinizer: Detecting javascript-based attacks in PDF documents," in *Proc. 10th Annu. Int. Conf. Privacy, Secur. Trust (PST)*, Paris, France, N. Cuppens-Boulahia, P. Fong, J. García-Alfaro, S. Marsh, and J. Steghöfer, Eds., Jul. 2012, pp. 104–111.
- [10] D. Liu, H. Wang, and A. Stavrou, "Detecting malicious javascript in PDF through document instrumentation," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Atlanta, GA, USA, Jun. 2014, pp. 100–111.
- [11] A. Lemay and S. P. Leblanc, "Is eval () evil: A study of JavaScript in PDF malware," in *Proc. 13th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Nantucket, MA, USA, Oct. 2018, pp. 13–22.
- [12] M. Mimura, Y. Otsubo, and H. Tanaka, "Evaluation of a brute forcing tool that extracts the RAT from a malicious document file," in *Proc. 11th Asia Joint Conf. Inf. Secur. (AsiaJICIS)*, Aug. 2016, pp. 147–154.
- [13] A. Cohen, N. Nissim, L. Rokach, and Y. Elovici, "SFEM: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods," *Expert Syst. Appl.*, vol. 63, pp. 324–343, Nov. 2016.
- [14] R. Bearden and D. C.-T. Lo, "Automated microsoft office macro malware detection using machine learning," in *Proc. IEEE Int. Conf. Big Data (Big-Data)*, Boston, MA, USA, J.-Y. Nie, Z. Obradovic, T. Suzumura, R. Ghosh, R. Nambiar, C. Wang, H. Zang, R. A. Baeza-Yates, X. Hu, J. Kepner, A. Cuzzocrea, J. Tang, and M. Toyoda, Eds., Dec. 2017, pp. 4448–4452.
- [15] S. D. L. Santos and J. Torres, "Macro malware detection using machine learning techniques—a new approach," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, Porto, Portugal, P. Mori, S. Furnell, and O. Camp, Eds., Feb. 2017, pp. 295–302.
- [16] E. Aboud and D. O'Brien, "Detection of malicious VBA macros using machine learning methods," in *Proc. 26th AIAI Irish Conf. Artif. Intell. Cogn. Sci. Trinity*, vol. 2259, Dublin, Ireland, R. Brennan, J. Beel, R. Byrne, J. DeBattista, and A. C. Junior, Eds., Dec. 2018, pp. 374–385.
- [17] S. Kim, S. Hong, J. Oh, and H. Lee, "Obfuscated VBA macro detection using machine learning," in *Proc. DSN*, 2018, pp. 490–501.
- [18] H. Miura, M. Mimura, and H. Tanaka, "Macros finder: Do you remember loveletter?" in *Proc. 14th Int. Conf. Inf. Secur. Pract. Exper. (ISPEC)*, Tokyo, Japan, Sep. 2018, pp. 3–18.
- [19] H. Miura, M. Mimura, and H. Tanaka, "Discovering new malware families using a linguistic-based macros detection method," in *Proc. 6th Int. Symp. Comput. Netw. Workshops (CANDARW)*, Nov. 2018, pp. 431–437.
- [20] M. Mimura and H. Miura, "Detecting unseen malicious VBA macros with NLP techniques," *J. Inf. Process.*, vol. 27, pp. 555–563, Sep. 2019.
- [21] M. Mimura and T. Ohminami, "Towards efficient detection of malicious VBA macros with LSI," in *Proc. 14th Int. Workshop Secur. (IWSEC)*, in Lecture Notes in Computer Science, vol. 11689, N. Attrapadung and T. Yagi, Eds. Tokyo, Japan: Springer, Aug. 2019, pp. 168–185.
- [22] M. Mimura and T. Ohminami, "Using LSI to detect unknown malicious VBA macros," *J. Inf. Process.*, vol. 28, pp. 493–501, Sep. 2020.
- [23] M. Mimura, "Using fake text vectors to improve the sensitivity of minority class for macro malware detection," *J. Inf. Secur. Appl.*, vol. 54, Oct. 2020, Art. no. 102600.
- [24] S. Marchal and N. Asokan, "On designing and evaluating phishing webpage detection techniques for the real world," in *Proc. 11th USENIX Workshop Cyber Secur. Experimentation Test (CSET)*, Baltimore, MD, USA, C. S. Collberg and P. A. H. Peterson, Eds., Aug. 2018, pp. 1–8.
- [25] M. Mimura, "Adjusting lexical features of actual proxy logs for intrusion detection," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102408.
- [26] R. Komatwar and M. Kokare, "A survey on malware detection and classification," *J. Appl. Secur. Res.*, pp. 1–31, Aug. 2020.
- [27] F. Boldewin, "Analyzing msoffice malware with officemalscanner," Tech. Rep., Jul. 2009.
- [28] C.-K. Chen, S.-C. Lan, and S. W. Shieh, "Shellcode detector for malicious document hunting," in *Proc. IEEE Conf. Dependable Secure Comput. (DSC)*, Taipei, Taiwan, Aug. 2017, pp. 527–528.
- [29] Y. Otsubo, M. Mimura, and H. Tanaka, "O-checker: Detection of malicious documents through deviation from file format specifications," Black Hat USA, Tech. Rep., 2016.
- [30] N. Nissim, A. Cohen, and Y. Elovici, "ALDOCX: Detection of unknown malicious microsoft office documents using designated active learning methods based on new structural feature extraction methodology," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 3, pp. 631–646, Mar. 2017.
- [31] K. Kancherla and S. Mukkamala, "Image visualization based malware detection," in *Proc. IEEE Symp. Comput. Intell. Cyber Secur. (CICS)*, Apr. 2013, pp. 40–44.
- [32] M. Mimura, Y. Otsubo, H. Tanaka, and A. Goto, "Is emulating 'binary grep in eyes' possible with machine learning?" in *Proc. CANDAR*, 2017, pp. 337–343.
- [33] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, "Malware analysis of imaged binary samples by convolutional neural network with attention mechanism," in *Proc. 8th ACM Conf. Data Appl. Secur. Privacy (CODASPY)*, Tempe, AZ, USA, Mar. 2018, pp. 127–134.
- [34] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, "Neural malware analysis with attention mechanism," *Comput. Secur.*, vol. 87, Nov. 2019, Art. no. 101592.
- [35] T. Mikolov, W.-T. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proc. Hum. Lang. Technol. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, L. Vanderwende, H. Daumé, III, and K. Kirchhoff, Eds. Atlanta, GA, USA: Westin Peachtree Plaza Hotel, Atlanta, The Association for Computational Linguistics, Jun. 2013, pp. 746–751.
- [36] V. Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. 31th Int. Conf. Mach. Learn. (ICML)*, Beijing, China, Jun. 2014, pp. 1188–1196.
- [37] M. Mimura and H. Tanaka, "A linguistic approach towards intrusion detection in actual proxy logs," in *Proc. 20th Int. Conf. ICICS*, Lille, France, Oct. 2018, pp. 708–718.
- [38] S. C. Vitel, G. Balan, and D. B. Prelicean, "Improving detection of malicious office documents using one-side classifiers," in *Proc. 21st Int. Symp. Symbolic Numeric Algorithms Sci. Comput. (SYNASC)*, Timisoara, Romania, Sep. 2019, pp. 243–247.



MAMORU MIMURA received the B.E. and M.E. degrees in engineering from the National Defense Academy of Japan, in 2001 and 2008, respectively, the Ph.D. degree in informatics from the Institute of Information Security, in 2011, and the M.B.A. degree from Hosei University, in 2014. From 2001 to 2017, he was a member of the Japan Maritime Self Defense Force. From 2011 to 2013, he was with the National Information Security Center. Since 2014, he has been a Researcher with the Institute of Information Security. Since 2015, he has been with the National Center of Incident Readiness and Strategy for Cybersecurity. He is currently an Associate Professor with the Department of Computer Science, National Defense Academy of Japan.

...