Human-centric Computing and Information Sciences

June 2021 | Volume 11

www.hcisjournal.com

KIPS Korea Information Processing Society

KIPS CSWRG Korea Information Processing Society Computer Software Research Group

HCIS | Human-centric Computing and Information Sciences

RESEARCH(Research Manuscript) Open Access

Human-centric Computing and Information Sciences (2021) 11:27
DOI: https://doi.org/10.22967/HCIS.2021.11.027
Received: November 28, 2020; Accepted: June 9, 2021; Published: June 30, 2021

Two-Stage Hybrid Malware Detection Using Deep Learning

Seungyeon Baek, Jueun Jeon, Byeonghui Jeong, and Young-Sik Jeong*

Abstract

With the increasing number and variety of Internet of Things (IoT) devices supporting a wide range of services such as smart homes, smart transportation, and smart factories in smart cities, malware carrying various cybersecurity threats are rapidly increasing in terms of type and number. To protect IoT devices from cyberattacks, studies on malware detection using artificial intelligence are being conducted. However, with the emergence of IoT malware and their various evasion techniques, the probability of falsely detecting malware as benign is also increasing. In this study, we propose a two-stage hybrid malware detection (2-MaD) scheme for the protection of IoT devices from obfuscated malware in a smart city setting. The 2-MaD consists of two stages of IoT malware detection. First, after performing static analysis, the opcode is extracted, and using the learned information through a bidirectional long short-term memory model, benign files are detected. In the next stage, a dynamic analysis is performed on files classified as benign in a nested virtual environment. After extracting information on behavior and process memory from the behavior log based on system changes, malware can be detected through the trained EfficientNet-B3 model.

Keywords

Malware Detection, Hybrid Analysis, Internet of Things, Deep Learning, Bi-LSTM, CNN

1. Introduction

To address urban problems—such as disasters, crime, and traffic congestion—smart cities have fused a variety of industries and services—such as public services, transportation, and energy—to provide integrated services. Many different smart Internet of Things (IoT) devices are important components of these smart cities and the integrated services they offer. IoT devices provide useful data by collecting and storing a variety of information generated within a smart city [1, 2]. As the number of IoT devices connected to the network increases, cyberattacks against smart cities have also increased in scale and complexity [3–6]. When IoT devices are infected with malware due to distributed denial of service attacks, not only is sensitive information leaked, but the smart city is also subjected to large-scale cybersecurity threats [7, 8].

In traditional methods, malicious attacks are detected by monitoring a small number of IoT devices

***Corresponding Author:** Young-Sik Jeong (ysjeong@dongguk.edu)
Department of Multimedia Engineering, Dongguk University, Seoul, Korea

through a security operation center (SOC) [9, 10]. However, it is difficult to protect the myriads of IoT devices from advanced cyberattacks using conventional security methods [11, 12]. To address such limitations, various studies have been conducted on the automatic detection of malware using artificial intelligence (AI) techniques [13–15].

However, malware authors evade these detection techniques by rapidly generating various types of malware using code injection and packing techniques [16, 17]. With an alarming rate of increase in obfuscated malware—such as malware with behavior similar to that of a benign file—accurate detection of malware has become more challenging [18, 19].

Consequently, this study proposes a two-stage hybrid malware detection (2-MaD) scheme in which files classified as benign in Stage 1 are re-examined in Stage 2 to achieve a low false negative rate (FNR). After performing static analysis on the overall file structure in Stage 1, the opcode is extracted, and benign files are detected using the bidirectional long short-term memory (Bi-LSTM) model. Dynamic analysis is performed in Stage 2 on the datasets classified as benign in Stage 1, and various features are extracted from the monitored behavior. After transforming the extracted process memory and behavior features into a three-dimensional tensor, malware is detected using the EfficientNet-B3 model.

The 2-MaD aims to increase the accuracy of malware detection by reducing the malware false detection rate. In addition, to improve the learning and detection performance of the EfficientNet-B3 model, various behavioral features are extracted and utilized from the system log generated through dynamic analysis. The proposed 2-MaD scheme protects various IoT devices in a smart city from malicious threats.

The remainder of this paper is organized as follows. Section 2 reviews previous studies related to malware detection. Section 3 describes the proposed 2-MaD scheme, whereas Section 4 discusses the implementation of the 2-MaD model. Section 5 presents the performance evaluation results of the 2-MaD model.

2. Related Work

Various studies have been conducted to protect devices connected to the Internet from the intrusion of malware. Research on static analysis—which is a representative malware analysis technique—identifies malware patterns without executing files and source code [20–24]. However, this static analysis is unable to detect obfuscated malware or malware with unknown patterns. Conversely, malware detection techniques using dynamic analysis can detect obfuscated or unknown malware by analyzing real-time behavior via recording system changes through the actual execution of the files [25–29]. Hybrid analysis, another malware analysis technique, can accurately detect malware by performing both static and dynamic analyses; this method is used in a wide range of cybersecurity studies [30–33].

In this section, we describe representative previous studies that analyze features of malware through static analysis, dynamic analysis, and hybrid analysis, and perform automatic malware detection using AI techniques.

2.1 Static Malware Detection

Zhang et al. [23] proposed a deep learning-based malware detection framework for the classification of eight types of ransomware. To reduce the dimensionality of the opcode sequences used as features, term frequency-inverse document frequency (TF-IDF) was applied to perform filtering. After opcode sequence was divided into several patches, a self-attention convolutional neural network (SA-CNN) model was trained using these patches. Finally, to analyze the relationship between opcode sequences and classify the ransomware, a directional self-attention network (Di-SAN) model was used.

Santos et al. [24] proposed a malware detection method based on the frequency of the opcode sequence. Noise was removed by assigning weights based on the dependency between the opcode sequences measured using mutual information techniques. The calculated opcode sequence frequency vectors were

used for training data mining algorithms such as the decision tree (DT), support vector machine (SVM), k-nearest neighbor (KNN), and Bayesian network for malware detection.

2.2 Dynamic Malware Detection

Devesa et al. [28] proposed an automated malware detection system based on emulation and simulation for malware behavior analysis. In their proposed system, an application programming interface (API) calls for the registry; files and memory were extracted through dynamic analysis in the sandbox, then transformed to binary vectors, after which malware detection was performed using naïve Bayes (NB), random forest (RF), J48, and sequential minimal optimization (SMO) techniques.

Jeon et al. [29] proposed a decreasing additive-increase/multiplicative-decrease (DAIMD) model for the detection of new and variant IoT malware. By performing dynamic analysis in a cloud-based nested virtual environment, behaviors such as network, process, memory, and virtual file system behaviors were extracted as features and converted into images. The generated images were used to train a CNN model for the detection of IoT malware.

2.3 Hybrid Malware Detection

Darabian et al. [32] proposed a method for detecting cryptomining malware through hybrid analysis. Using opcode extracted from static analysis and system calls generated from dynamic analysis, LSTM, attention-based LSTM (ATT-LSTM), and CNN models—which are deep learning techniques—were trained for the detection of cryptomining malware.

Santos et al. [33] proposed using OPEM for unknown malware detection based on a hybrid approach. OPEM extracted opcode sequences of a fixed length through static analysis and then generated frequency vectors. In addition, execution traces—such as operation and system calls extracted through dynamic analysis—were transformed into binary vectors. KNN, DT, SVM, NB, and Bayesian networks were trained using these vectors for malware detection.

The 2-MaD scheme proposed in this study performs dynamic analysis on files classified as benign through malware detection based on static analysis to perform the final malware detection. By performing hybrid analysis in two stages, the detection rate of IoT malware that is incorrectly classified as benign significantly increases.

Table 1. Comparison between conventional malware detection methods and the proposed 2-MaD scheme

Related works	Analysis technology	Feature	AI technology
Zhang et al. [23]	Static analysis	Opcode	SA-CNN, Di-SAN
Santos et al. [24]	Static analysis	Opcode	DT, SVM, KNN, Bayesian network
Devesa et al. [28]	Dynamic analysis	API calls	NB, RF, J48, SMO
Jeon et al. [29]	Dynamic analysis	Network, process, virtual file system, memory, system calls	ZFNet
Darabian et al. [32]	Hybrid analysis	Opcode, system call events	LSTM, ATT-LSTM, CNN
Santos et al. [33]	Hybrid analysis	Opcode, execution trace	KNN, DT, SVM, NB, Bayesian network
2-MaD	Hybrid analysis	Opcode, API calls, process memory	Bi-LSTM, EfficientNet-B3

Table 1 compares the conventional methods of malware detection using AI techniques with the 2-MaD scheme proposed in this study. The malware analysis technology used to understand the structure and characteristics of malware, the features representing malware behavior, and the AI techniques used to classify malware and benign files are presented for comparison.

3. 2-MaD Scheme

This study proposes a 2-MaD scheme that detects IoT malware with high accuracy by performing malware detection based on hybrid analysis to lower the probability of misclassification of malware as benign. Fig. 1 shows the overall 2-MaD scheme.

The 2-MaD scheme is divided into Stage 1, in which malware detection is performed using static analysis, and Stage 2, in which malware detection is performed using dynamic analysis. The datasets used for IoT malware detection in Stage 2 are composed of files classified as benign in Stage 1.

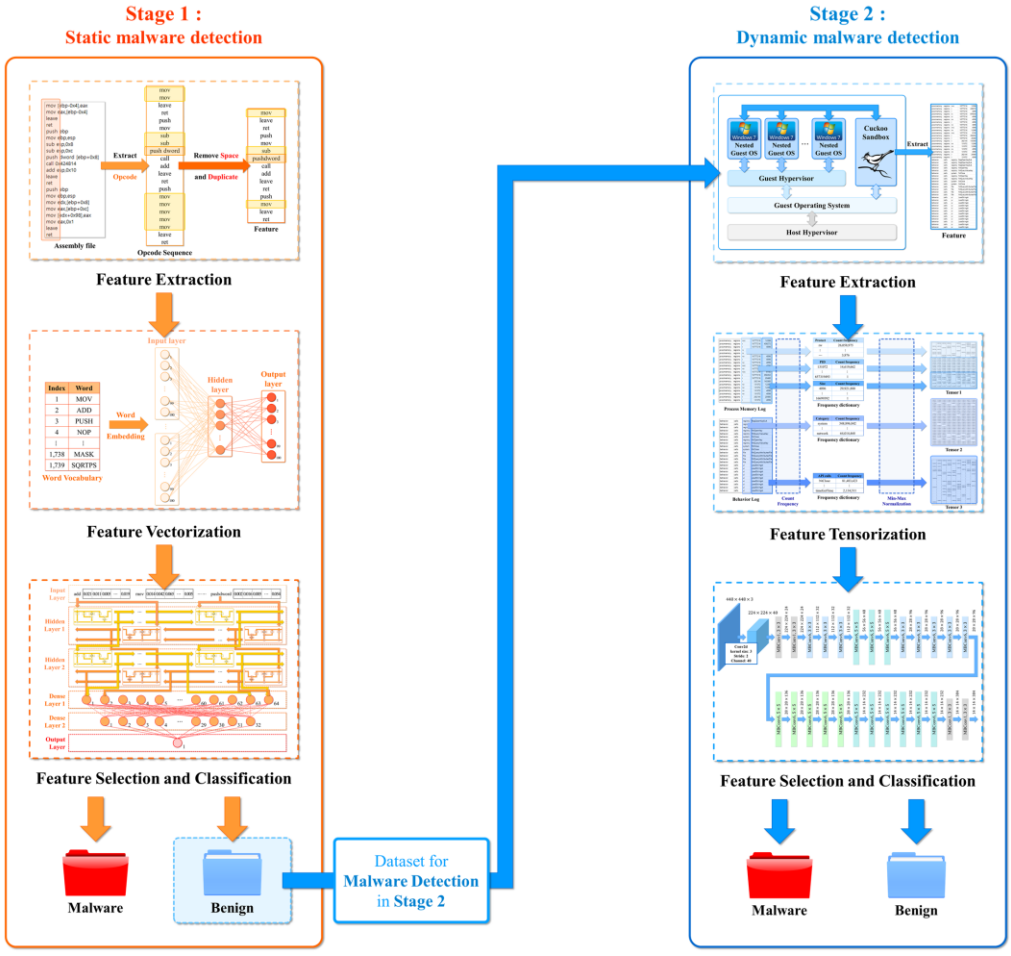

Fig. 1. Two-stage hybrid malware detection (2-MaD) scheme.

3.1 Stage 1: Static Malware Detection

Stage 1 consists of feature extraction, feature vectorization, feature selection, and classification for analysis and training of the overall structure of the file.

3.1.1 Feature extraction

In feature extraction, the opcode sequences representing the operation are extracted by performing a disassembly of the files. The generated opcode sequences are used as features to distinguish benign files from malware.

3.1.2 Feature vectorization

In feature vectorization, features composed of a string are transformed into a vector through an embedding process to be used in the training of a deep learning model. When the one hot encoding method—an embedding technique—is used, the dimensionality of the vector increases rapidly, causing memory overflow. Consequently, in this study, word embedding is performed using the FastText method developed by Facebook [34], and a vector with an embedding size is created for each opcode.

3.1.3 Feature selection and classification

The vectors obtained by the feature vectorization process are used for training the Bi-LSTM model to perform feature selection and classification. Various AI techniques can be used to perform the feature selection process and filter unnecessary information from the opcode sequence. However, in this study, features are selected and learned through the Bi-LSTM model in order to consider both the meaning and association of the sequence IoT malware detection. The Bi-LSTM model minimizes data loss by bilateral analysis of the sequence, thus improving the malware detection accuracy [35].

3.2 Stage 2: Dynamic Malware Detection

With a higher ratio of classifying true malware as benign in Stage 1, the probability of malicious attacks on IoT devices increases. Consequently, Stage 2 detection is performed to protect IoT devices from cyber threats. Stage 2 consists of feature extraction, feature tensorization, feature selection and classification, whereby the real-time behaviors of files executed in a virtual environment are analyzed, and training is performed using these behaviors.

3.2.1 Feature extraction

In feature extraction, a file is first executed in a Cuckoo Sandbox, after which dynamic analysis is performed to monitor system changes and record log data. To understand the primary malware behavior, logs of process memory and behavior are extracted as features from the log data. In terms of process memory, region information when a process is loaded into memory for file execution is recorded, and in terms of behavior, information from the analysis of the behavior of the file in real time through API function calls is described. Table 2 lists the types and compositions of these features.

Table 2. Types and composition of features extracted from the log of the Cuckoo Sandbox

Type	Composition	Description
Process memory	Protect	Permission for reading/writing
	Type	Process ID
	Size	Memory size
Behavior	Category	Category where the API is used
	API calls	Function used in the execution of the file

3.2.2 Feature tensorization

In feature tensorization, vectorization is performed for two types of features, which are then transformed into three-dimensional (3D) tensors. First, in order to derive a vector, the protection, type, and size of process memory and the category and frequency for each API call in behavior are calculated. To scale the frequency for each feature—currently distributed over a wide range, between 0.1 and 1—min-max normalization is applied, calculated as shown in Equation (1). In this case, assuming that the normalization is applied based on a specific item, $freq_i$ represents the i-th frequency, $freq_{min}$, and $freq_{max}$ represent the minimum and maximum values of the frequency, respectively, and V_i indicates the vector values scaled within a set range.

$$V_i = 0.9 \times \frac{freq_i - freq_{min}}{freq_{max} - freq_{min}} + 0.1 \quad (1)$$

When one-dimensional embedding is applied to transformed vectors, information loss occurs because all vectors cannot be used. Therefore, in this study, in order to use vectors of a large size as inputs to the deep learning models, a 3D tensor is created with process memory as the first tensor, and category and API calls in behavior as the second and third tensors, respectively.

3.2.3 Feature selection and classification

In feature selection and classification, representative features are selected from the 3D tensor and learned through deep learning models to detect IoT malware. To remove unnecessary features in the tensor, dimension reduction is performed through the EfficientNet-B3 model, which is a CNN technique. EfficientNet-B3 is a model that extracts meaningful features by adjusting the balance between the depth, width, and resolution of a network. The training results for the ImageNet datasets have been reported to be excellent [36].

4. 2-MaD Implementation

To detect malware that attacks IoT devices in a smart city, 2-MaD was constructed in a computing environment built on an Intel Core i7-9700K and GeForce RTX 3090 platform. To prevent device infection during the feature extraction process using dynamic analysis in Stage 2, the virtual environment was nested by setting Linux 18.04 as the guest OS and Windows 7 as the nested guest OS. In addition, only the portable executable (PE) file format from KISA-datachallenge2019-Malwares.04, provided by the Korea Internet & Security Agency (KISA), was used for the 2-MaD datasets.

4.1 Stage 1: Static Malware Detection

In Stage 1, a malware detection model based on a static analysis was used. Out of a total of 31,272 datasets, 25,016 were used as training datasets and 3,128 as validation and test datasets, respectively, to construct the Bi-LSTM model.

4.1.1 Feature extraction

For feature extraction, opcode sequences were extracted using Pydasm, a Python-based disassembler. To improve performance, the same continuous opcodes were removed, and opcodes such as *mov dword* and *push dword* were used without including spaces. In addition, files without extracted opcode were excluded from the dataset.

4.1.2 Feature vectorization

Fig. 2 shows the distribution of the opcode sequence lengths extracted for each file. The opcode sequence length of approximately 60% of all files was less than 600; files with other lengths were few in number and scattered over a wide range. As the length of the opcode sequence increases, a memory overflow occurs in the training process; thus, in this study, the maximum length of the opcode sequence was limited to 600. Word embedding was performed using a word vocabulary composed of 1,739 opcodes in total. The generated vectors had a size of 100 and were used as inputs to the Bi-LSTM model.

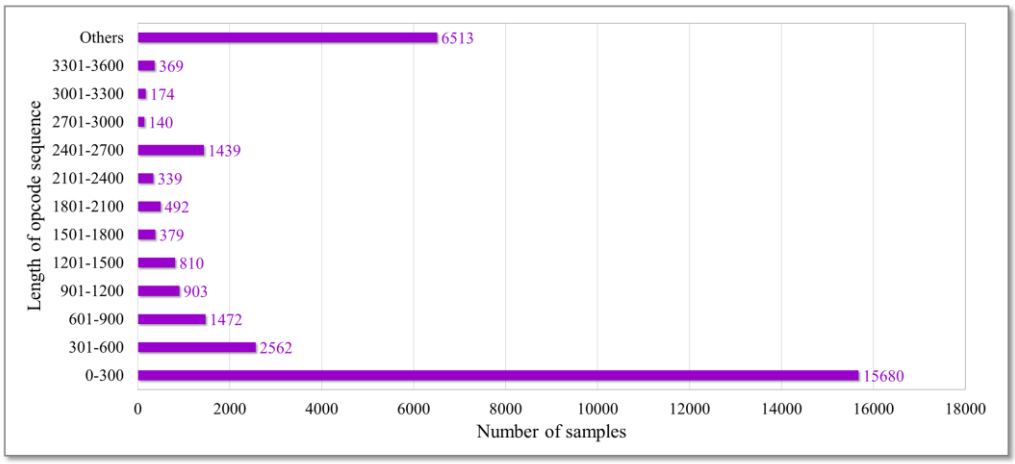

Fig. 2. Distribution of opcode sequence lengths extracted from each file.

4.1.3 Feature selection and classification

The Bi-LSTM model for feature selection and classification has the architecture outlined in Table 3, and training, validation, and tests were performed with the model.

The Adam optimizer was used for model optimization; the learning rate was set to 0.001; the batch size was set to 32; and training was performed for a total of 100 epochs. To validate the performance of the constructed model, the validation accuracy measured in epoch 15, which had the lowest validation loss value of 0.1545, was 93.80%. In addition, the test accuracy was 93.89%.

Table 3. Architecture of the Bi-LSTM model used in feature selection and classification

Layer (type)	Output shape	Number of parameters
Embedding	(None, 600, 100)	174,000
Bidirectional 1	(None, 600, 256)	234,496
Bidirectional 2	(None, 128)	164,352
Dense 1	(None, 64)	8,256
Dense 2	(None, 32)	2,080
Dense 3	(None, 1)	33

4.2 Stage 2: Dynamic Malware Detection

To perform dynamic malware detection, among 37,216 datasets, 31,310 were used as training datasets and 4,476 were used as validation datasets. In addition, a total of 1,475 files classified as benign in Stage 1 were used as test datasets to implement the EfficientNet-B3 model.

4.2.1 Feature extraction

For feature extraction, a file was executed for up to 120 seconds in a virtual environment, and then a report was created on real-time behavior using the Cuckoo Sandbox. The report was composed of various data on system changes. In this study, only the information on process memory and behavior was extracted as features.

Table 4. Calculated frequency by item in dynamic features

Type	Composition	Name	Frequency
Process memory	Protect	RW	26,058,975

Type	Composition	Name	Frequency
		R	25,629,594
		⋮	⋮
		-	3,876
	PID	131072	14,619,662
		16777216	52,719,212
		⋮	⋮
		657319493	1
	Size	4096	29,921,880
		65536	2,139,215
		⋮	⋮
		16699392	1
Behavior	Category	System	348,896,842
		Exception	591,779
		⋮	⋮
		Network	64,014,468
	API calls	NtClose	81,483,423
		SetFilePointer	52,079,566
		⋮	⋮
		timeGetTime	2,134,511

Table 5. Architecture of EfficientNet-B3 model used in feature selection and classification

Layer	Input channel	Output channel	Kernel size	Stride	Expand ratio	SE ratio
Conv 3×3	-	-	-	-	-	-
MBConvBlock	32	16	3	1	1	0.25
	16	24	3	2	6	0.25
	16	24	3	2	6	0.25
	24	40	5	2	6	0.25
	24	40	5	2	6	0.25
	40	80	3	2	6	0.25
	40	80	3	2	6	0.25
	40	80	3	2	6	0.25
	80	112	5	1	6	0.25
	80	112	5	1	6	0.25
	80	112	5	1	6	0.25
	112	192	5	2	6	0.25
	112	192	5	2	6	0.25
	112	192	5	2	6	0.25
	112	192	5	2	6	0.25
	192	320	3	1	6	0.25
Conv 1×1	-	-	-	-	-	-
Dropout	-	-	-	-	-	-
Dense (output)	-	-	-	-	-	-

4.2.2 Feature tensorization

To perform tensorization by transforming features in a string format into a real number format, the frequency count for each item was first calculated. Table 4 lists the frequency of each item.

After constructing a dictionary using the name and frequency of each item as a key and value, respectively, the values were normalized to a value between 0.1–1. In addition, a 3D tensor of 3×448×448 was created by mapping the dictionary to the process memory and behavior features of the file.

4.2.3 Feature selection and classification

In feature selection and classification, training, validation, and tests were performed using the EfficientNet-B3 model. Table 5 shows the layers and hyperparameters used in it.

The Adam optimizer was used to determine the weights optimized for the model. The learning rate was set to 0.001, the batch size was set to 32, and training was performed for a total of 30 epochs. When validation was performed on the constructed model, the validation loss was the lowest (0.2600) at 27 epochs, and the validation accuracy was 89.23%. In addition, as a result of performing a test on the validated model, the test accuracy was 94.98%.

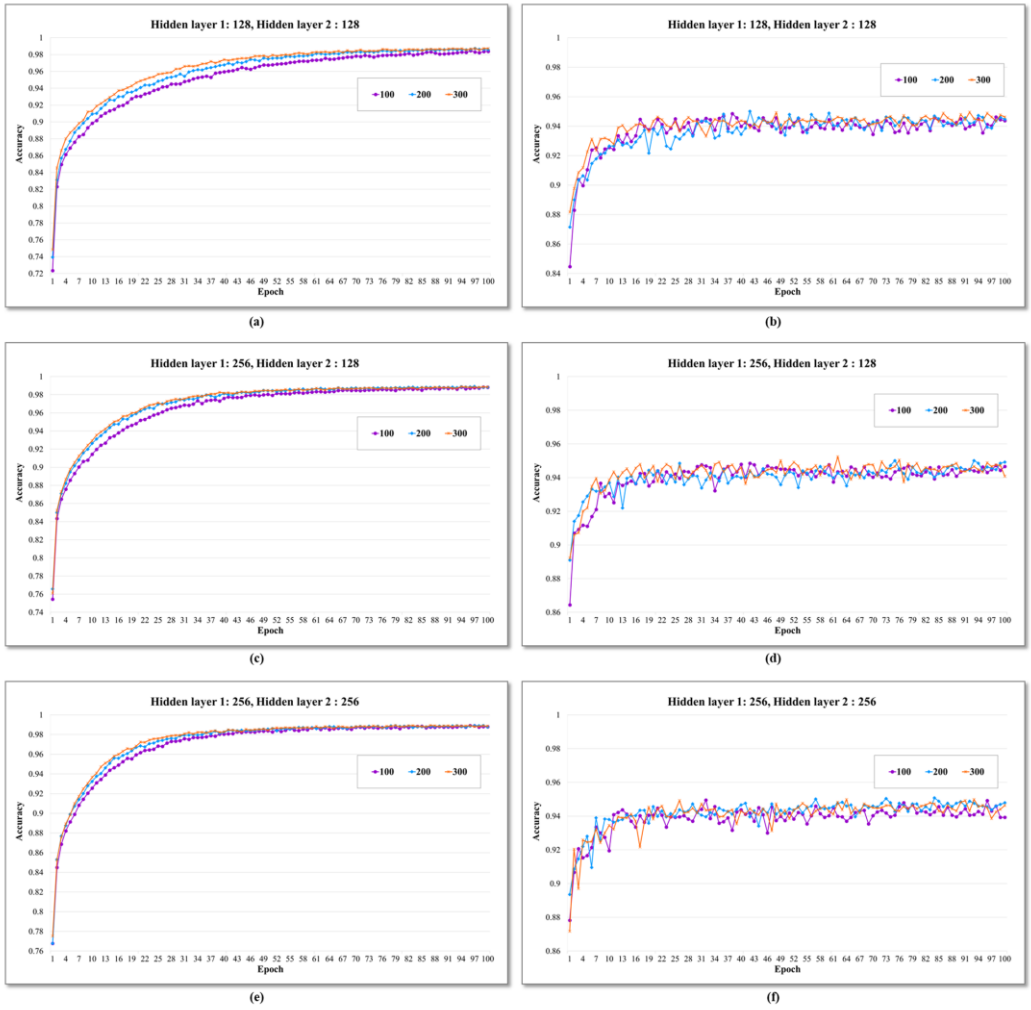

Fig. 3. Performance comparison of static malware detection using the Bi-LSTM model. (a)–(b) Training and validation accuracy of the model with the number of cells 128–128. (c)–(d) Training and validation accuracy of a model with the number of cells 256–128. (e)–(f) Training and validation accuracy of a model with the number of cells 256–256.

5. Performance Evaluation

An evaluation was performed to determine whether the 2-MaD scheme proposed in this study was capable of accurately detecting IoT malware with a low false detection rate by performing static and dynamic analyses for each applicable stage. The FNR and accuracy were used as performance indicators.

As shown in Equations (2) and (3), the results are represented as true positive (TP), false positive (FP), true negative (TN), and false negative (FN). FNR indicates the probability of misdetection of malware as benign, and accuracy indicates the rate of accurately detecting malware and benign files.

$$FNR = \frac{FN}{TP+FN} \tag{2}$$

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{3}$$

Fig. 3 shows the accuracy when training and validation are performed under various settings on the architecture of the Bi-LSTM model and the size of the embedding vector. When two hidden layer cells were set differently, as the number of cells in the hidden layer increased, the accuracy increased at lower epochs.

Table 6 shows the test accuracy and FNR of static malware detection with different settings of the embedding model, embedding vector size, and number of cells in the hidden layer. To test the performance of the model using the word embedding method, FastText, Word2Vec [37]—developed by Google—was used for comparison. In Table 6, the accuracy of detecting IoT malware using FastText can be seen to be consistently higher than that obtained using WordVec. In addition, the test accuracy was the highest (93.89%) when the number of cells of hidden layers 1 and 2 was 256 and 128, respectively, and the size of the embedding vector was 100.

Table 6. Various architectures of the Bi-LSTM model and performance comparison based on the embedding model

Number of cells in hidden layer 1	Number of cells in hidden layer 2	Embedding vector size	Embedding model	Accuracy (%)	FNR
128	128	100	Word2Vec	93.48	0.049
			FastText	93.54	0.062
		200	Word2Vec	93.16	0.056
			FastText	93.73	0.058
		300	Word2Vec	93.41	0.052
			FastText	93.57	0.064
256	128	100	Word2Vec	93.09	0.058
			FastText	93.89	0.062
		200	Word2Vec	92.97	0.061
			FastText	93.64	0.060
		300	Word2Vec	93.35	0.064
			FastText	93.61	0.064
256	256	100	Word2Vec	93.00	0.067
			FastText	93.25	0.065
		200	Word2Vec	93.22	0.053
			FastText	93.45	0.059
		300	Word2Vec	92.49	0.057
			FastText	93.64	0.069

Fig. 4 shows the training and validation results of dynamic malware detection using various CNN models and tensor sizes. To evaluate the performance of dynamic malware detection using the EfficientNet-B3 model, ResNet-50 [38] and ResNeXt-50 [39] models were used for comparison. Overall, the EfficientNet-B3 model showed higher training accuracy than the other two models, but the validation accuracy for the three CNN models was lower than the training accuracy due to overfitting. In addition, the validation accuracy of the EfficientNet-B3 model was higher than that of other models when the tensor size was 3×448×448, but the validation accuracy was lower when the tensor size was 3×224×224 and 3×336×336.

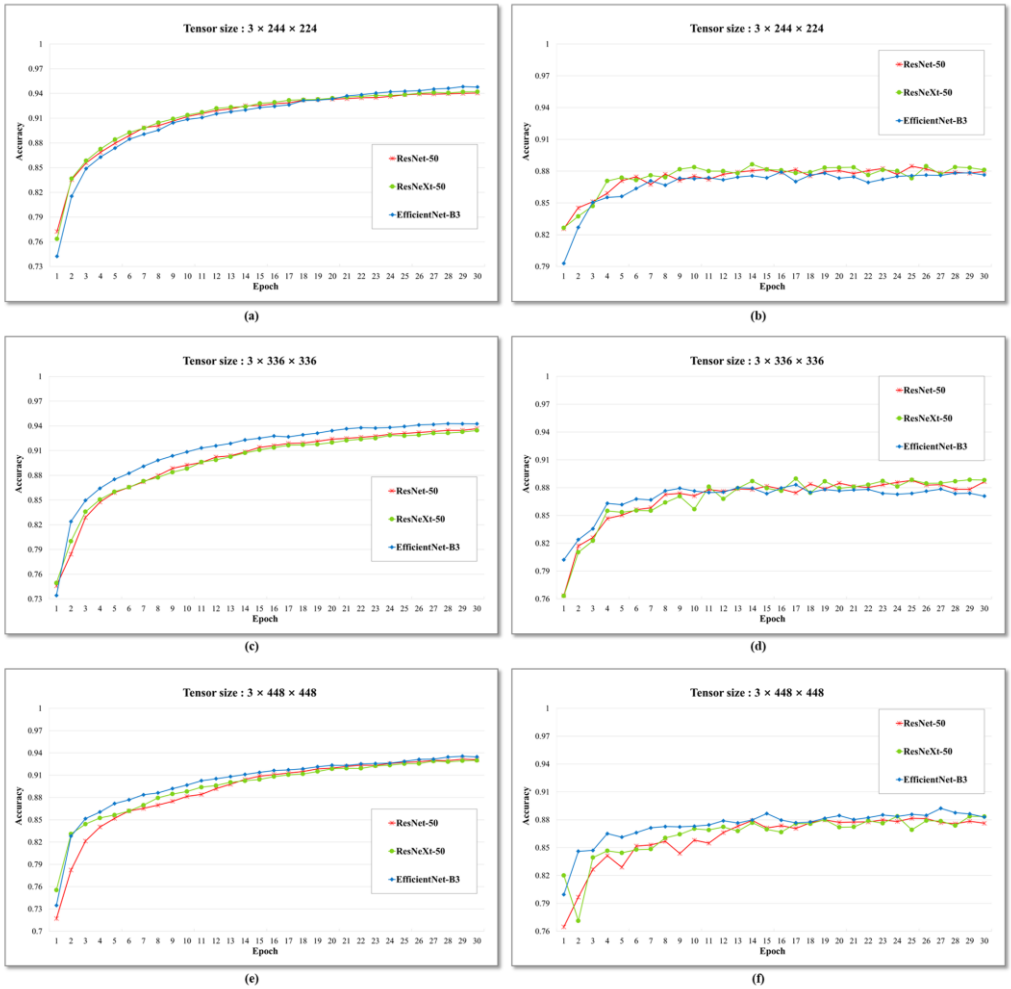

Fig. 4. Performance comparison of dynamic malware detection using various CNN models. (a)–(b) Training and validation accuracy with tensor size 3×224×224. (c)–(d) Training and validation accuracy with tensor size 3×336×336. (e)–(f) Training and validation accuracy with tensor size 3×448×448.

Table 7 compares the test accuracy and FNR of dynamic malware detection measured using three different types of CNN models and tensor sizes. The accuracy was measured to be the highest (94.98%) with IoT malware detection using the EfficientNet-B3 model with tensor size 3×448×448.

Table 7. Various CNN models and performance comparison with tensor size

CNN model	Tensor size	Accuracy (%)	FNR
ResNet-50	3 × 244 × 244	93.70	0.059
	3 × 336 × 336	94.57	0.060
	3 × 448 × 448	94.12	0.059
ResNeXt-50	3 × 244 × 244	94.09	0.065
	3 × 336 × 336	93.77	0.069
	3 × 448 × 448	94.73	0.062
EfficientNet-B3	3 × 244 × 244	93.60	0.083
	3 × 336 × 336	93.32	0.089
	3 × 448 × 448	94.98	0.062

Table 8. Performance comparison between static analysis, dynamic analysis and 2-MaD

Malware detection model	Accuracy (%)	FNR
Static analysis	93.89	0.053
Dynamic analysis	94.98	0.062
2-MaD	94.46	0.019

Table 8 outlines the test accuracy and FNR of the conventional malware detection model based on static and dynamic analyses, respectively, and the IoT malware detection performance of the 2-MaD model proposed in this study. When using the 2-MaD scheme, the accuracy was 94.46%, indicating a more accurate detection of obfuscated malware compared to the malware detection performance based on static analysis. Furthermore, the detection accuracy of 2-MaD was lower than that of the malware detection performance using dynamic analysis, but in this case, the FNR was only 0.019, indicating that the probability of incorrect classification of malware as benign files was extremely low.

6. Conclusion

To protect various IoT devices used in smart cities from advanced cyberattacks, a 2-MaD scheme that performs malware detection in two stages was proposed. 2-MaD performed dynamic analysis on files classified as benign from malware detection based on static analysis so that malware misclassified as benign could be detected in the second stage. The performance evaluation for 2-MaD showed that the detection accuracy was 94.46%, which was higher than that of the malware detection based on static analysis. Conversely, the accuracy of malware detection based on dynamic analysis was 94.98%, higher than that of 2-MaD, but the false detection rate of 2-MAD was significantly lower than that of dynamic analysis with an FNR of 0.019 for 2-MaD compared to 0.062 for dynamic analysis.

In this study, the 2-MaD model has resulted in loss of features due to the fixed size of the features generated through dynamic analysis in consideration of the computational performance. Therefore, in order to prevent such problem, a feature selection technique that uses weights calculated from TF-IDF and hashing will be further investigated. In addition, we aim to develop a method for reducing the time required for the feature extraction process in Stage 2.

Acknowledgements

Not applicable.

Author's Contributions

Conceptualization, SB. Investigation and methodology, SB, BJ. Project administration, SB, JJ, BJ, YSJ. Supervision, YSJ. Writing of the original draft, JJ. Writing of the review and editing, SB, JJ, BJ, YSJ. Software, SB, BJ, JJ. Validation, SB, BJ, JJ. All the authors have proofread the final version.

Funding

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1088383).

Competing Interests

The authors declare that they have no competing interests.

References

[1] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-khah, and P. Siano, "IoT-based smart cities: a survey," in *Proceedings of 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, Florence, Italy, 2016, pp. 1-6.

[2] X. Xiao, and C. Xie, "Rational planning and urban governance based on smart cities and big data," *Environmental Technology & Innovation*, vol. 21, article no. 101381, 2021. https://doi.org/10.1016/j.eti.2021.101381

[3] J. C. S. Sicato, S. K. Singh, S. Rathore, and J. H. Park, "A comprehensive analyses of intrusion detection system for IoT environment," *Journal of Information Processing Systems*, vol. 16, no. 4, pp. 975-990, 2020.

[4] A. Nieto, and R. Rios, "Cybersecurity profiles based on human-centric IoT devices," *Human-centric Computing and Information Sciences*, vol. 9, article no. 39, 2019. https://doi.org/10.1186/s13673-019-0200-y

[5] Y. S. Jeong, and J. H. Park, "Security, privacy, and efficiency of sustainable computing for future smart cities," *Journal of Information Processing Systems*, vol. 16, no. 1, pp. 1-5, 2020.

[6] M. U. Tariq, M. Babar, M. A. Jan, A. S. Khattak, M. D. Alshehri et al., "Security requirement management for cloud-assisted and internet of things—enabled smart city," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 625-639, 2021.

[7] Y. Al-Hadhrami and F. K. Hussain, "DDoS attacks in IoT networks: a comprehensive systematic literature review," *World Wide Web*, vol. 24, no. 3, pp. 971-1001, 2021.

[8] J. Cheng, J. Li, X. Tang, V. S. Sheng, C. Zhang, and M. Li, "A novel DDoS attack detection method using optimized generalized multiple kernel learning," *Computers, Materials & Continua*, vol. 62, no. 3, pp. 1423-1443, 2020.

[9] D. Ferraris, C. Fernandez-Gago, and J. Lopez, "A model-driven approach to ensure trust in the IoT," *Human-centric Computing and Information Sciences*, vol. 10, article no. 50, 2020. https://doi.org/10.1186/s13673-020-00257-3

[10] Y. Lee, S. Rathore, J. H. Park, and J. H. Park, "A blockchain-based smart home gateway architecture for preventing data forgery," *Human-centric Computing and Information Sciences*, vol. 10, article no. 9, 2020. https://doi.org/10.1186/s13673-020-0214-5

[11] S. R. T. Mat, M. F. A. Razak, M. N. M. Kahar, J. M. Arif, S. Mohamad, and A. Firdaus, "Towards a systematic description of the field using bibliometric analysis: Malware evolution," *Scientometrics*, vol. 126, no. 3, pp. 2013-2055, 2021.

[12] F. O. Catak, J. Ahmed, K. Sahinbas, and Z. H. Khand, "Data augmentation based malware detection using convolutional neural networks," *PeeJ Computer Science*, vol. 7, article no. e346, 2021. https://doi.org/10.7717/peerj-cs.346

[13] R. Sihwail, K. Omar, and K. A. Z. Ariffin, "An effective memory analysis for malware detection and classification," *Computers, Materials & Continua*, vol. 67, no. 2, pp. 2301-2320, 2021.

[14] J. Abawajy, A. Darem, and A. A. Alhashmi, "Feature subset selection for malware detection in smart IoT platforms," *Sensors*, vol. 21, no. 4, article no. 1374, 2021. https://doi.org/10.3390/s21041374

[15] Y. Hu, Y. Guo, J. Liu, and H. Zhang, "A hybrid method of coreference resolution in information security," *Computers, Materials & Continua*, vol. 64, no. 2, pp. 1297-1315, 2020.

[16] A. Afianian, S. Niksefat, B. Sadeghiyan, and D. Baptiste, "Malware dynamic analysis evasion techniques: a survey," *ACM Computing Surveys*, vol. 52, no. 6, pp. 1-28, 2019.

[17] T. Apostolopoulos, V. Katos, K. K. R. Choo, and C. Patsakis, "Resurrecting anti-virtualization and anti-debugging: Unhooking your hooks," *Future Generation Computer Systems*, vol. 116, pp. 393-405, 2021.

[18] M. Sewak, S. K. Sahay, and H. Rathore, "DRLDO: a novel DRL based de-obfuscation system for defence against metamorphic malware," *Defence Science Journal*, vol. 71, no. 1, pp. 55-65, 2021.

[19] J. Singh, and J. Singh, "A survey on machine learning-based malware detection in executable files," *Journal of Systems Architecture*, vol. 112, article no 101861, 2021. https://doi.org/10.1016/j.sysarc.2020.101861

[20] Z. Chen, X. Zhang, and S. Kim, "A learning-based static malware detection system with integrated feature," *Intelligent Automation & Soft Computing*, vol. 27, no. 3, pp. 891-908, 2021.

[21] C. Acarturk, M. Sirlanci, P. G. Balikcioglu, D. Demirci, N. Sahin, and O. A. Kucuk, "Malicious code detection: run trace output analysis by LSTM," *IEEE Access*, vol. 9, pp. 9625-9635, 2021.

[22] S. Kim, S. Yeom, H. Oh, D. Shin, and D. Shin, "Automatic malicious code classification system through static analysis using machine learning," *Symmetry*, vol. 13, no. 1, article 35, 2021. https://doi.org/10.3390/sym13010035

[23] B. Zhang, W. Xiao, X. Xiao, A. K. Sangaiah, W. Zhang, and J. Zhang, "Ransomware classification using patch-based CNN and self-attention network on embedded n-grams of opcodes," *Future Generation Computer Systems*, vol. 110, pp. 708-720, 2020.

[24] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," **Information Sciences**, vol. 231, pp. 64-82, 2013.

[25] A. Mahindru and A. L. Sangal, "MLDroid: framework for android malware detection using machine learning techniques," *Neural Computing & Applications*, vol. 33, no. 10, pp. 5183-5240, 2021.

[26] P. Mishra, I. Verma, and S. Gupta, "KVMInspector: KVM based introspection approach to detect malware in cloud environment," *Journal of Information Security and Applications*, vol. 51, article no. 102460, 2020. https://doi.org/10.1016/j.jisa.2020.102460

[27] G. D'Angelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on autoencoders and API-images," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 26-33, 2020.

[28] J. Devesa, I. Santos, X. Cantero, Y. K. Penya, and P. G. Bringas, "Automatic behaviour-based analysis and classification system for malware detection," in *Proceedings of the 12th International Conference on Enterprise Information Systems*, Madeira, Portugal, 2010, pp. 395-399.

[29] J. Jeon, J. H. Park, and Y. S. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96899-96911, 2020.

[30] R. Surendrana, T. Thomas, and S. Emmanuel, "A TAN based hybrid model for android malware detection," *Journal of Information Security and Applications*, vol. 54, article no. 102483, 2020. https://doi.org/10.1016/j.jisa.2020.102483

[31] A. Karim, V. Chang, and A. Firdaus, "Android botnets: a proof-of-concept using hybrid analysis approach," Journal of Organizational and End User Computing, vol. 32, no. 3, pp. 50-67, 2020. http://doi.org/10.4018/JOEUC.2020070105

[32] H. Darabian, S. Homayounoot, A. Dehghantanha, S. Hashemi, H. Karimipour, R. M. Parizi, and K. K. R. Choo, "Detecting cryptomining malware: a deep learning approach for static and dynamic analysis," *Journal of Grid Computing*, vol. 18, pp. 293-303, 2020.

[33] I. Santos, J. Devesa, F. Brezo, J. Nieves, and P. G. Bringas, "OPEM: a static-dynamic approach for machine-learning-based malware detection," in *International Joint Conference CISIS'12-ICEUTE´ 12-SOCO´ 12 Special Sessions*. Berlin, Germany: Springer, 2013, pp. 271-280.

[34] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135-146, 2017.

[35] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in *Proceedings of the IEEE International Conference on Neural Networks*, Montreal, Canada, 2005, pp. 2047-2052.

[36] M. Tan and Q. V. Le, "EfficientNet: rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, CA, 2019, pp. 6105-6114.

[37] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the 1st International Conference on Learning Representations (ICLR)*, Scottsdale, AZ, 2013, pp. 1-12.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, 2016, pp. 770-778.

[39] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, 2017, pp. 1492-1500.