Static Analysis for Malware Detection with Tensorflow and GPU

Jueun Jeon, Juho Kim, Sunyong Jeon, Sungmin Lee, and Young-Sik Jeong

Abstract With the advent of malware generation toolkits that automatically generate malware, anyone without a professional skill can easily generate malware. As a result, the number of new/modified malware samples is rapidly increasing. The malware generated in this way attacks vulnerabilities, such as PCs and mobile devices without security patch, causing damages involving malicious actions, such as personal information leakage, theft of authorized certificates, and cryptocurrency mining. To solve this problem, most security companies use the signature-based malware detection technique to detect malware, in which the signatures of known malware and files suspected to be malware are compared before detecting malware. However, the signature-based malware detection technique has a limitation in that it is not efficient for detecting new/modified malware which is generated rapidly. Recently, research is underway to utilize deep learning technology for detecting new/modified malware. In this study, we propose a SAT scheme that can detect not only known malware but also new/modified malware more quickly and accurately, thereby reducing malware-induced damages to PCs and mobile devices. The SAT scheme employs an open source library called Tensorflow in the GPU environment to learn malware signatures and then to statically analyze malware.

Keywords Malware analysis · Malware detection · Static analysis · Deep learning · Signature

J. Jeon · J. Kim · S. Jeon · S. Lee · Y.-S. Jeong (✉)
Department of Multimedia Engineering, Dongguk University, Seoul, Republic of Korea
e-mail: ysjeong@dongguk.edu

J. Jeon
e-mail: jry02107@dongguk.edu

J. Kim
e-mail: 2015112624@dongguk.edu

S. Jeon
e-mail: sunyongj1004@dongguk.edu

S. Lee
e-mail: bearbear11@dongguk.edu

J. J. Park et al. (eds.), *Advances in Computer Science and Ubiquitous Computing*,
Lecture Notes in Electrical Engineering 715,
https://doi.org/10.1007/978-981-15-9343-7_76

1 Introduction

About two billion malware attacks occurred in 2018 alone. Since the introduction of automated malware toolkits, about 340,000 new types of malware are detected every day. For rapidly growing new/modified malware, the spreading method to other PCs and mobile devices as well as the symptoms of infected devices are gradually becoming more complicated and intelligent. As a result, PCs and mobile devices infected with such malware experience various hacking-related damages, such as personal and confidential information leakage, cryptocurrency mining, and spam mailing. In the case of recently detected Vidar malware, it is installed in the device by taking advantage of the vulnerability of the Internet Explorer browser where security patch for the vulnerability was not applied, and then infects and spreads the malware by exploiting normal advertisement services [1–5].

In order to protect the user's PCs and mobile devices from such malware, a variety of techniques for analyzing malware have been investigated. Malware analyzing techniques can be divided into the following three types as shown in Fig. 1. First, the signature-based malware detection technique detects malware by storing signatures of previously detected malware in a database and then comparing them with signatures of files suspected to be malware. In the heuristics-based malware detection technique, which is also called behavior-based malware detection technique, if a certain degree of match is found between specific parts of previously detected malware and a file suspected to be malware, the file is determined to be malware. The specification-based malware detection technique is one type of the heuristics-based malware detection technique. The specification-based malware detection technique is not a method for analyzing the signature of malware, but a method for detecting malware by detecting deviations between the program specification and the

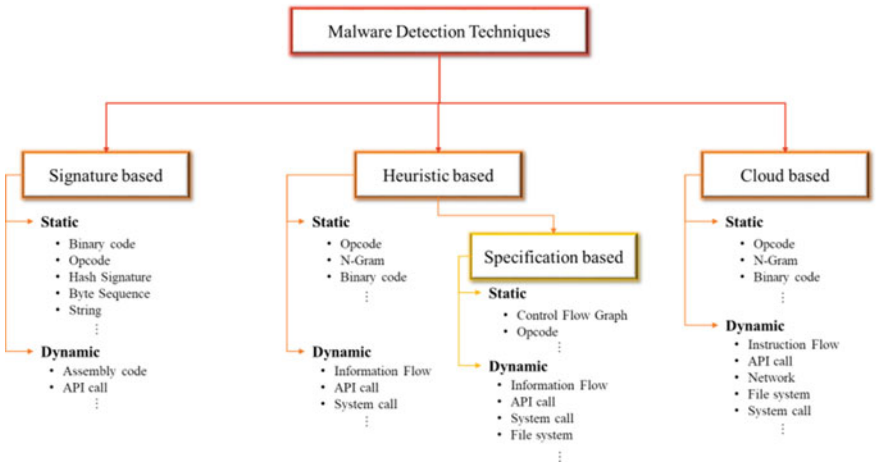

Fig. 1 Various techniques for identifying and detecting malware

program behavior. Lastly, the cloud-based malware detection technique is a method for detecting malware by transferring the file suspected to be malware to a cloud server to analyze the signatures of the malware within the cloud server, and then sending the malware detection result to the client. Most security companies rely on the signature-based malware detection technique to detect malware which attacks and damages PCs and mobile devices and to analyze and identify malware [1–3, 6, 7].

However, while this signature-based malware detection technique can detect known malware accurately, its detection result may not be as accurate for new/modified malware, where part of the malware has been modified or packaged. To solve this problem, many studies have been conducted to apply deep learning to the signature-based malware detection technique to detect malware [1, 3–6, 8].

In an effort to keep the alert level high against the threats of both well-known existing malware and new/modified malware, in this paper, we propose the static analysis for malware detection with Tensorflow (SAT) scheme, which can detect malware quickly and thus prevent it from spreading to other PCs and mobile devices. This SAT scheme employs the Long Short Term Memory (LSTM) model through the Tensorflow library to perform a static analysis of known malware and new/modified malware using the signature-based malware detection technique [9]. The SAT scheme proposed in this study is intended to show the most efficient overall performance with both the speed and accuracy of malware detection being equally considered.

2 Related Works

In order to detect new/modified malware as well as known malware, various studies have been conducted to statically analyze malware using deep learning. Static analysis is defined as a method of analyzing the code and binary file information of malware without directly executing it. In static analysis, opcodes are mainly used as the key signatures for detecting malware. In terms of the opcode, although the opcode itself is important, the sequence between the opcodes is also considered important. For this reason, many studies have been conducted to detect malware by utilizing the LSTM model, in which the learning process is based on the sequence of text strings [10].

As a method to detect IoT malware that threatens to compromise IoT devices used in diverse industries, HaddadPajouh et al. [3] proposed a strategy to build a detection model by extracting the opcodes from decompiled malware and then having them learned by the LSTM. To train the detection model, they used an IoT application data set consisting of 281 malware and 270 non-malware files. In addition, they constructed three different LSTM models to evaluate the detection models which had been trained based on the data set of 100 new malware files. They found that the LSTM model consisting of two hidden layers showed the highest accuracy in detecting new malware compared to the other LSTM models.

Kang et al. [4] created a vector with 1369 dimensions using the one-hot encoding method to classify malware files according to their types and features. After reducing

Table 1 Comparison between the SAT scheme and previous studies

Related works	Feature	Number of branches classified	Performance evaluation factors considered	Target environment
A deep recurrent neural network based approach for internet of things malware threat hunting	Opcode	2 (Benign, Malware)	Accuracy	ARM based IoT devices
Long short-term memory-based malware classification method for information security	Opcode, API call	9 (Malware family)	Accuracy	Window
Our proposed scheme	Opcode, API call	9 (Malware family)	Execution time, accuracy	Window

the number of dimensions from 1369 to 300 using CBoW, which is one of the word2vec technique, they proposed a model that trains an LSTM model consisting of two hidden layers with 128 dimensions to detect malware. For the training of the malware detection model, they used the malware data set published by Microsoft in the Microsoft Malware Classification Challenge (BIG 2015), which is composed of opcodes and API calls extracted through the static analysis of the file contents and characteristics of malware [11]. The performance assessment on the opcode-embedding method of the proposed model led to the conclusion that malware can be detected more quickly and accurately when CBOW, one of the word2vec models, was used compared to the one-hot encoding method.

Table 1 shows the comparison between the SAT scheme proposed in this paper and previous studies.

3 Scheme of SAT

In order to establish a malware detection model with efficient performance in terms of the speed and accuracy of detection and classification during the detection and classification process of malware with various characteristics, in this paper, we propose the SAT scheme that performs static analysis of malware in the GPU environment utilizing the Tensorflow library.

The SAT scheme consists of three phases: data processing, pre-processing, and learning phases, as shown in Fig. 2, where malware learning and detection are processed based on this schematic. In the first data processing phase, for the training of

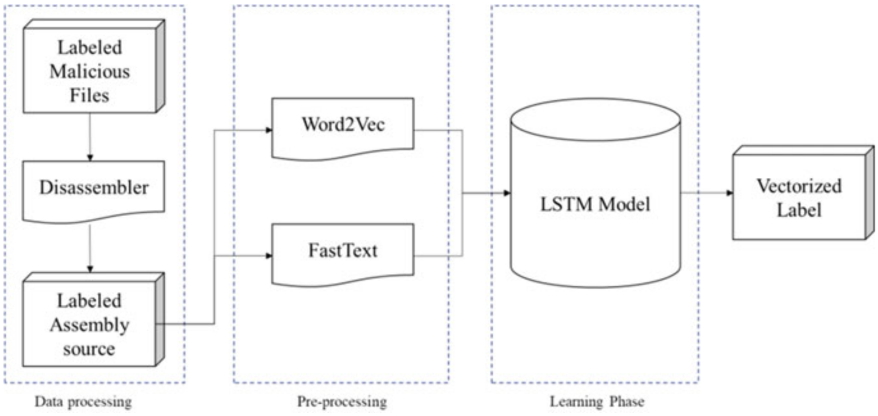

Fig. 2 Overall schematic of the SAT scheme for malware detection and classification

malware signatures, known malware files are labeled and statically analyzed through a disassembler to extract their opcodes and API calls. In the pre-processing phase, word embedding is performed using word2vec or fasttext to apply the extracted opcode sequence data to the LSTM model. In the last learning phase, the malware is classified according to the characteristics of the malware based on the signatures of the vectorized malware, which are opcodes and API calls, followed by malware learning and detection.

3.1 Data Processing

In the data processing phase, the malware, which has been classified into one of the nine kinds of malware, is labeled to extract its opcodes and API calls, which are the signatures of the malware, and through a disassembler process, the malware is converted from a machine code form to an assembly language code form to statically analyze the file information and the code contents of the malware.

3.2 Pre-processing

In the pre-processing phase, a word embedding method, through which opcodes and API calls in the form of natural language are converted to numbers for computers to understand and efficiently process them, is applied so that the learning process is carried out in the LSTM model based on the opcodes and API calls extracted in the static analysis during the previous data processing phase. The SAT scheme proposed in this paper employs a word embedding method of word2vec or fasttext, whichever shows optimal performance, in the pre-processing phase. The word2vec technique was developed by Google in 2013 and is one of the techniques for vectorizing key words by analyzing surrounding assertions [12]. Word2vec consists of the CBoW technique that predicts words based on context and the Skip-gram technique that predicts context based on one word. Fasttext is a technique developed by Facebook that considers many subwords to exist in a single word and vectorizes the word in consideration of the subwords [13].

3.3 Learning Phase

Lastly, the learning phase, where malware is detected by learning and classifying the signatures of malware according to their characteristics, is composed as shown in Fig. 3. The opcodes and API calls vectorized through the pre-processing process are assigned to two LSTM layers composed of 128 cells, which are hidden layers, in order to classify them into one of the nine kinds of malware which have been classified by the characteristics of malware. Here, the nine kinds of malware includes Ramnitm, Lollipop, Kelihos_ver3, Vundo, Simda, Tracur, Kelihos_ver1, Obfuscator.ACY, and Gatack. Malware is learned through Softmax layer and Adam Optimizer, and the new malware is tested based on the model established in this way.

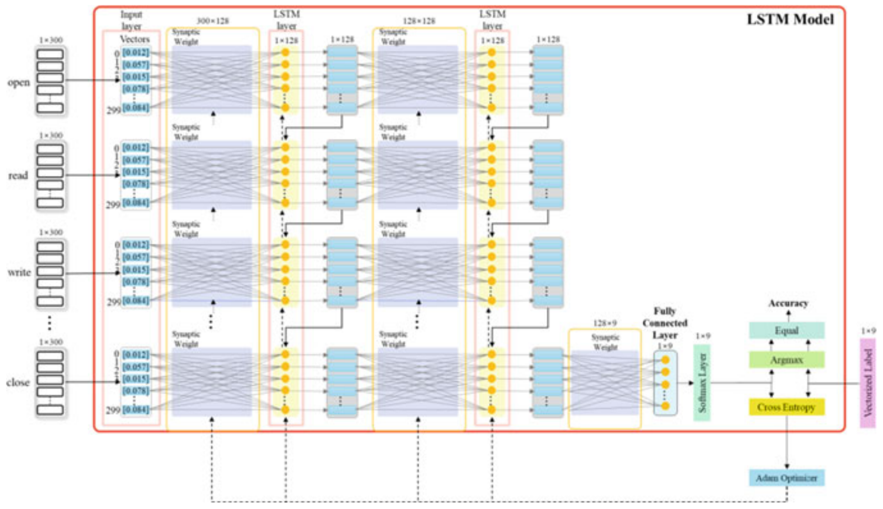

Fig. 3 The learning phase in the SAT scheme for detecting and classifying malware

4 Experience

The SAT scheme proposed in this paper was tested under the environments with the configurations of CPU with eight AMD FX-8370E Eight-Core Processors and of GPU of Quadro P4000 with 32.9 GB memory using the data set published by Microsoft in BIG 2015 [11]. The main purpose of the experiment conducted for the SAT scheme is to determine whether to use word2vec or fasttext as the word embedding method in the static analysis of malware to obtain the most optimized results for the SAT scheme. The input data size, window size, hidden layer number, and cell number were used as parameters for both word2vec and fasttext. Additionally, the embedding method was used as a parameter for word2vec for analysis, while the n-gram range was used as a parameter for fasttext for analysis. Input data refers to the data to be entered in the word-embedding method after extracting API calls and opcodes from the assembly code and window refers to the range of data analysis in the word-embedding method. In addition, cell refers to the size of the hidden layer and the hidden layer refers to the step of input data processing within the LSTM model. The initial values used in the experiment were as follows: 300 for the input data, 5 for the window size, 128 for the number of cells, and 2 for the hidden layer.

Tables 2 and 3 show the accuracy of malware detection when the word2vec model or the fasttext model was used as the word embedding method. The accuracy of malware detection under the various conditions ranged from 96.61 to 97.60%. In addition, a higher accuracy of 97.60% was observed in detecting malware when fasttext was used compared to when word2vec was used.

The word2vec technique showed a malware detection accuracy of 97.59%, and the fasttext technique showed a malware detection accuracy of

Table 2 Detection accuracy when using the word2vec model as the word embedding method

Type	Input data	Window size	Cell	Hidden layer	Embedding method	Accuracy (%)
Word2vec	200	5	128	2	CBoW	97.57
	300	5	128	2	CBoW	97.59
	400	5	128	2	CBoW	97.59
	300	3	128	2	CBoW	97.59
	300	5	128	2	CBoW	97.59
	300	7	128	2	CBoW	97.59
	300	5	64	2	CBoW	97.54
	300	5	128	2	CBoW	97.59
	300	5	256	2	CBoW	97.59
	300	5	128	2	CBoW	97.59
	300	5	128	3	CBoW	97.19
	300	5	128	2	CBoW	97.59
	300	5	128	2	Skip-gram	97.59

Table 3 Detection accuracy when using the fasttext model as the word embedding method

Type	Input data	Window size	Cell	Hidden layer	N-gram	Accuracy (%)
Fasttext	100	5	128	2	3–6	97.59
	200	5	128	2	3–6	97.59
	300	5	128	2	3–6	97.60
	300	3	128	2	3–6	97.60
	300	5	128	2	3–6	97.60
	300	7	128	2	3–6	97.60
	300	5	64	2	3–6	97.51
	300	5	128	2	3–6	97.60
	300	5	256	2	3–6	97.59
	300	5	128	2	3–6	97.59
	300	5	128	3	3–6	96.61
	300	5	128	2	2–5	96.60
	300	5	128	2	3–6	97.60
	300	5	128	2	4–7	97.60

Table 4 Execution time when word2vec or fasttext was used in the pre-processing phase

Word2vec	12 min
Fasttext	41 min

Table 5 Execution time under the CPU and GPU environments in the learning phase

		Device	
		CPU	GPU
Word embedding	Word2Vec	42 h 20 m 30 s	8 h 12 m 23 s
	Fasttext	43 h 45 m 10 s	8 h 16 m 2 s

97.60%. Tables 4 and 5 show the analysis results of the execution time when word2vec and fasttext were executed under the optimal conditions in the pre-processing and learning phases. Table 4 shows the execution time when word2vec and fasttext were used as the word embedding method in the pre-processing phase. Table 5 shows the execution time when word2vec and fasttext were executed in the CPU and GPU environments in the learning phase.

5 Conclusion

With the advent of toolkits that can automatically generate malware, anyone can create new/modified malware without being an expert. As a result, the number of malware samples is rapidly increasing, and security companies spend a lot of time in analyzing the characteristics of new/modified malware and generating signatures of malware to help detect malware. To solve this problem, various malware detection techniques have emerged, and many relevant studies have been conducted. However, catching up with the generation rate of new/modified malware is impossible, which makes it difficult to analyze and detect the characteristics of new/modified malware. For this reason, deep learning has been applied to help detect malware, and the related research has begun.

In static analysis, opcodes and API calls are considered as the signatures of malware, which are the feature elements of malware. For these opcodes, a single word itself is not meaningful, but the surrounding words are rather meaningful. Therefore, the LSTM model has been mainly used to analyze malware using opcodes. However, previous studies utilizing the LSTM model focused solely on the accuracy of malware detection.

Therefore, in this paper, we proposed the SAT scheme that statically analyzes malware in the GPU environment to detect malware quickly and accurately using the LSTM model. The SAT scheme consists of three phases: data processing, pre-processing, and learning phases. The results of the experiments on the malware detection time and accuracy, which were performed based on the SAT scheme, indicated that fasttext was more efficient in terms of accuracy than word2vec, but when fasttext was used in the pre-processing phase, an inefficient execution time was observed. However, when the learning phase, which is the time when malware is classified and learned in the LSTM model, was compared, the execution time under the GPU environment was about three times more efficient than that under the CPU environment. This suggests that when fasttext is used, the problem of inefficient execution time in the pre-processing phase can be compensated.

Acknowledgements This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-00644, Linux Malware Dynamic Detection & Protection Solution on Embedded Device).

References

1. Souri A, Hosseini R (2018) A state-of-the-art survey of malware detection approaches using data mining techniques. Human-Centric Comput Inf Sci 8:1–22
2. Keegan N, Ji S-Y, Chaudhary A, Concolato C, Yu B, Jeong DH (2016) A survey of cloud-based network intrusion detection analysis. Human-Centric Comput Inf Sci 6:1–16
3. HaddadPajouh H, Dehghantanha A, Khayami R, Choo K-KR (2018) A deep recurrent neural network based approach for internet of things malware threat hunting. Future Generat Comput Syst 85:88–96

4. Kang J, Jang S, Li S, Jeong Y-S, Sung Y (2019) Long short-term memory-based Malware classification method for information security. Comput Electr Eng 77:366–375
5. Choi S-Y, Lim CG, Kim Y-M (2019) Automated link tracing for classification of malicious websites in malware distribution networks. J Inf Process Syst 15:100–115
6. Daoud WB, Obaidat MS, Meddeb-Makhlouf A, Zarai F, Hsiao K-F (2019) TACRM: trust access control and resource management mechanism in fog computing. Human-Centric Comput Inf Sci 9:1–18
7. Belaoued M, Mazouzi S (2016) A Chi-square-based decision for real-time malware detection using PE-file features. J Inf Process Syst 12:644–660
8. Nagpal B, Chauhan N, Singh N (2017) A survey on the detection of SQL injection attacks and their countermeasures. J Inf Process Syst 13:689–702
9. Tensorflow. https://www.tensorflow.org/?hl=ko
10. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neu Computat 9:1735–1780
11. Microsoft Malware Classification Challenge (BIG 2015). https://www.kaggle.com/c/malware-classification
12. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: ICLR 2013, International conference on learning representations. Conference Track Proceedings, Arizona, pp 1–12
13. Mikolov T, Grave E, Bojanowski P, Puhrsch C, Joulin A (2017) Advances in pre-training distributed word representations. In: The Eleventh international conference on language resources and evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, pp 52–55