BERT for Malware Classification

Joel Alvares and Fabio Di Troia

Abstract In this paper, we aim to accomplish malware classification using word embeddings. Specifically, we trained machine learning models using word embeddings generated by BERT. We extract the “words" directly from the malware samples to achieve multi-class classification. In fact, the attention mechanism of a pre-trained BERT model can be used in malware classification by capturing information about the relation between each opcode and every other opcode belonging to a specific malware family. As means of comparison, we repeat the same experiments with Word2Vec. Differently than BERT, Word2Vec generates word embeddings where words with similar context are considered closer, being able to classify malware samples based on similarity. As classification algorithms, we used and compared Support Vector Machines (SVM), Logistic Regression, Random Forests, and Multi-Layer Perceptron (MLP). We found that the classification accuracy obtained by the word embeddings generated by BERT is effective in detecting malware samples, and superior in accuracy when compared to the ones created by Word2Vec.

1 Introduction

Malware is a computer program created with the intention to cause harm and damage to personal data, or gain unauthorized access to a user’s system. Many are the techniques used by malware to conceal their malicious intent. One way is to masquerade itself as a legitimate program. This behavior has been observed, among others, in trojans and ransomware programs [2].

Identification and classification of malware is very critical to information security. According to the Sophos 2021 threat report [23], malware programs contributed to 34% of all the breaches in a survey consisting of 3500 IT professionals who worked on remote infrastructure and cloud-based infrastructure. Each malicious

J. Alvares · F. Di Troia (✉)
San Jose State University, San Jose, CA, USA
e-mail: fabio.ditroia@sjsu.edu

M. Stamp et al. (eds.), *Artificial Intelligence for Cybersecurity*, Advances in Information Security 54, https://doi.org/10.1007/978-3-030-97087-1_7

piece of code shares common characteristics within a certain family and tends to differ from malware samples belonging to a different family. It is necessary to identify these unique characteristics which would help classify malware codes belonging to numerous families [4]. Word embeddings can be used to quantify these unique characteristics of a malware sample, and they can be generated by *state-of-the-art* machine learning models, such as BERT [26] and Word2Vec [3]. The embeddings capture useful information that serves as training features for the classification models. In this paper, the focus is on the effectiveness of the word embeddings generated in the context of malware classification.

The remainder of the paper is organized in the following Sections. It starts with a survey of relevant work in Sect. 2. Then, the building blocks of the research are introduced in Sect. 3, that is, the background of the word embedding models and the applied classification models. Next, the dataset used, the applied methodology, and the accomplished experiments and the results are analyzed in Sect. 4. Finally, Sect. 5 contains the conclusions and suggestions for future work.

2 Related Work

Malware writers are constantly analyzing computer systems and their software in search of security faults that can be exploited by specific malware programs. To obfuscate their malicious intent, such programs implement sophisticated techniques to mask them as benign software and, thus, becoming invisible to malware recognition software [18]. This is the reason malware detection has become a challenging task. A lot of malware recognition techniques rely on signature-based detection. The antivirus program that relies on signature-based detection generally computes the hash of the files and compares it with the hash of known malware signatures [28]. However, modifying the code by inserting dead code within the malicious code is one easy way to avoid detection. Furthermore, this malware recognition technique is also inefficient, because all the files of a given user are scanned and compared with known available malicious signatures, which is a time consuming process. According to [25], a number of metamorphic malware families, such as, MetaPHOR, Zmist, Zperm, Regswap, and Evol morph after each new infection. Detecting these malware samples is challenging and it can defeat signature-based detection. Metamorphic malware morphs the code by using a combination of substitution, insertion, deletion, and transposition. However, the metamorphic malware can be identified by machine learning techniques because they are able to notice the subtle differences between malware and benign samples despite the use of morphing [30]. The effectiveness of the different machine learning techniques depends on the input features extracted from the dataset. Some possible features that can be used are signatures [28], API calls [27], and opcodes [5].

Natural Language Processing (NLP) techniques extract rich information, known as word embeddings, from sentences of a language, and are able to identify the meaning of a sentence, generate sentences with similar meaning, or fill the blanks

within a sentence. The NLP models extract information of the relation of a word with every other word of a sentence. The model groups together words with similar meaning and maps them to a higher dimensional space where similar words in meaning are grouped together. This information helps NLP models accomplish several classification and prediction tasks. The NLP models can be used in the field of malware recognition to generate embeddings for malware samples. The malware samples that belong to the same family have features that are closely related. This information can be used by classifiers to group together malware samples that belong to the same family. BERT is one type of NLP model that can be used to generate word embeddings to capture information of every component of the input with respect to every other component. More details about the architecture of transformers and the attention mechanism of BERT can be found in [26], while an analysis of the attention heads of the BERT model can be found in [8]. The attention heads of BERT capture various patterns and linguistic notions.

Another example is Word2Vec, that was used in previous research to generate word embeddings for malware samples, with performance comparable to traditional machine learning techniques, such as, Hidden Markov Models and Principal Component Analysis [7]. The opcode sequences within malware samples are treated as a language in [1], and context is captured using Word2Vec. The classification is carried out using k-nearest neighbors (k-NN). The results derived by utilizing word embeddings generated by Word2Vec to achieve malware classification proves that NLP based models can extract rich features that assist with classification accuracy. This success induces to test newer NLP based models. Thus, differently than the previous work and in addition to it, we introduce the use of BERT in malware detection. BERT implements a transformer-based model that consists of encoders and decoders along with an attention mechanism [26]. The BERT model will be explained in further detail in Sect. 3. The experiments performed in this paper primarily focus on generating embeddings using BERT and comparing the classification accuracy with Word2Vec using a variety of classifiers.

3 Background

This section provides more details on the key components of this paper, that is, the NLP models and the implemented classifiers. The NLP models introduced are BERT and Word2Vec, while the classifiers are SVM, Random Forests, Logistic Regression, and MLP. The dataset, the results, and the experiments implemented using these building blocks are described in Sect. 4.

3.1 NLP Models

Natural Language Processing (NLP) is the subfield of Artificial Intelligence (AI) that enables machines to understand the language spoken by humans. The models that help achieve this result are known as NLP models. Training an NLP model from scratch is a tedious task and it requires a massive dataset and computational resources. For this reason, a pre-trained NLP model is often used to achieve the tasks related to NLP. Transfer learning is an example of technique used to transfer the knowledge gained by the model during the training phase to achieve other tasks on a different dataset to which it has never been exposed before. The tasks subject to NLP application are, for instance, sentiment analysis, next sentence prediction, word embedding generation, and more [21].

3.1.1 Word Embeddings

Word Embeddings are used in natural language processing as a representation of the words of a sentence in vector values such that words of similar meaning are grouped together in the vector space. This information can be used by classifiers to identify key features and efficiently accomplish classification. Features need to be extracted from the malware samples, which can be done by generating word embeddings from the malware samples. These word embeddings capture information and group together features that are unique to a specific malware family. They are generated using NLP based models such as Word2Vec and BERT. These word embeddings generated for every opcode in a malware sample can be represented as unit vectors and plotted in a circular heat map, as shown in Fig. 1 for, respectively, the malware families CeeInject, FakeRean, OnlineGames, Renos, and Winwebsec.

The circular heat map representation of the opcodes seem to differ for every malware family, even though the opcodes with higher frequencies across all the malware families are the opcodes $push$, mov, and add.

3.1.2 Word2Vec

Word2Vec is used to convert the input sequence of words to vectors, and map them to a higher dimensional space. The tutorial in [16] explains how Word2Vec uses neural networks to group together words with a similar meaning. For example, we can consider the following set of words:

$$w_0 = \text{“queen”}, w_1 = \text{“man”}, w_2 = \text{“woman”}, w_3 = \text{“king”}$$

In Fig. 2, we see how these words are mapped to a higher dimensional space by Word2Vec. Cosine similarity can then be used to identify words that are synonymous in nature. We can color the values using numbers, such that red

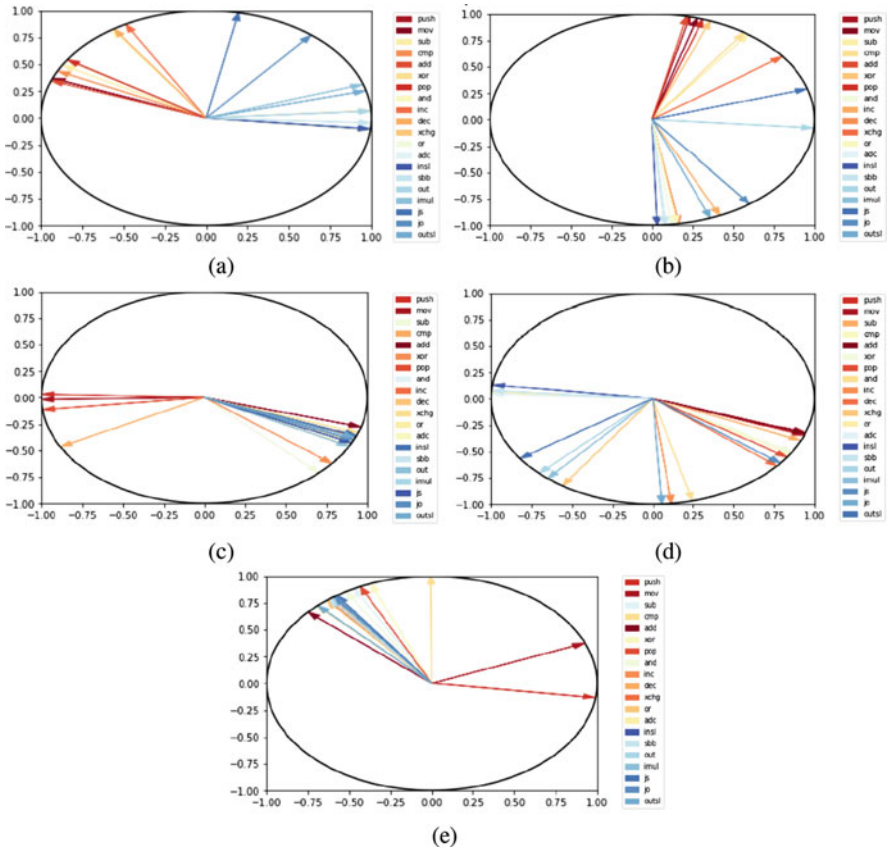

Fig. 1 Circular heatmaps for (**a**) CeeInject (**b**) FakeRean (**c**) OnlineGames (**d**) Renos (**e**) Winwebsec

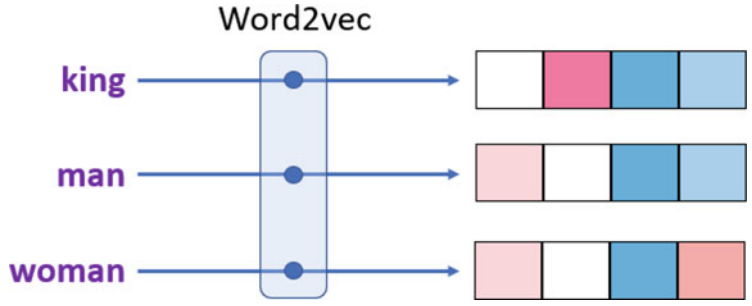

Fig. 2 Using Word2Vec to generate embeddings

represents a value close to 2, blue represents a value close to −2, and white represents a value close to 0.

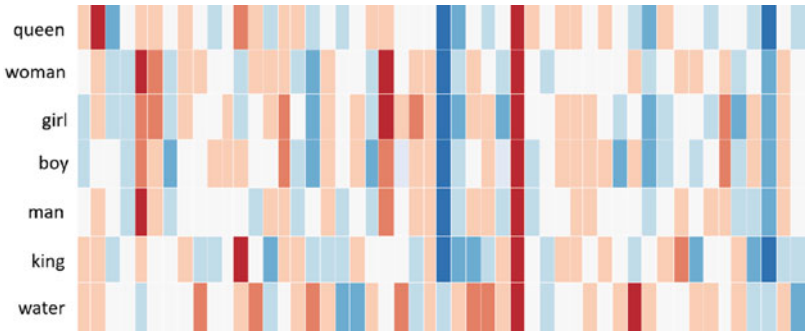

Fig. 3 Word embeddings represented as a color map

Based on Fig. 3, it can be observed that:

- The words "woman" and "girl" are considered similar to each other in several positions.
- The words "boy" and "girl" are similar in certain positions, but these positions are different from "woman" or "man". The algorithm could be capturing something similar between the words "boy" and "girl" such as "youth".
- The embeddings can be added and subtracted in order to form relations between words. For instance, in the case where the word embedding for the word "queen" is subtracted with the word embedding for the word "woman", and the word embedding for "man" is added, then the resultant word embedding is very close to the word embedding for the word "king". This can be represented as follows: "queen" − "woman" + "man" = "king"

Associating negative weights with frequently used words is another technique to improve the rate of training. In generating the output vectors, the positive weights associated with the model are all updated, while only a sample set of the negative weights are also updated. This reduces the impact of frequently used words while training the Word2Vec model. The Word2Vec model is used to generate word embedding for malware samples by using a window of size 6 and output size of 2 dimensions. We use the output generated by the Word2Vec model to generate unit vectors and plot a circular heat map which will be discussed in further detail in Sect. 3.1.1.

3.1.3 BERT

BERT is a transformer-based NLP model that is used to accomplish language-based tasks, such as, masked word prediction, sentiment classification, and more. The architecture consists of a stack of trained Transformer Encoders. BERT is able to generate the word embedding for a particular word by also taking into account the context in which it was used, known as contextualized word embeddings. The

encoder uses attention to map the input to a set of vectors which store information of a given word with respect to every other word in the sentence. For instance, if we have the following input sentence: "*The boy drank water because he was thirsty*", the word "*he*" is associated with the word "*boy*", and the BERT model can identify this relation using attention. Attention helps BERT understand other relevant words in the sentence compared to the one that is currently being processed.

As shown in Fig. 4, the BERT model can accept at most 512 words as input. In general, a sentence in natural language does not exceed 512 words but the opcodes in a malware sample can exceed such value. In our experiments, the first 400 opcodes from each malware sample were sufficient to obtain good results. The BERT model used as a part of the experiments is DistilBERT, which is a smaller version of BERT that was open sourced by the HuggingFace team [9]. DistilBERT performs similarly to BERT but it is lightweight and, hence, more efficient. The DistilBERT model used is pre-trained on the English language. However, the model is neither trained nor fine-tuned to achieve malware classification. The classification token (CLS) from BERT, used to represent sentence-level classification output, captures the information about the entire sentence. In case of a malware sample, the CLS token captures the entire information of the sample. This information can be used in malware classification because the CLS token from the generated embedding collects information that helps with classification. For instance, if there are 2000 malware samples that BERT was trained on, and if 66 is the length of the tokens in the longest malware opcode sequence, as seen in Fig. 5, only the first column representing the CLS token is extracted from the 768 hidden units of BERT. A label is assigned to each of the 2000 sentences depending on the class of the malware sample.

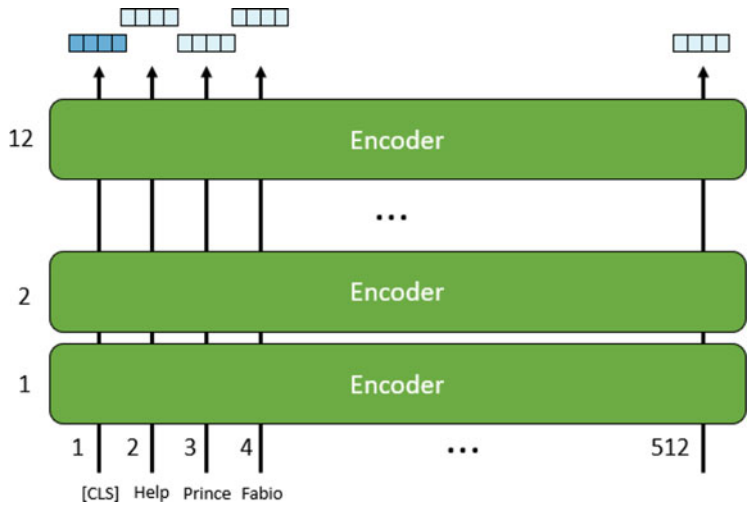

Fig. 4 Trained BERT components

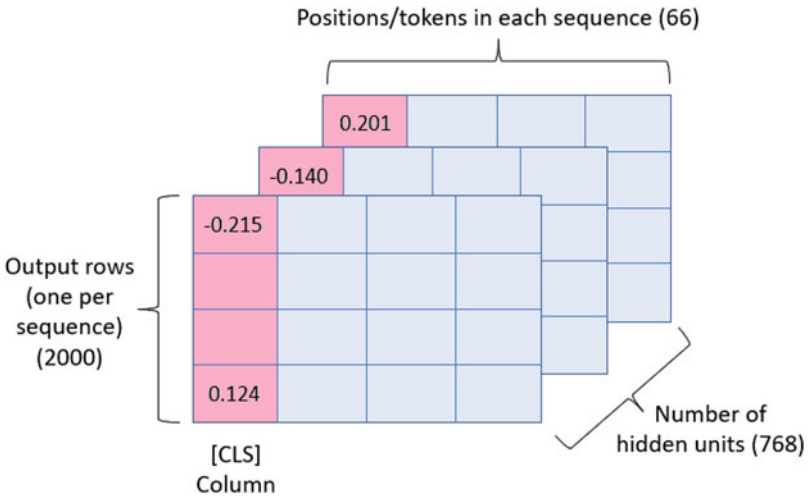

Fig. 5 Slicing BERT word embedding

3.2 Classifiers

Classification is the process of predicting the class or label of the input dataset. The input dataset is mapped to the desired output class depending on the features of the input data. The machine learning models, which enable the user to map the input data to its corresponding class, are known as classifiers. This Section will briefly introduce the classifiers used in our experiments.

3.2.1 Logistic Regression

Logistic regression is used to describe the input data and to find a correlation between them. The result of logistic regression is dichotomous in nature. A logistic regression model used to fit more than two classes is referred to as multinomial logistic regression. The model achieves classification using multinomial probability distribution. The assumption of logistic regression is a sigmoid function that can be defined as follows:

$$f(x) = \frac{1}{1 + e^{-(x)}} \tag{1}$$

The disadvantages of logistic regression are similar to linear regression. It is, in fact, prone to outliers, and assumption of linearity amongst dependent and independent variables. However, logistic regression model provides probabilities, and it is not just a classification model. It enables the user to identify the percentage

with which a certain instance was assigned to a class. A detailed explanation and various strategies guidelines for logistic regression can be found in [19].

3.2.2 SVM

The main objective of SVM is to accomplish classification within the dataset by maximizing the distance between the separating hyperplane and the dataset.

As shown in Fig. 6, the hyperplane with the maximum distance from the dataset is chosen. The support vectors are the data points closest to the hyperplane. These are used by SVM to maximize the separation between the data points and the hyperplane. SVMs can be used to identify the subtle changes in malware samples belonging to a certain family as discussed in [29]. SVM identifies that the dataset may not be linearly separable by itself. Hence, the dataset is mapped to a higher dimensional space where a separating hyperplane can classify the dataset. This approach is often referenced to as kernel trick. For example, in Fig. 7 we see that the data on the left side is not linearly separable. However, the data can be easily separated by a hyperplane if the data is mapped as seen on the right side. One of the ways to achieve this is by using a polynomial kernel. There are many kernels that can be applied, and identifying the right one can be a challenging task, but it can significantly improve the classification accuracy without causing a major computation overhead. The classification process of SVMs and the mathematical proof can be found in [24].

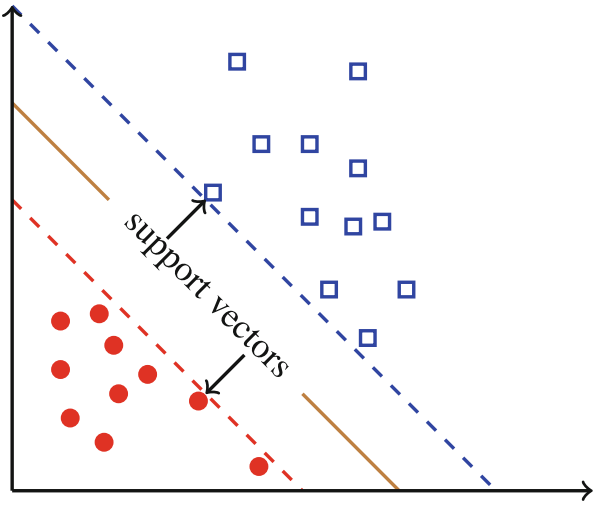

Fig. 6 SVM for binary classification

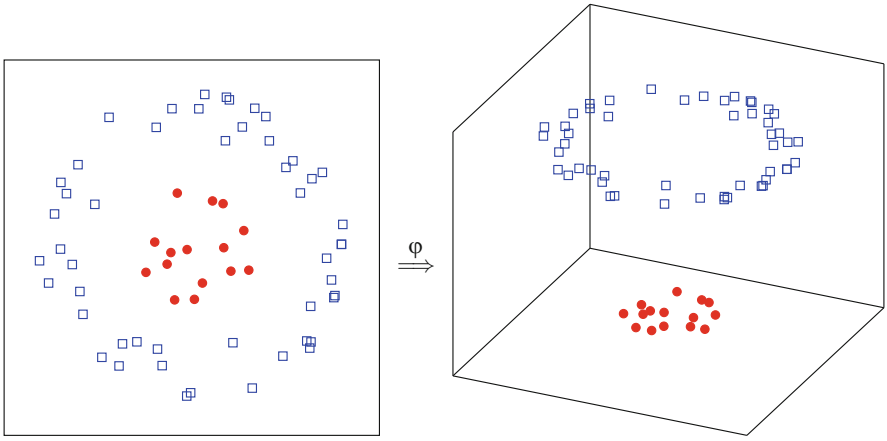

Fig. 7 Mapping input data to a higher dimension

3.2.3 Random Forests

Random Forests accomplish the classification of the dataset using an ensemble of decision trees. Every tree classifies the data independently from the others and votes for a specific class based on its prediction. The class with the highest number of votes is, then, selected as the classification output of the Random Forest. Basically, a large number of decision trees achieve the classification together as a committee, and the overall accuracy of such a committee outperforms the accuracy of an individual tree. In fact, an individual decision tree tends to overfit the input dataset, while a group of trees tends to protect each other from their individual errors. A problem that arises with Random Forests is that the decision trees may be too correlated with each other. Hence, bagging, which stands for bootstrap aggregation, is used to overcome this issue. The decision trees are formed using random samples of the training data which may or may not overlap. In this way, bagging prevents the Random Forest from overfitting the data by reducing the correlation among the decision trees. Further details on Random Forests can be found at [6].

3.2.4 MLP

A neuron, known as McCulloch-Pitts Artificial neuron [17], is the building block for a Multi Layered Perceptron (MLP). Multiple neurons are placed in different layers and the inputs of the neurons in the hidden or intermediate layers are outputs of the neurons in the previous layer. A neuron with three inputs and a single output is depicted in Fig. 8, where the inputs are X_0, X_1, and X_2, while the weights associated with these inputs are w_0, w_1, and w_2. The neuron generates an output $Y \in [0, 1]$ where 1 implies that the neuron was activated, while 0 implies that

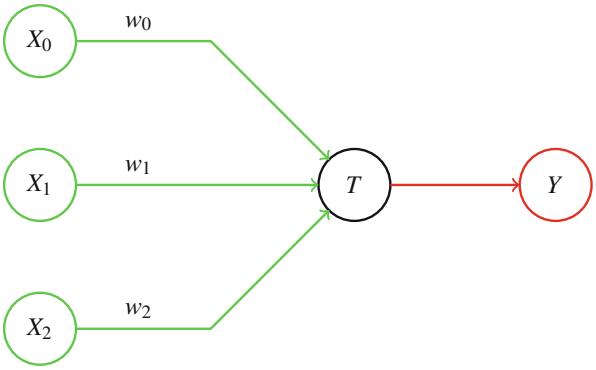

Fig. 8 Neuron of a neural network

the neuron remained inactive. The weights together with the input determine if the neuron should output or not. If the value $\sum w_i X_i$ is greater than the threshold T, then the neuron is activated. Equation 2 represents the function that a neuron of an MLP utilizes. An independent bias b is also introduced and updated during the training of the MLP.

$$f(x, y) = \sum_{i=0}(n-1)w_i X_i + b \tag{2}$$

In case of binary classification, if the Eq. 2 generates a positive value, then we classify the input as class 1, or, if the function generates a negative value, the input is classified as class 2. The decision boundary of the binary classifier is represented by the Eq. 3. The decision boundary separates the inputs into the two classes in the output dimension space.

$$f(x, y) = w_0 x + w_1 y + b \tag{3}$$

An MLP consists of multiple layers of these perceptron's, as shown in Fig. 9 which consists of two hidden layers. Each edge of the MLP has a weight associated with it, and the definitive values of the weights are finalized after the training phase. More details on the MLP architecture can be found at [20].

4 Experiments and Results

In this section, we describe dataset used for the experiments, the parameters used for the machine learning models, and their classification results accomplished on the word embeddings generated by BERT and Word2Vec.

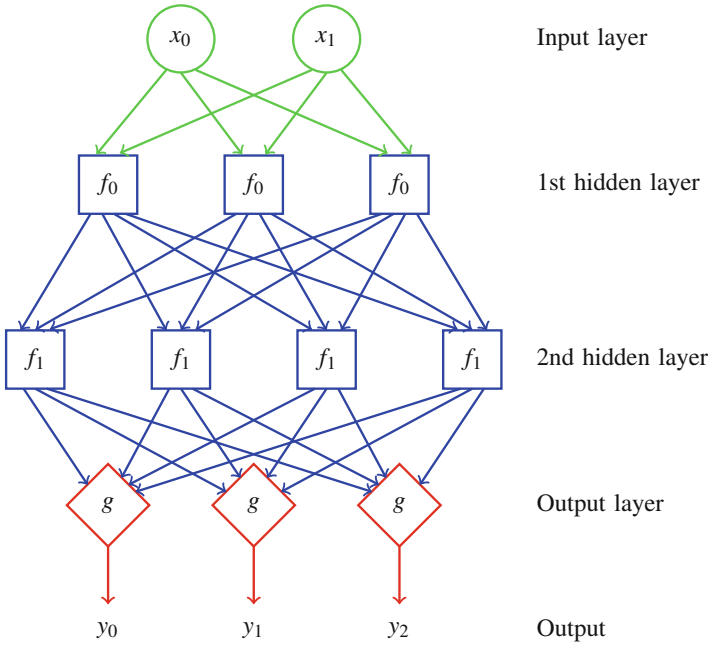

Fig. 9 Multi layer perceptron

Table 1 Malware dataset information

Malware family	Malware type	Nr. of samples
CeeInject	VirTool	899
FakeRean	Rogue	899
OnlineGames	Password stealer	900
Winwebsec	Rogue	897
Renos	Trojan downloader	900

4.1 Dataset

All our experiments were based on the malware families described in Table 1, along with the number of malware samples for each family [15].

A brief description of each malware family is given here.

1. CeeInject is malware that is generally used in combination with other malware families as it is used to conceal the other malware samples. The malware that CeeInject is used along with is installed in a user's machine without requesting any permissions [11].
2. FakeRean alerts the user for issues or viruses that do not exist on the system and asks for money in order to assist the user [14].
3. OnlineGames is used to track the login information of online games and keeps track of information of online gamers without consent [12].

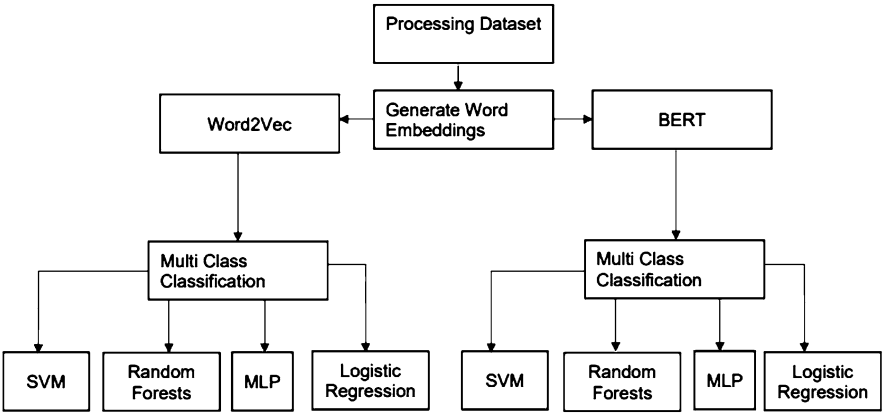

Fig. 10 Illustration of the organization of the project

4. Winwebsec is a trojan that pretends to be a legitimate antivirus software, informing the user that the system is corrupt and needs to be fixed. It tries to scare the user with the intention of extorting money [13].
5. Renos is a malware that shows to the user fake security warnings once it is downloaded and requests for payments to resolve the issues [10].

4.2 Methodology

In Fig. 10, we see an illustration of our approach. The input dataset of malware samples is processed and transformed into inputs for BERT and Word2Vec that generate the word embeddings. Then, these are directly used to train the machine learning models to achieve multi-class classification on the malware samples. The word embeddings generated are, thus, classified to their respective malware families with the help of the classifiers described in Sect. 3, that is, Support Vector Machines (SVMs), Random Forests, Multi Layer Perceptron (MLP), and Logistic Regression. In this way, the overall accuracy depends on the classification of the word embeddings which capture the essential characteristics of the malware samples.

4.3 Classifier Parameters

The parameters that were selected for the classification are shown in Table 2. We found these values to be the optimal ones by experimenting using GridSearchCV

Table 2 Parameters used by the classifiers

Classifier	Model parameter	Word2Vec	BERT
Logistic regression	C	42.1	42.1
	Solver	Lbfgs	newton-cg
	Multiclass	Auto	Multinomial
SVM	C	1000	1000
	Kernel	rbf	rbf
	Gamma	1	1
Random forests	max depth	20	20
	n estimators	100	100
MLP	Hidden layer size	(150,150,100)	(100,100,100)
	Activation function	ReLU	ReLU
	Solver	adam	adam
	Nr. of iterations	3000	10,000
	Learning rate	Constant	invscaling

from the scikit-learn library [22]. The parameters obtained are almost identical for the features generated by both BERT and Word2Vec.

4.4 Logistic Regression Results

Optimal results were obtained by the logistic regression model using the regularization parameter value $C = 42.1$. The different values for C were obtained using numpy's linspace function by dividing the range 0.0001 to 100 into 20 parts. The test accuracy of this model was 81.2% using the word embeddings generated by Word2Vec, and 83.54% using the word embeddings generated by BERT. The confusion matrices of the obtained results for BERT and Word2Vec are shown in Fig. 11.

The overall accuracy is unsatisfactory when compared with the other classifiers. One of the possible reasons is that the model is overfitting the decision boundary to the training dataset. This causes the model to perform poorly when exposed to new data.

4.5 SVM Results

Experiments were achieved on the SVM model and the ideal set of parameters that produced the maximum accuracy were selected. We tested different types of kernels, that is, radial basis function (rbf) kernel, linear, and polynomial, along with the regularization parameter C in the range 10 to 1000, and gamma value in the range 0.001 to 0.1. SVM maps the input features to a higher dimensional space in

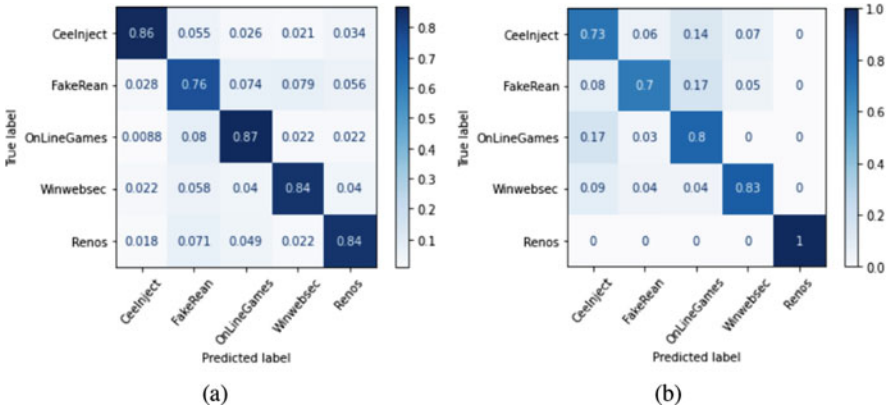

Fig. 11 Confusion matrix of logistic regression for (**a**) BERT features (**b**) Word2Vec features

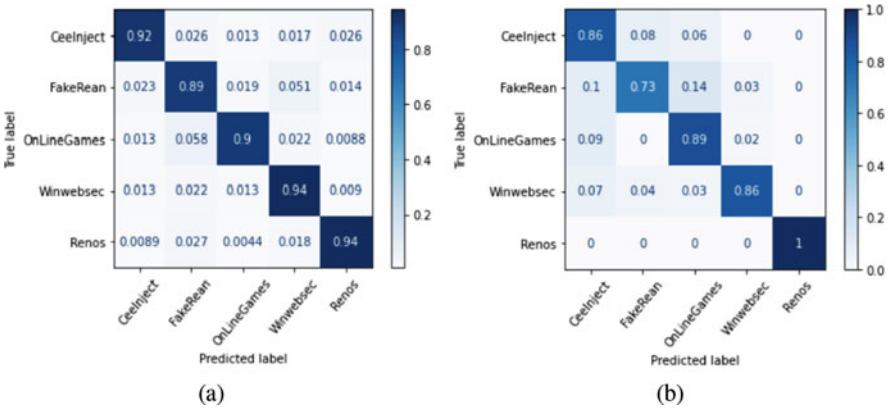

Fig. 12 Confusion matrix of SVM for (**a**) BERT features (**b**) Word2Vec features

order to form a decision boundary that separates the features into different classes. For this reason, SVM is able to successfully leverage the features in the word embeddings and group together malware samples with similar features obtaining a high classification accuracy of around 91.01% using the embedding generated by BERT. The embeddings generated by Word2Vec, instead, obtained a classification accuracy of 86.8%. The confusion matrices are shown in Fig. 12.

4.6 Random Forest Results

Random Forest is a neighborhood-based algorithm that classifies input features by grouping the ones that are closer to each other, and making decisions at

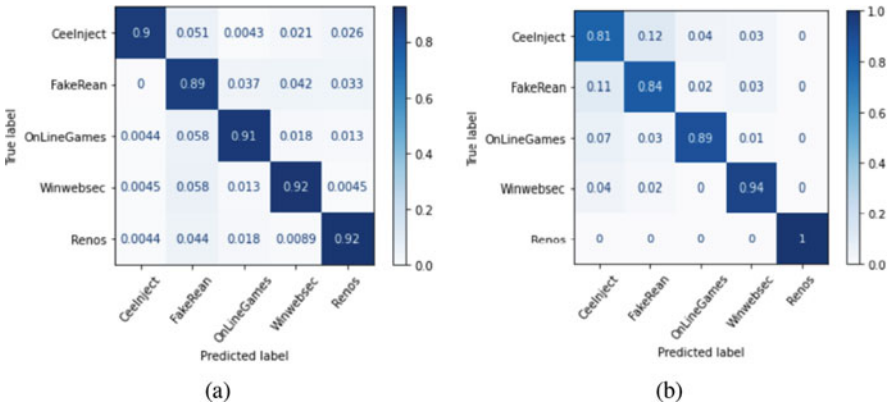

Fig. 13 Confusion matrix of random forest for (**a**) BERT features (**b**) Word2Vec features

different stages which segregate the inputs into different classes. The results of the experiments conducted show that the Random Forest classifier performs better when the number of trees and the depth is increased. The optimal parameters lead to a classification accuracy of 91.81% with embeddings generated by BERT, while the embeddings generated by Word2Vec gave a classification accuracy of 89.6%. The confusion matrices are shown in Fig. 13.

4.7 MLP Results

The multilayered perceptron performs quite closely as SVM by mapping the input features to a higher dimensional space, and accomplishing classification by forming a decision boundary to group together features that are closer to each other. A constant learning rate with a 30,30,30) hidden layer width and ReLU activation function provided the best results. The classifier converged and gave optimal outcome at around 10,000 iterations. The final accuracy obtained using the word embeddings generated by BERT was 86.83%, which is not surprisingly close to the accuracy obtained by SVM. The word embeddings generated by Word2Vec obtained a similar result with a final accuracy of around 86.6%. The confusion matrices are shown in Fig. 14.

4.8 Further Analysis

Random Forest is a neighborhood-based classification model. By our experiments, we noticed that the model performs poorly when the depth of the binary tree of the decision is shallow. It tended to overfit to the training data, as the training

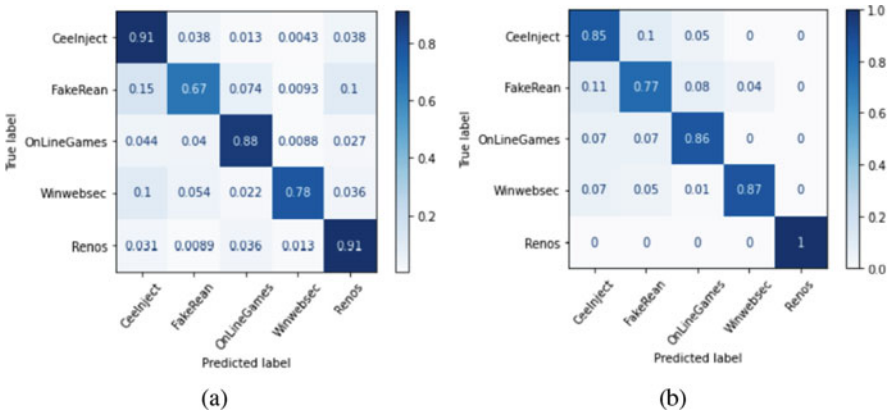

Fig. 14 Confusion matrix of MLP for (**a**) BERT features (**b**) Word2Vec features

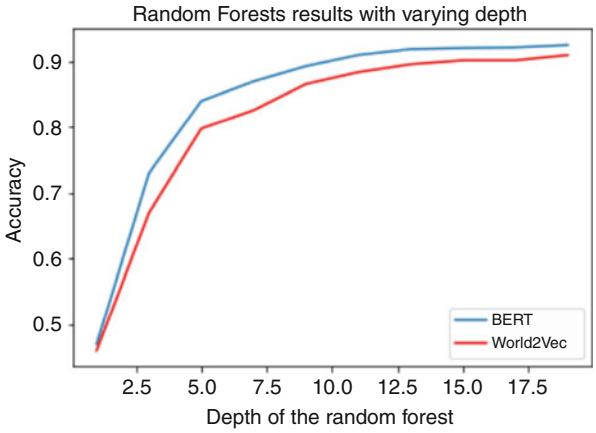

Fig. 15 Depth vs accuracy for RF using BERT and Word2Vec

accuracy was high, while the model performed insufficiently when tested on the test data. Figure 15 shows that the embeddings generated from both BERT and Word2Vec demonstrate improvement in the classification accuracy when the depth of the Random Forest was increased. The accuracy plateaus at depth 10 and gradually increases beyond this point. After further analysis, it was observed that such behavior was similar when both the depth and number of trees of the Random Forest were increased. It was also observed that a larger number of trees in the Random Forest classification model compensate for shallow depths. As seen in Fig. 16, the accuracy of the Random Forest model was high even when the depth of the decision trees was around 2.5. Beyond a depth of 2.5 there was a gradual increase in classification accuracy as the number of trees of the Random Forest classifier was increased. As described in Sect. 3.2.3, a larger number of decision trees can

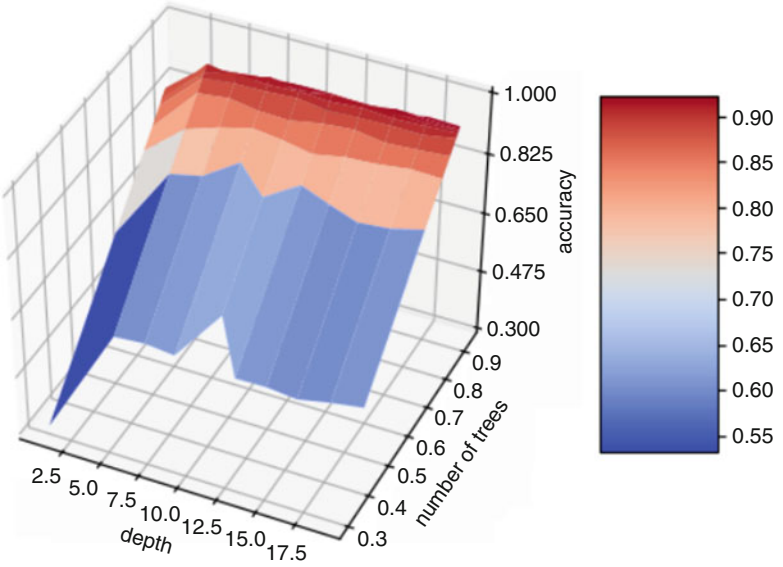

Fig. 16 Accuracy for depth vs number of tress in random forest

generalize to the training data. A class is chosen for the input data only when a majority of the decision trees generate the same classification, which protects the classification result from errors caused by the individual decision trees. This is in line with the results obtained as a part of the conducted experiments.

4.9 Summary

Word embeddings were generated by BERT and Word2Vec and they were classified using classifiers such as Logistic Regression, SVM, MLP, and Random Forests. Classification of malware samples by using word embedding generated with BERT performs better overall in comparison to Word2Vec, as shown in the Fig. 17 that summarizes the results. SVM, MLP, and Random Forests perform better overall in comparison to Logistic Regression, which is an expected outcome. MLP and SVM perform similarly as they try to find the decision boundary that best fits the data without overfitting it. Random Forests use an ensemble of decision trees to carry out the classification of the dataset and obtain a high classification accuracy. The results of this experiment prove that word embeddings generated by BERT can be used to accomplish multi-class malware classification of the dataset. When compared to Word2Vec, we see that is obtains better results. This is likely due to the attention mechanism that characterizes the BERT algorithm. In fact, by implementing the attention mechanism, BERT is able to maintain long-term dependencies as easily

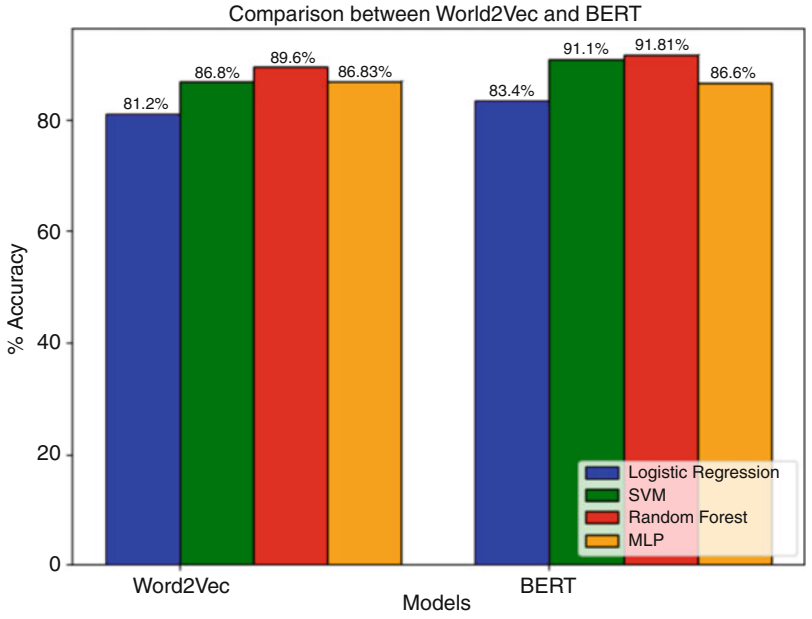

Fig. 17 Accuracies obtained after classification

as short-term dependencies. The number of opcodes selected per malware sample was 400, since BERT required a maximum of 512 words per sentence. Furthermore, while some of the malware samples had over 105 unique opcodes, the classification accuracy was not impacted, even when there was a smaller number of opcodes for some samples. This proves that word embeddings generated by BERT can capture rich features with even a limited subset of the opcodes for each input file.

5 Conclusions and Future Work

Based on the experiments conducted in this paper, it was observed that when the malware samples were mapped to word embeddings by capturing, grouping, and enriching the key components of the input features, it led to an improvement in classification accuracy while achieving malware classification. The promising results show that BERT was able to capture information that helps the classifier improving the classification accuracy. The results were superior to the ones obtained using Word2Vec on the same set of input parameters and the same set of classifiers. It proves that a transformer-based model such as BERT has applications beyond NLP. For future work, more research can be conducted in this area by using different versions of BERT. DistilBERT was used in these paper but further research can be accomplished using the other available versions of BERT. Morover, the BERT

model was pre-trained on natural language input, but it may be able to generate richer features if it is trained on malware samples directly. Finally, research can be implemented using more malware families with more complex sets of data and observing how BERT captures the key information across multiple malware families.

References

1. Yara Awad, Mohamed Nassar, and Haidar Safa. Modeling malware as a language. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, 2018.
2. P. Baldi and Y. Chavin. Smooth on-line learning algorithms for hidden markov models. *Neural Computation*, 6:307–318, 1994.
3. S. Banerjee. Word2vec — a baby step in deep learning but a giant leap towards natural language processing. https://laptrinhx.com/word2vec-a-baby-step-in-deep-learning-but-a-giant-leap-towards-natural-language-processing-3998188269/, 2018.
4. S. Basole, F. Di Troia, and M. Stamp. Multifamily malware models. *Journal of Computer Virology and Hacking Techniques*, 16:79–92, 2020.
5. D. Bilar. Opcodes as predictor for malware. *Int. J. Electron. Secur. Digit. Forensic*, 1(2):156–168, January 2007.
6. L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
7. Aniket Chandak, Wendy Lee, and Mark Stamp. A comparison of word2vec, hmm2vec, and pca2vec for malware classification. https://arxiv.org/abs/2103.05763, 2021.
8. K. Clark, U. Khandelwal, O. Levy, and C. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics.
9. HuggingFace. Distilbert. https://huggingface.co/transformers/model_doc/distilbert.html.
10. Microsoft Security Intelligence. Renos. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=TrojanDownloader:Win32/Renos&threatId=16054, 2006.
11. Microsoft Security Intelligence. Ceeinject. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=VirTool%3AWin32%2FCeeInject, 2007.
12. Microsoft Security Intelligence. Onlinegames. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=PWS%3AWin32%2FOnLineGames, 2008.
13. Microsoft Security Intelligence. Winwebsec. https://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Win32%2fWinwebsec, 2010.
14. Microsoft Security Intelligence. Fakerean. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Win32/FakeRean, 2011.
15. Samuel Kim. Pe header analysis for malware detection. Master's thesis, San Jose State University, Department of Computer Science, 2018.
16. C. McCormick. Word2vec tutorial - the skip-gram model. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model, 2016.
17. W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
18. C. Mihai and J. Somesh. Testing malware detectors. In *Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA '04, page 34–44, New York, NY, USA, 2004. Association for Computing Machinery.
19. Fred C. Pampel. *Logistic Regression: A Primer*. SAGE Publications, Inc., 2000.
20. H. Ramchoun, M. A. J. Idrissi, Y. Ghanou, and M. Ettaouil. Multilayer perceptron: Architecture optimization and training. *Int. J. Interact. Multim. Artif. Intell.*, 4(1):26–30, 2016.

21. N. Ranjan, K. Mundada, K. Phaltane, and S. Ahmad. A survey on techniques in nlp. *International Journal of Computer Applications*, 134(8):6–9, 2016.
22. sklearn. Gridsearchcv. https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.
23. SophosLabs. Sophos 2021 threat report. https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophos-2021-threat-report.pdf, 2021.
24. Mark Stamp. *Introduction to Machine Learning with Applications in Information Security*. Chapman and Hall/CRC, 2020.
25. Symantec. Internet security threat report: Malware. https://interactive.symantec.com/istr24-web, 2019.
26. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. https://arxiv.org/abs/1706.03762, 2017.
27. S. Vemparala, F. Di Troia, C. Visaggio, T. Austin, and M. Stamp. Malware detection using dynamic birthmarks. In *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics*, IWSPA '16, page 41–46, New York, NY, USA, 2016. Association for Computing Machinery.
28. P. Vinod, R. Jaipur, R. Laxmi, and M. Gaur. Survey on malware detection methods. In *Proceedings of the 3rd Hackers Workshop on Computer and Internet Security*, pages 74–79, 2009.
29. M. Wadkar, F. Di Troia, and M. Stamp. Detecting malware evolution using support vector machines. *Expert Systems with Applications*, 143:113022, 2020.
30. W. Wong and M. Stamp. Hunting for metamorphic engines. *Journal of Computer Virology and Hacking Techniques*, 2:211–229, 2017.