IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 October 2023, accepted 2 December 2023, date of publication 5 December 2023,
date of current version 13 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3339827

RESEARCH ARTICLE

A Feasibility Study on Evasion Attacks Against NLP-Based Macro Malware Detection Algorithms

MAMORU MIMURA[1] **AND RISA YAMAMOTO**[2]
[1]National Defense Academy of Japan, Yokosuka, Kanagawa 2398686, Japan
[2]Japan Ground Self-Defense Force, Shinjuku-ku, Tokyo 162-8801, Japan

Corresponding author: Mamoru Mimura (mim@nda.ac.jp)

This work was supported by JSPS KAKENHI under Grant 21K11898.

ABSTRACT Machine learning-based models for malware detection have gained prominence in order to detect obfuscated malware. These models extract malicious features and endeavor to classify samples as either malware or benign entities. Conversely, these benign features can be employed to imitate benign samples. With respect to Android applications, numerous researchers have assessed the hazard and tackled the problem. This evasive technique can be extended to other malicious scripts, such as macro malware. In this paper, we investigate the potential for evasive attacks against natural language processing (NLP)-based macro malware detection algorithms. We assess three language models as methods for feature extraction: Bag of Words, Latent Semantic Analysis, and Paragraph Vector. Our experimental result demonstrates that the detection rate declines to 2 percent when benign features are inserted into actual macro malware. This approach is effective even against advanced language models.

INDEX TERMS Macro malware, machine learning, evasion attack, LSA, paragraph vector.

I. INTRODUCTION

Contemporary antivirus programs predominantly employ a pattern matching methodology to detect malware [1]. Malicious actors can effortlessly evade detection by employing obfuscation techniques. Machine learning-based models for malware detection have gained prominence in order to detect obfuscated malware. These models extract malicious features and endeavor to classify samples as either malware or benign entities. Just as malicious features exist, benign samples possess their own distinctive characteristics. These benign features can be employed to imitate benign samples. For instance, numerous investigations have indicated the vulnerabilities of machine learning-based detection techniques, including adversarial attacks [2], [3], [4], [5], [6], [7], [8]. skillful adversaries can evade machine learning-based malware detection models by incorporating benign code. In relation to Android applications, several researchers have evaluated the associated risks and endeavored to address this matter. Repackaging attacks represent a frequently encountered assault type. In this form of attack, assailants modify widely used applications obtained from application marketplaces, embed malicious code, and subsequently distribute the altered applications. One reason behind this trend is the ease with which attackers can obtain source code from APK files. This evasion technique can also be extended to other types of malicious scripts. For example, macro malware is written in a scripting language, so similar techniques can be easily applied. Despite this, no evasion attacks have been attempted against macro malware detection models.

In this paper, we use a macro malware detection model as a target for evasion attacks. The purpose of this research is to attempt evasion attacks against macro malware detection models and evaluate their feasibility. Existing macro malware detection models include feature extraction using natural language processing (NLP) technology. However, in the existing research on evasion attacks, there are almost no

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Pau.

studies that focus on natural language processing technology. While conventional machine learning models require fixed features, these NLP algorithms have the capacity to extract variable features. The primary focus of previous studies lies in classifiers [2], [3], [4], [5], [6], [7], [8]. Limited attention has been given to evasion attacks involving variable features. Furthermore, some studies focusing on evasion attacks hardly discuss the feasibility [6]. These attacks may be not feasible on actual environment. To adequately evaluate the associated risks, it is imperative that the feasibility of evasion attacks is duly considered.

Our research questions are as follows.

1) Are evasion attacks possible on macro malware?
2) Are evasion attacks effective to language models for feature extraction?
3) What is the most resilient language model for feature extraction?

Our contributions can be summarized as follows:

1) Our evasion attacks demonstrate efficacy even when confronted with sophisticated language models.
2) The utilization of both Latent Semantic Analysis (LSA) and Paragraph Vector (PV) resulted in a noteworthy decline in detection rates, reaching a mere 27 percent.
3) LSA emerges as the most resilient method for feature extraction.
4) On the other hand, Paragraph Vector exhibits susceptibility to evasion attacks, resulting in an immediate decrease in the detection rate to 37 percent.

The subsequent sections of this paper are structured as follows: Section II provides the related works. Section III describes NLP algorithms. Section IV provides our scheme and Section V demonstrates the experimental result. Section VI discusses the result and reveals our findings. Finally, Section VII concludes this paper.

II. RELATED WORK

This study delves into the realm of exploring the plausibility of evasion attacks targeting certain natural language processing NLP-based algorithms used for malware detection. This section explains the novelty associated with evasion attacks, detection of macro malware, and the utilization of NLP techniques.

A. EVASION ATTACKS

There have been numerous studies into evasion attacks targeting machine learning-based malware detection techniques. Srndic and Laskov [9] investigated the effectiveness of evasion attacks against the SVM and Random Forest classifiers with PDF samples. They assumed that the adversary knows the features of the detection method and modified the features to mimic benign PDF samples. In their experiments, the significant drop in detection rates was observed. Xu et al. [10] proposed a method to automatically find variants to evade structural feature-based PDF malware classifiers. They evaluated the effectiveness of their system in evading existing PDF malware classifiers such as PDFrate [11] and Hidost [12]. Grosse et al. [6] expanded the method originally proposed by [13] to attack Android malware detection. They adapted it to handle binary features while at the same time preserving the malicious functionality. Abaid et al. [2] presented a statistical analysis of the impact of adversarial evasion attacks on various linear and non-linear classifiers with Android malware. They showed non-linear classifiers such as Random Forest and Neural Network to be more resilient to these attacks. Hu and Chen [14] proposed a generative adversarial network (GAN) based algorithm named MalGAN to generate adversarial malware examples, which are able to bypass black-box machine learning-based detection models. Their method used 160-dimensional binary features based on 160 system level APIs. Chen et al. [4] explored the feasibility of constructing crafted malware samples and proposed KuafuDet, a two-phase learning enhancing approach that learns mobile malware by adversarial detection. They showed that their method can significantly reduce false negatives on more than 250,000 mobile applications. Maiorca et al. [8] provided a comprehensive taxonomy of the different approaches used to generate PDF malware and of the corresponding learning-based detection systems. They categorized threats targeted against learning-based PDF malware detectors using a framework in the field of adversarial machine learning. Ehteshamifar et al. [5] presented a method of testing the capability of malware scanners to cope with evasions. We applied their method to malicious PDF documents and presented how current PDF evasions affect 41 malware scanners. Chen et al. [3] proposed several methods against CNN-based malware detectors. They used saliency vectors to represent the features. These saliency vectors can be generated by using the Gradient-weighted Class Activation Mapping (Grad-CAM) method [15]. Huang et al. [7] investigated evasion attacks to an machine learning-based malware detectors and conducted performance evaluations. In this study, they extracted 491 API features from log files of executable files. Their attack can be executed by inserting benign API calls to the source code. Abusnaina et al. [16] provided a GEA approach to preserve the functionality and practicality of the generated adversarial sample through a careful embedding of a benign sample to a malicious one. They showed that adversarial attack methods are able to misclassify IoT malware samples as benign. Chen et al. [17] proposed a method of applying optimal perturbations onto Android APK that can successfully deceive the machine learning detectors. The adversarial examples can also deceive recent machine learning-based detectors that rely on semantic features such as control-flow-graph. Furthermore, Li et al. [18] developed an adversarial-example attack method based on their bi-objective GAN to evade both malware detection and adversarial example detection.

Thus, the primary emphasis of evasion attacks against machine learning-based malware detection methods lies in the domains of PDF or Android malware. One contributing factor to this is the ease with which attackers can access the unencrypted contents or source code of these files. In these files, attackers can manipulate the unencrypted contents or source code to evade machine learning-based malware detection methods. Some alternative studies have endeavored to apply evasion attacks to executable files or IoT malware. Nevertheless, macro malware has received limited attention from researchers. Evasion attacks can indeed be employed against macro malware, given their straightforward nature and susceptibility to modifications. Furthermore, numerous prior studies on evasion attacks have concentrated on machine learning models employing fixed features. In contrast, our focus centered on employing NLP algorithms as feature extraction methods. These NLP algorithms offer flexibility and enable the extraction of variable features. However, there exists a scarcity of studies concerning evasion attacks utilizing variable features.

B. MACRO MALWARE DETECTION

Some researchers proposed detection methods for macro malware with machine learning. Bearden and Chia-Tien Lo [19] proposed a method of classifying MS Office files containing macros as malicious or benign using the K-Nearest Neighbors machine learning algorithm. They extracted the features from the p-code opcode with TFIDF and n-grams. Santos and Torres [20] proposed a method to detect macro malware with machine learning algorithms. They extracted several binary features from the source code. Kim et al. [21] investigated the obfuscation techniques and proposed an obfuscated macro code detection method using five machine learning classifiers. They extracted 15 discriminant static features of the obfuscated VBA macros. Constantin et al. present a study on the detection performance of MSOffice-embedded malware [22]. Their detection models were trained and tested using a very large database of malicious and benign MSOffice documents (1.8 million files), collected over a long period of time (1995-2021). Yan et al. propose DitDetector for macro malware detection, which leverages bimodal learning based on deceptive images and text [23]. They extracted preview images of documents based on an image export SDK of Oracle and textual information from preview images based on an open-source OCR engine.

This paper focused on NLP algorithms as feature extraction methods. NLP algorithms are used to extract the features of VBA macros. For instance, PV was used to represent VBA macros [24], [25], [26], [27]. LSA was also used to extract the features [28], [29], [30].

Thus, several detection methods use NLP algorithms as feature extraction methods. According to our experimental result, these methods can be vulnerable to our evasion attacks. However, as far as we know, these methods are not evaluated against evasion attacks. This study could evaluate the risk of evasion attacks.

C. NLP-BASED DETECTION

NLP algorithms are also used to extract the features in other security fields. For instance, PV was used to represent proxy logs for intrusion detection [31]. This method was extended to represent network packets with a protocol analyzer [32], [33]. While these methods detect slight malicious traffic from massive benign traffic, they did not deal with the class imbalance problem. To mitigate the class imabalance problem, the important featuress from both classes were equally selected [34]. Thus, NLP algorithms can be applied to represent network traffic.

Furthermore, PV was used to extract the features of JavaScript snippets [35], [36] Ndichu et al. proposed a malicious JavaScript detection method with Abstract Syntax Tree (AST) for code structure representation and PV [37]. Wang et al. proposed an NLP-based memory corruption detection system [38]. This method identifies memory operation functions through a semantic-aware source code analysis automatically. NLP algorithms are also applied to represent PowerShell source code [39], [40]. Thus, NLP algorithms are popular to represent the features of source code.

Another application is extracting printable strings from executable files [41], [42], [43]. These strings are nonsensical at first sight, but may be useful for NLP algorithms to detect malware. Wang et al. provided an anomaly detection based approach to detect stealthy impersonation malware using OS level provenance graphs [44]. They used PV to learn the embedding of the paths of provenance graphs. Thus, NLP algorithms are used to extract the various features. Therefore, our evasion attacks may have great effect on many machine learning based detection systems.

III. NATURAL LANGUAGE PROCESSING ALGORITHMS

In this study, NLP algorithms serve as feature extraction methods for classification. This section delineates the evaluation of three language models conducted in the purview of this study.

A. BAG OF WORDS

Bag of Words (BoW) is a fundamental approach used to transform a document into a feature vector. When working with a corpus of documents, distinct words are extracted from the entire collection. This method essentially converts a document into a feature vector, where each unique word corresponds to an element in the vector, sharing the same index. As the number of unique words increases, the number of elements in the feature vector also grows. However, certain insignificant words may not significantly contribute to the document classification process. To efficiently represent a document, it becomes necessary to reduce the dimensionality of the feature space. This simplistic model solely captures

word frequencies. It is important to note that this model fails to encapsulate the contextual importance of words.

B. LATENT SEMANTIC ANALYSIS

Latent Semantic Analysis (LSA) or Latent Semantic Indexing (LSI) is a method employed to diminish the dimensionality. This model generates topics consisting of interconnected words in a given document. To identify these interconnected words, Term Frequency-Inverse Document Frequency (TF-IDF) is commonly utilized. TF represents the frequency of a word in the documents and is defined as (1.1).

$$TF = \frac{\text{frecuancy of word x in document } A}{\text{sum of frequency of all words in document } A} \tag{1.1}$$

IDF indicates the inverse document frequency and is defined as (1.2).

$$IDF = log\left(\frac{\text{the number of all documents}}{\text{the number of documents that contain word } x}\right) \tag{1.2}$$

TF-IDF is defined as (1.3).

$$TFIDF = TF \times IDF \tag{1.3}$$

As a word exhibits infrequent occurrences across the entire corpus but appears frequently in a specific document, its TF-IDF value escalates. In this context, a matrix T incorporating TF-IDF is denoted as (1.4).

$$T = \begin{bmatrix} t_{1,1} & \cdots & t_{1,m} \\ \vdots & \ddots & \vdots \\ t_{n,1} & \cdots & t_{n,m} \end{bmatrix} \tag{1.4}$$

n indicates the number of documents. m indicates the number of unique words in all documents. The matrix T can be transformed with singular value decomposition as (1.5).

$$T = USV^{\mathrm{T}} \tag{1.5}$$

The matrices U and V consist of eigenvectors of the matrix T, with m rows and n columns respectively. These eigenvectors exhibit orthogonality to one another. Matrix U represents the topic weights of each word, while matrix V represents the topic weights of each document. The matrix S is a diagonal matrix with n rows and m columns. Except for the main diagonal elements, all other elements in the matrix are zero. These non-zero elements are referred to as singular values. By selecting the k largest singular values and their corresponding singular vectors from matrices U and V, we can obtain a rank-k approximation of matrix T with minimal error. In this model, the dimensionality, denoted by k, can be arbitrarily chosen. Hence, this model enables the reduction of dimensionality, facilitating practical analysis in a feasible timeframe.

C. PARAGRAPH VECTOR

Paragraph Vector (PV), also known as Doc2vec, presents an alternative approach to address the limitations of Bag of Words (BoW). PV extends the Word2Vec framework and aims to generate a vector representation for a document. Word2Vec employs a straightforward neural network with a singular hidden layer to learn the associated weights. In this framework, these weights correspond to the word embeddings being learned. The Skip-Gram model and the Continuous Bag of Words model (CBOW) stand out as the most renowned algorithms for creating these representations. In the Skip-Gram model, the objective of the neural network is as follows: Given a particular word in a sentence, the network predicts the likelihood of each vocabulary word being the adjacent word. Conversely, the CBOW model's neural network task is inverted: Given a context of words in a sentence, the network predicts the likelihood of each vocabulary word being the word itself. Consequently, the simple neural network can be trained, enabling the acquisition of word embeddings. While this model adequately represents individual words, it falls short in representing paragraphs or sentences. To overcome this limitation, PV builds upon Word2Vec and introduces an additional vector as input. This vector, known as the paragraph vector, is responsible for representing a paragraph. In this extension, the weights signify the paragraph embeddings being learned. Similar to Word2Vec, PV offers two prominent algorithms: the Distributed Bag of Words version of Paragraph Vector (PV-DBOW) and the Distributed Memory version of Paragraph Vector (PV-DM). PV-DBOW and PV-DM correspondingly align with the Skip-Gram and CBOW models. As a result, this framework becomes capable of representing paragraphs or sentences in a corpus.

IV. EVASION ATTACK

A. OUTLINE

This study delves into the potentiality of evasion attacks targeting NLP-based macro malware detection algorithms. Given the need for consistent features in conventional models, these NLP algorithms extract mutable features. In our framework, NLP algorithms serve as feature extraction methodologies for classification purposes. This study assesses the resilience of feature extraction methodologies against evasion attacks. Figure 1 illustrates our evasion attack scheme.

Our methodology comprises two components: the training process and the evasion process.

The training process mirrors a conventional supervised machine learning procedure. Language models are generated and employ feature extraction from labeled training samples. The classifier is then trained using the labeled features and is capable of classifying unknown samples.

The evasion process encompasses the parsing of two types of samples: evasion samples and original samples. Evasion samples serve as the basis for identifying benign features for

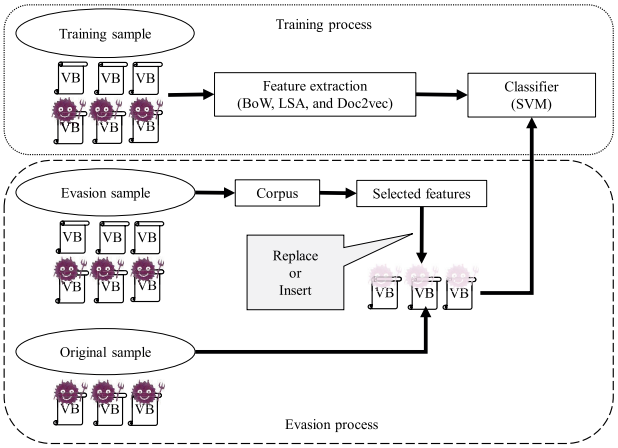

FIGURE 1. The scheme of evasion attacks.

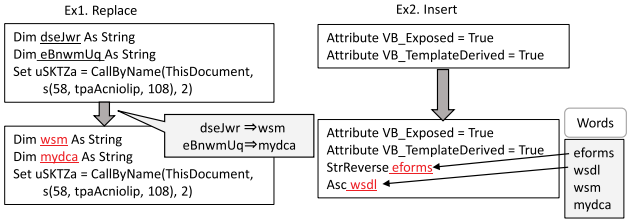

FIGURE 2. Illustration of the replacing and inserting methods.

evasion attacks. It should be noted that the evasion samples encompass both benign and malicious samples. To extract effective benign features, an analysis of the malicious samples is required. This is because frequent features may be shared between both sample types. The original samples, which are malicious in nature, act as the recipients of the inserted benign features. During this process, benign features are selected from the corpus of evasion samples. The chosen benign features are then inserted into the original samples. The resulting modified original samples are subsequently employed in conjunction with the trained classifier.

B. THEORY

The formula of our evasion attack can be defined as a function $y = F(x)$, which maps the input x to the corresponding output y. Attackers can modify a sample x', and the differences between x' and x are small. The modified sample x' from the original sample x can be formalized as below:

$$x' = x + \delta x = x + min\|z\| s.t. F(x + z) \neq F(x) \quad (1)$$

where δx is the minimal perturbation z contributing to misclassification, according to a norm $\| \bullet \|$ which is suitable for the input domain. In the field macro malware detection, the input x of $F(x)$ is usually a VBA script, and the output y is the corresponding label. The goal of the attacker is to turn the output of a macro malware detection model from malicious to benign. In this study, the attacker simply adds benign features in the original sample to deceive the machine learning model without changing its original functions.

C. TRAINING PROCESS

The initial step in the training process entails the extraction of all lexemes from the training samples. The source code of each training sample is partitioned into individual lexemes. Thereafter, our method replaces some random hexadecimal patterns, strings, digits, and arrays. Because many malicious macros are obfuscated with these patterns. If our method does not replace these patterns, each word is handled as each feature respectively. These words, however, have a common context or meaning. Thus, our method replaces these patterns into single words to improve accuracy. These lexemes, combined from all samples, give rise to a corpus. This corpus serves as the foundation for constructing three distinct language models: Bag-of-Words (BoW), Latent Semantic Analysis (LSA), and Doc2vec. These language models are leveraged to extract the features inherent in the samples. The extracted features, accompanied by their respective labels, are then utilized to train a classifier. Thus, the training process embodies a fundamental amalgamation of feature extraction and classifier training. In this study, the primary focus lies on feature extraction, rather than the classifiers themselves. We undertake a comparative analysis of the three models for feature extraction. In a prior investigation, our research scrutinized various classifiers, ultimately demonstrating that Support Vector Machines (SVM) yielded the most optimal performance [30]. To evaluate the efficacy of the three feature extraction models, we employ an SVM classifier.

D. EVASION PROCESS

The evasion process endeavors to evade detection by the trained classifier. Evasion samples encompass a mixture of benign and malicious instances. The source code of an evasion sample is partitioned into lexemes. These lexemes, extracted from all samples, generate a corpus. This corpus serves as the foundation for selecting benign features. The chosen benign features are subsequently inserted into the original samples. To insert the benign features, we explore two methodologies: replacing and inserting. Figure 2 exemplifies instances of the replacing and inserting techniques.

In the replacing method, variable names in the source code are substituted with benign features. In VBA macros, users can arbitrarily define variable names as many programing languages. Even if we change variable names in the source code, the function would remain the same. Because users can arbitrarily define all variable names. In fact, dis-assemblers and de-compilers generate code with arbitrarily determined variable names. Consequently, we can substitute these names with benign features. In Figure 2, the variables "dseIwr" and "eBnwmUq" are replaced with "wsm" and "mydca," respectively, which have been selected from the evasion samples. The benign features are chosen sequentially from the feature list. In this approach, the maximum number of words replaced is contingent upon the source code.

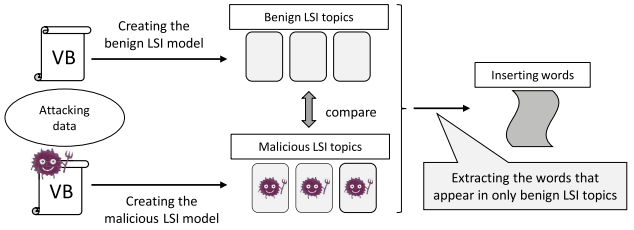

FIGURE 3. The modified approach for selecting effective features utilizing LSA topics.

TABLE 1. Environment.

CPU	Intel Core i7 (3.40 GHz)
Memory	16GB
OS	Windows 10 Home

In the inserting method, the benign features are incorporated as arguments in some functions, such as "Asc", "AscB", "AscW", "LCase", "LTrim", "Trim", "StrReverse", "Ucase", or "Split". Even if arbitrary values are assigned to these arguments, these functions operate independently and do not alter their behavior. Thus, users can arbitrarily define function arguments in these functions. This means we can insert arbitral benign features as function arguments. The benign features are selected individually from the feature list. In this approach, the insertion of benign terms can be performed irrespective of the source code.

To generate the feature list, we employ two methodologies. One straightforward approach involves extracting frequently occurring words from benign samples. In this method, the frequent words are sequentially appended to the list, starting from the most prevalent. To represent more effective features, it is imperative to exclude common features shared by both sample types. Figure 3 illustrates the modified approach.

In this approach, two additional LSA models are constructed, one from benign samples and the other from malicious samples. The LSA model serves to reduce dimensionality and generate multiple topics. A comparison between the benign and malicious topics allows for the selection of words exclusively present in the benign topics. These selected topics are then segmented into words and appended to the feature list.

E. ENVIRONMENT

Our scheme was implemented using Python 3.6. The environmental details are presented in Table 1. For the implementation of the LSA model, we utilized gensim 3.6.0,[1] whereas sckit-learn 0.22.1[2] was employed for the SVM model.

V. EVALUATION

A. DATASET

To assess the risk associated with evasion attacks, we employed genuine benign and malicious macros. The chronological order of these samples is presented in Table 2. These samples were sourced from VirusTotal.[3] Our selection criterion encompassed samples that were initially uploaded between 2016 and 2017, with file extensions of doc, docs, xls, xlsx, ppt, or pptx. For the designation of malicious samples, we identified macros that were flagged as malware by over half of the 58 anti-virus programs. Benign samples, on the other hand, were consistently identified as benign by all anti-virus programs. No sample duplication was observed.

TABLE 2. Samples in chronological order.

2016		2017	
benign	malicious	benign	malicious
1200	1150	2220	719

TABLE 3. Confusion matrix.

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

B. METRICS

To gauge the accuracy of the models, precision, recall, and F1 score were utilized as evaluation metrics. The confusion matrix is presented in Table 3.

These metrics are defined as follows.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

$$F1 = \frac{2Recall \times Precision}{Recall + Precision} \tag{2.4}$$

Our objective is to evaluate the risk associated with evasion attacks on macro malware. If our assumption holds true, the detection rate will decrease through the application of our method. Therefore, our primary focus lies on the detection rate (recall).

C. THREAT MODEL

The training and evasion samples consist of benign and malicious samples from 2016. However, the original samples are exclusively malicious samples from 2017. Hence, there exists a distinction between the training samples and the original samples. This scenario implies that the adversary possesses knowledge of the training data used for the classifier, but lacks information about the correct classifier [45]. In this study, our focus lies on feature extraction rather than classifiers. It is essential to take the practical situation into consideration. During the evaluation, the training and evasion

[1]https://radimrehurek.com/gensim/
[2]https://scikit-learn.org/stable/
[3]https://www.virustotal.com/

TABLE 4. The performance of the replacing method for the BoW model.

	No attack	Evasion attack
Accuracy	0.78	0.76
Precision	1.00	1.00
Recall	0.78	0.76
F1	0.88	0.86

processes are performed independently. In other words, both models are completely distinct. Consequently, the adversary remains unaware of the correct feature extraction method and classifier.

D. REPLACING METHOD

Prior to conducting evasion attacks, we conducted parameter optimization through grid search. We opted for the linear kernel in SVM. The dimension of LSA was set to 400. For PV, the iteration count was adjusted to 30, with a vector size of 100 and learning rate of 0.025. These parameter settings align with previous studies [28], [29].

Evasion attacks were carried out using the replacement method. The target model employed is a simple BoW model. Table 4 presents the performance results of the replacement method.

Without the evasion attack, this straightforward BoW model achieved a recall of 0.78. Upon comparing the results, it is evident that this basic evasion attack led to a mere 3 percent decrease in the detection rate. In this evasion attack, the maximum number of replaced words is determined by the number of variables present in the source code. Consequently, we cannot increase the number of words to insert. Hence, even in the case of a simple feature extraction method, the replacing method proves to be ineffective.

E. SIMPLE INSERTING METHOD

We conducted evasion attacks using the simple inserting method. In this approach, we simply selected frequent words from benign samples and randomly insert them. The number of inserted words is gradually increased. The target feature extraction methods include BoW, LSA, and PV. Figures 4, 5, and 6 illustrate the transition of accuracy.

The vertical axis represents the accuracy, precision, recall, or F1 score. The horizontal axis represents the number of inserted words. The recall score of the BoW model ultimately experienced a decrease of only 2 percent. In contrast, the detection rate of the LSA model was scarcely affected. As for the PV model, the detection rate immediately dropped to 37 percent and remained constant. Thus, the simple inserting method proves to be highly effective in the case of the simple feature extraction method using BoW. Moreover, this method also demonstrates effectiveness in the feature extraction method employing PV. LSA emerges as the most resilient model against this simple inserting method. To mitigate the impact of the feature extraction method with LSA, more sophisticated approaches are necessary.

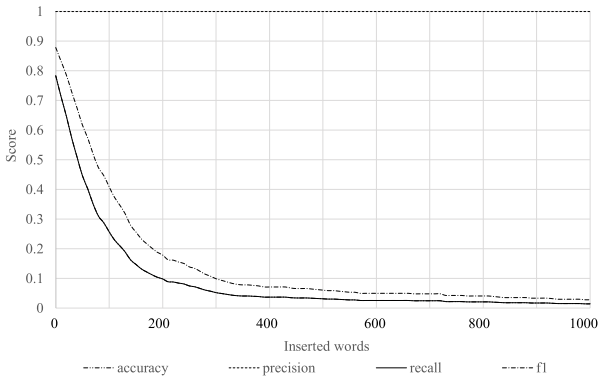

FIGURE 4. Transition of the accuracy by the simple inserting method (BoW model).

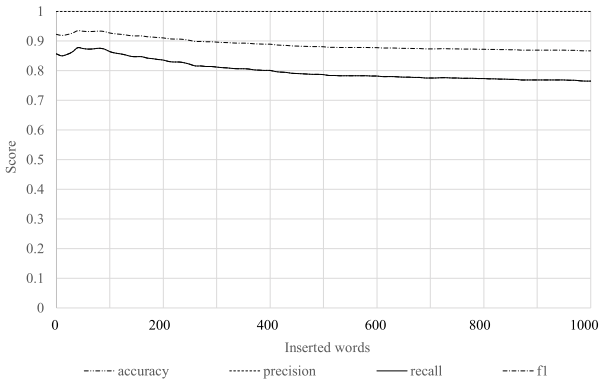

FIGURE 5. Transition of the accuracy by the simple inserting method (LSA model).

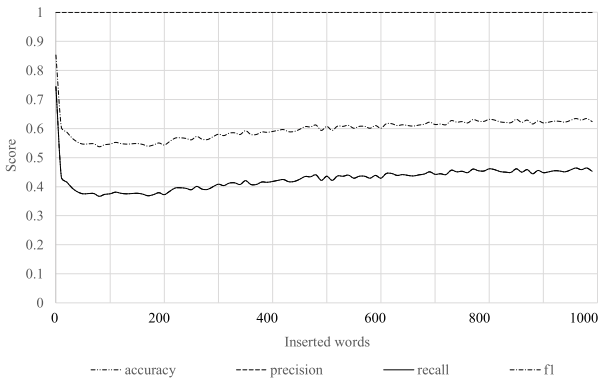

FIGURE 6. Transition of the accuracy by the simple inserting method (PV model).

F. MODIFIED INSERTING METHOD

We performed evasion attacks using the modified inserting method. In this approach, we excluded the common features between the samples. We carefully selectd impactful words from both sets and insert them at random positions. The number of inserted words gradually increased. The target feature extraction methods are LSA and PV. Figures 7 and 8 illustrate the accuracy progression.

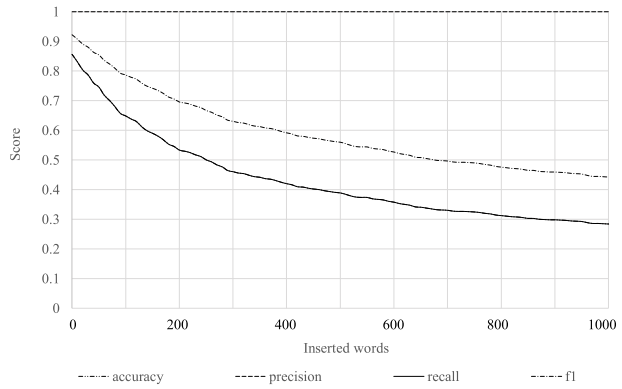

FIGURE 7. Transition of the accuracy with the modified inserting method (LSA model).

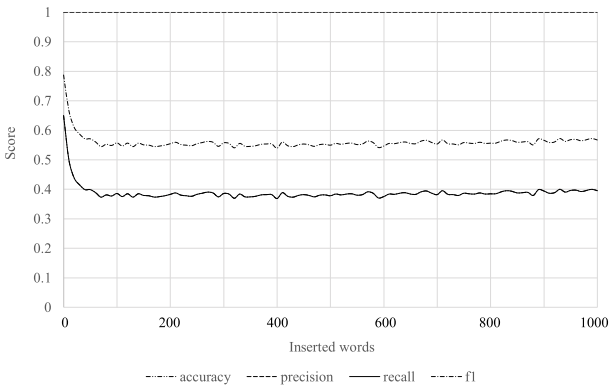

FIGURE 9. Inserting benign features at the end position (PV model).

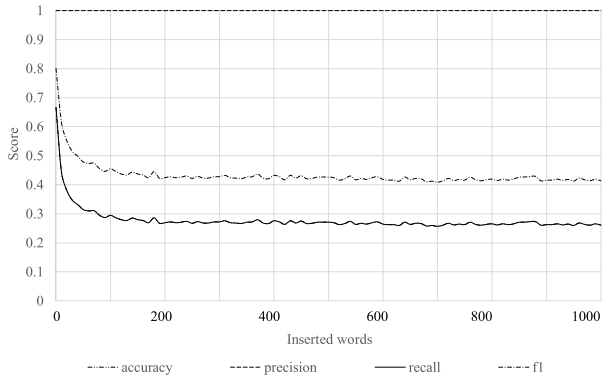

FIGURE 8. Transition of the accuracy with the modified inserting method (PV model).

The vertical axis represents the accuracy, precision, recall, or F1 score. The horizontal axis represents the number of inserted words. In the LSA model, the detection rate eventually decreased to 27 percent. As for the PV model, the detection rate experienced a sudden decrease to 27 percent. Thus, the modified inserting method proves effective in both LSA and PV feature extraction methods.

PV, which aims to predict the words surrounding the given context, might exhibit sensitivity to word order. To validate this hypothesis, we conducted an additional experiment. We carefully selected sophisticated words and inserted them at the end position. Figure 9 displays the accuracy progression.

As expected, the detection rate decreased to 37 percent. Hence, inserting benign features at random positions proves effective for PV.

VI. DISCUSSION

A. METHODOLOGY

In this study, we conducted evasion attacks using the replacing method and inserting method.

The primary contribution of our research lies in assessing the potential for evasion attacks against NLP-based macro malware detection algorithms. Previous studies have predominantly evaluated the risk using fixed features [2], [3], [4], [5], [6], [7], [8]. These studies primarily focused on classifiers with constant features, while few researchers have addressed the evasion attack problem with variable features. In contrast, our study emphasizes feature extraction rather than classifiers.

For this proposed application, we devised the evasion process involving replacing and inserting methods. This process leverages the characteristics of variable names and function arguments in the source code. These elements can be arbitrarily defined by the user, thereby ensuring that the modifications remain independent and do not alter the behavior. Some studies that primarily focus on evasion attacks often neglect to discuss the original behavior [6]. To appropriately evaluate the risk, it is crucial to ensure the feasibility of the evasion attacks.

B. EFFECTIVENESS

In the replacement method, the maximum number of words to be substituted is contingent upon the number of variables present in the source code. Hence, we were unable to augment the number of words to be inserted. Consequently, this method proved ineffective even in the case of a simple feature extraction method using BoW.

On the other hand, the insertion method offers greater flexibility. With this approach, we have the ability to increase the number of words to be inserted. We simply selected frequently occurring words from benign samples and inserted them randomly. The detection rate using BoW decreased to a mere 2 percent. Thus, the straightforward feature extraction method using BoW appears to be vulnerable to evasion attacks. The detection rate with LSA showed minimal decrease. In the case of the PV model, however, the detection rate immediately dropped to 37 percent. Therefore, LSA stands out as the most robust model against this simple insertion method.

In the modified insertion method, we excluded the common features shared by both samples and instead selected effective words. Both detection rates using LSA and PV were reduced to 27 percent. Hence, the modified insertion

method proves effective in both feature extraction methods, employing LSA and PV. In the replacing method, the maximum number of the replaced words depends on the numbers of variables in the source code. Therefore, we could not increase the words to insert. As a result, this method was not effective even in the simple feature extraction method with BoW.

C. FEASIBILITY

In summary, these feature extraction methods appear to exhibit susceptibility to evasion attacks. Specifically, the simple feature extraction method utilizing BoW proves to be highly vulnerable. Even sophisticated feature extraction methods employing LSA and PV can be circumvented by evasion attacks. The most notable outcome of the experiment is the sensitivity of PV to such attacks. In this feature extraction method, the detection rate experienced an immediate decrease with the inclusion of just a few words. Furthermore, these words could be inserted at any position in the source code, rendering evasion attacks undetectable. Hence, these feature extraction methods are susceptible to evasion attacks.

Based on the results, LSA emerges as the most resilient feature extraction method. The detection rate ultimately decreased to 27 percent. However, achieving this requires the insertion of a significant number of words. Considering that the original samples contain an average of 3300 words per file, the detection of evasion attacks may be comparatively easier.

In this experiment, the time required to train the model ranges from 1 second to 5 seconds at most. The overhead of the modification process is less than 1 second. The time required for classification is less than 1 second. Therefore, the time constraints for this attack method are considered to be minor.

D. COUNTERMEASURES

One approach to mitigate these evasion attacks is the exclusion of words that users can define arbitrarily. The feature extraction methods employed in this study extracted features from the entirety of the source code. The inserted benign features were meticulously chosen to ensure they did not alter the behavior. In essence, these selected words hold no significant importance for classification. Hence, their exclusion can serve as a countermeasure. Another approach involves retraining the model with the knowledge gained from evasion attacks. However, this method can only be applied post-evasion attacks. Therefore, both of these countermeasures should be implemented simultaneously.

E. LIMITATIONS

Our study has certain limitations. This study specifically focuses on selected NLP algorithms as feature extraction methods, which extract variable features. However, it does not consider machine learning-based methods with constant features or conventional detection methods based on pattern matching. To obtain a more comprehensive evaluation of performance, it is necessary to incorporate these conventional methods. Unfortunately, due to implementation constraints, this approach was not feasible. While in theory, these evasion attacks can preserve the behavior of malware, we did not confirm this behavior through dynamic analysis. However, this is a challenging task and beyond the scope of this study. In this study, an SVM classifier was chosen as the main target was feature extraction rather than classifiers. However, it is important to evaluate the performance of other classifiers as well. Our method was evaluated using thousands of benign and malicious samples. However, there is a scarcity of available malicious VBA macros, limiting the sample size for evaluation. Nevertheless, NLP algorithms have the ability to extract variable features and can be applied to diverse inputs. To enhance the validity of our experiments, a larger-scale dataset with recent malware samples should be utilized.

F. ETHICS

Ethics were given due consideration in this study. Our method does not require specific resources and can be easily implemented. As a consequence, potential attackers might attempt to replicate our method in order to evade detection. However, the successful implementation of our method relies on the availability of an attack dataset. urthermore, our method does not take into account conventional detection methods that employ pattern matching. These factors collectively restrict the brief utilization of our method by attackers. The primary objective of our study is to raise awareness about the risks associated with evasion attacks targeting feature extraction. We firmly believe that this study contributes to the advancement of detection methods that exhibit resilience against evasion attacks.

VII. CONCLUSION

In this study, we have elucidated the potential for evasion attacks against specific NLP-based macro malware detection algorithms. The experimental findings, utilizing actual macro malware, demonstrate a decrease in the detection rate to a mere 2 percent when employing the simple feature extraction method. Both advanced language models exhibit a reduction in their detection rates to 27 percent. Notably, the PV model exhibits a susceptibility to evasion attacks. Despite the average file size of the original samples being approximately 3300 words, the detection rate experiences an immediate decline with the insertion of just a few words. Moreover, these words can be inserted at any position in the source code. Among the feature extraction methods employed, the LSA model emerges as the most robust. To achieve a decreased detection rate, the insertion of hundreds of words is necessary. As a conclusion, we propose two countermeasures against evasion attacks.

This study primarily focused on three NLP algorithms as feature extraction methods and utilized an SVM classifier. It is essential to evaluate other models as well. Malware analysis that combines multiple detectors and

observation windows, proposing an approach based on ensemble detection is a popular topic in this field and should also be discussed in macro malware detection. While these evasion attacks can maintain the malware's behavior, the confirmation of such behavior through dynamic analysis is imperative. Our evaluation process involved thousands of benign and malicious samples. To enhance the validity of our findings, a more extensive dataset or a publicly available dataset should be considered. Further studies are warranted to implement and assess the effectiveness of the proposed countermeasures. Additionally, it is crucial to evaluate evasion attacks using adversarial learning techniques.

REFERENCES

[1] F. Biondi, T. Given-Wilson, A. Legay, C. Puodzius, and J. Quilbeuf, "Tutorial: An overview of malware detection and evasion techniques," in *Proc. 8th Int. Symp. Leveraging Appl. Formal Methods (ISoLA)*, Limassol, Cyprus, Nov. 2018, pp. 565–586, 2018.

[2] Z. Abaid, M. A. Kaafar, and S. Jha, "Quantifying the impact of adversarial evasion attacks on machine learning based Android malware classifiers," in *Proc. IEEE 16th Int. Symp. Netw. Comput. Appl. (NCA)*, Cambridge, MA, USA, Oct. 2017, pp. 1–10.

[3] B. Chen, Z. Ren, C. Yu, I. Hussain, and J. Liu, "Adversarial examples for CNN-based malware detectors," *IEEE Access*, vol. 7, pp. 54360–54371, 2019.

[4] S. Chen, M. Xue, L. Fan, S. Hao, L. Xu, H. Zhu, and B. Li, "Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach," *Comput. Secur.*, vol. 73, pp. 326–344, Mar. 2018.

[5] S. Ehteshamifar, A. Barresi, T. R. Gross, and M. Pradel, "Easy to fool? Testing the anti-evasion capabilities of PDF malware scanners," 2019, *arXiv:1901.05674*.

[6] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. D. McDaniel, "Adversarial examples for malware detection," in *Proc. 22nd Eur. Symp. Res. Comput. Secur. (ESORICS)*, in Lecture Notes in Computer Science, vol. 10493, S. N. Foley, D. Gollmann, E. Snekkenes, Eds., Oslo, Norway. Cham, Switzerland: Springer, Sep. 2017, pp. 62–79.

[7] Y. Huang, U. Verma, C. Fralick, G. Infantec-Lopez, B. Kumar, and C. Woodward, "Malware evasion attack and defense," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Portland, OR, USA, Jun. 2019, pp. 34–38.

[8] D. Maiorca, B. Biggio, and G. Giacinto, "Towards adversarial malware detection: Lessons learned from PDF-based attacks," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–36, Jul. 2020.

[9] N. Šrndic and P. Laskov, "Practical evasion of a learning-based classifier: A case study," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2014, pp. 197–211.

[10] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on PDF malware classifiers," in *Proc. 23rd Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb. 2016.

[11] N. Srndic and P. Laskov, "Detection of malicious PDF files based on hierarchical document structure," in *Proc. 20th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb. 2013.

[12] N. Šrndić and P. Laskov, "Hidost: A static machine-learning-based detector of malicious files," *EURASIP J. Inf. Secur.*, vol. 2016, no. 1, pp. 1–20, Dec. 2016.

[13] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Saarbrücken, Germany, Mar. 2016, pp. 372–387.

[14] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, *arXiv:1702.05983*.

[15] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 336–359, Feb. 2020.

[16] A. Abusnaina, A. Khormali, H. Alasmary, J. Park, A. Anwar, and A. Mohaisen, "Adversarial learning attacks on graph-based IoT malware detection systems," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Dallas, TX, USA, Jul. 2019, pp. 1296–1305.

[17] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren, "Android HIV: A study of repackaging malware for evading machine-learning detection," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 987–1001, 2020.

[18] H. Li, S. Zhou, W. Yuan, J. Li, and H. Leung, "Adversarial-example attacks toward Android malware detection system," *IEEE Syst. J.*, vol. 14, no. 1, pp. 653–656, Mar. 2020.

[19] R. Bearden and D. C. Lo, "Automated Microsoft office macro malware detection using machine learning," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Boston, MA, USA, Dec. 2017, pp. 4448–4452.

[20] S. de los Santos and J. Torres, "Macro malware detection using machine learning techniques—A new approach," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, P. Mori, S. Furnell, and O. Camp, Eds., Porto, Portugal, Feb. 2017, pp. 295–302.

[21] S. Kim, S. Hong, J. Oh, and H. Lee, "Obfuscated VBA macro detection using machine learning," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2018, pp. 490–501.

[22] S. C. Vitel, M. Lupascu, D. T. Gavrilut, and H. Luchian, "Detection of MSOffice-embedded malware: Feature mining and short- vs. long-term performance," in *Proc. 17th Int. Conf. Inf. Secur. Pract. Exper. (ISPEC)*, in Lecture Notes in Computer Science, vol. 13620, C. Su, D. Gritzalis, V. Piuri, Eds., Taipei, Taiwan. Cham, Switzerland: Springer, Nov. 2022, pp. 287–305.

[23] J. Yan, M. Wan, X. Jia, L. Ying, P. Su, and Z. Wang, "DitDetector: Bimodal learning based on deceptive image and text for macro malware detection," in *Proc. 38th Annu. Comput. Secur. Appl. Conf.*, Austin, TX, USA, Dec. 2022, pp. 227–239.

[24] 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.

[25] 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.

[26] M. Mimura and H. Miura, "Detecting unseen malicious VBA macros with NLP techniques," *J. Inf. Process.*, vol. 27, pp. 555–563, Sep. 2019.

[27] 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.

[28] M. Mimura and T. Ohminami, "Towards efficient detection of malicious VBA macros with LSI," in *Proc. 14th Int. Workshop Secur. (IWSEC)*, Tokyo, Japan, Aug. 2019, pp. 168–185.

[29] M. Mimura and T. Ohminami, "Using LSI to detect unknown malicious VBA macros," *J. Inf. Process.*, vol. 28, pp. 493–501, Sep. 2020.

[30] M. Mimura, "An improved method of detecting macro malware on an imbalanced dataset," *IEEE Access*, vol. 8, pp. 204709–204717, 2020.

[31] M. Mimura and H. Tanaka, "Leaving all proxy server logs to paragraph vector," *J. Inf. Process.*, vol. 26, pp. 804–812, Dec. 2018.

[32] M. Mimura and H. Tanaka, "Reading network packets as a natural language for intrusion detection," in *Proc. 20th Int. Conf. Inf. Secur. Cryptol. (ICISC)*, Seoul, South Korea, Nov. 2017, pp. 339–350.

[33] M. Mimura, "An attempt to read network traffic with Doc2vec," *J. Inf. Process.*, vol. 27, pp. 711–719, Nov. 2019.

[34] M. Mimura, "Adjusting lexical features of actual proxy logs for intrusion detection," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102408.

[35] M. Mimura and Y. Suga, "Filtering malicious Javascript code with Doc2Vec on an imbalanced dataset," in *Proc. 14th Asia Joint Conf. Inf. Secur. (AsiaJCIS)*, Aug. 2019, pp. 24–31.

[36] N. M. Phung and M. Mimura, "Detection of malicious Javascript on an imbalanced dataset," *Internet Things*, vol. 13, Mar. 2021, Art. no. 100357.

[37] S. Ndichu, S. Kim, S. Ozawa, T. Misu, and K. Makishima, "A machine learning approach to detection of JavaScript-based attacks using AST features and paragraph vectors," *Appl. Soft Comput.*, vol. 84, Nov. 2019, Art. no. 105721.

[38] J. Wang, S. Ma, Y. Zhang, J. Li, Z. Ma, L. Mai, T. Chen, and D. Gu, "NLP-EYE: Detecting memory corruptions via semantic-aware memory operation function identification," in *Proc. 22nd Int. Symp. Res. Attacks, Intrusions Defenses (RAID)*, Beijing, China, Sep. 2019, pp. 309–321.
[39] Y. Tajiri and M. Mimura, "Detection of malicious PowerShell using word-level language models," in *Proc. 15th Int. Workshop Secur. (IWSEC)*, in Lecture Notes in Computer Science, vol. 12231, K. Aoki and A. Kanaoka, Eds., Fukui, Japan. Cham, Switzerland: Springer, Sep. 2020, pp. 39–56.
[40] M. Mimura and Y. Tajiri, "Static detection of malicious PowerShell based on word embeddings," *Internet Things*, vol. 15, Sep. 2021, Art. no. 100404.
[41] M. Mimura and R. Ito, "Applying NLP techniques to malware detection in a practical environment," *Int. J. Inf. Secur.*, vol. 21, no. 2, pp. 279–291, Apr. 2022.
[42] M. Mimura, "Evaluation of printable character-based malicious PE file-detection method," *Internet Things*, vol. 19, Aug. 2022, Art. no. 100521.
[43] M. Mimura, "Impact of benign sample size on binary classification accuracy," *Expert Syst. Appl.*, vol. 211, Jan. 2023, Art. no. 118630.
[44] Q. Wang, W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, J. Rhee, Z. Chen, W. Cheng, C. A. Gunter, and H. Chen, "You are what you do: Hunting stealthy malware via data provenance analysis," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, San Diego, CA, USA, 2020.
[45] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, New York, NY, USA, Nov. 2017, pp. 27–38.

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 also 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.

RISA YAMAMOTO received the B.E. degree in engineering from the National Defense Academy of Japan, in 2020. She is currently a member of the Japanese Ground Self-Defense Force.