Hybrid Malware Detection Based on Bi-LSTM and SPP-Net for Smart IoT

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

***Abstract*—In this article, we propose the hybrid malware detection scheme, HyMalD, with bidirectional long short-term memory (Bi-LSTM) and the spatial pyramid pooling network (SPP-Net). Its purpose is to protect Internet of Things (IoT) devices and minimize the damage caused by infection through obfuscated malware. HyMalD performs the static and dynamic analyses logically simultaneously to detect obfuscated malware, which is impossible to do using static analysis alone. First, it extracts static features of the opcode sequence using a reconstructed dataset according to the obfuscation and extracts the application programming interface (API) call sequence dynamically. The extracted features are trained through the Bi-LSTM and SPP-Net models, which HyMalD uses to detect and classify IoT malware. The performance of HyMalD was evaluated, and its detection accuracy was 92.5%. The false-negative rate (FNR) of HyMalD was 7.67%. Thus, HyMalD detects IoT malware more accurately and with a lower FNR than in the static analysis, which had 92.09% detection accuracy and 9.97% FNR.**

***Index Terms*—Bidirectional long short-term memory (Bi-LSTM), hybrid malware detection, Internet of Things (IoT) malware, Shannon entropy, spatial pyramid pooling network (SPP-Net).**

I. Introduction

THE core technologies of the Fourth Industrial Revolution—cloud computing, big data, and artificial intelligence—are fused with the Internet of Things (IoT) and are becoming integral to smart IoT industries, such as smart cities, smart homes, and smart healthcare. These IoT environments are found in various industries, where many embedded devices are connected through networks to collect and share data.

The number of malware attacks has greatly increased in recent years. These attacks include distributed denial of service (DDoS) and cryptomining and increasingly affect infrastructure IoT devices based on mobile environments, such as those in the healthcare industry. IoT devices with outdated operating systems are a common target due to their unknown vulnerabilities. If vulnerable IoT devices are infected by malware, users' sensitive data are leaked, and the devices malfunction. In addition, the malware spreads from an infected IoT device to others on the same network.

Various studies on malware detection have been conducted to address the security issues in new smart IoT-based industries [1], [2]. There are three categories of malware analysis techniques: static analysis, dynamic analysis, and hybrid analysis. In static analysis, files are not executed directly; instead, reverse engineering is executed against the source code, thereby verifying the files' overall structure [3], [4]. This technique analyzes a file quickly, but it has difficulty in detecting malware obfuscated by packing or encryption. On the other hand, dynamic analysis executes a file directly in a virtual environment such as a sandbox or emulator, thereby monitoring changes to the system and analyzing its behavior [5], [6]. It detects obfuscated malware because it analyzes real-time behaviors, but it requires more time than static analysis. Last, hybrid analysis utilizes both static and dynamic analyses to overcome the drawbacks of both [7], [8].

IoT devices, however, are optimized for a specific purpose and are miniaturized. Thus, their hardware specifications (e.g., battery and memory) are usually more limited than those of general devices. Accordingly, it is difficult for IoT devices to analyze and detect a large amount of malware because of their hardware limitations [9], [10].

We propose a hybrid malware detection based on bidirectional long short-term memory (Bi-LSTM) and spatial pyramid pooling network (SPP-Net): HyMalD analyzes and detects a huge variety of malware in a nested virtual environment. Our HyMalD scheme performs static and dynamic analyses logically and simultaneously to overcome the drawback of static analysis (the difficulty of detecting obfuscated malware) by utilizing a classified and reconstructed dataset according to whether there is obfuscation. The part of the HyMalD model that performs static analysis extracts the opcode sequence as a feature, and then learned by the Bi-LSTM model. On the other hand, the dynamic analysis part of HyMalD analyzes and monitors behavior related to application programming interface (API) call sequence features in real time by executing obfuscated files in a nested virtual environment. A great many API call sequence features are converted into images for deep-learning-based obfuscated malware detection. Here, obfuscated malware is detected by

Manuscript received July 2, 2021; revised September 16, 2021; accepted October 6, 2021. Date of publication October 14, 2021; date of current version April 13, 2022. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) under Grant 2019R1A2C1088383. Paper no. TII-21-2789. *(Corresponding author: Young-Sik Jeong.)*

The authors are with the Department of Multimedia Engineering, Dongguk University, Seoul 04620, South Korea (e-mail: jry02107@dongguk.edu; qudgml6323@dgu.ac.kr; tmddus-dls12@dongguk.edu; ysjeong@dongguk.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2021.3119778.

Digital Object Identifier 10.1109/TII.2021.3119778

1551-3203 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Milner Library Illinois State University. Downloaded on September 30,2024 at 23:45:52 UTC from IEEE Xplore. Restrictions apply.

training in the SPP-Net model using images of various sizes as input data.

HyMalD uses dynamic analysis to raise the detection rate of obfuscated malware, which is difficult to detect through static analysis. The size of the images created from the dynamic analysis is not fixed by cropping or resizing, thus eliminating any omission or distortion of important malware features, so HyMalD detects obfuscated malware in IoT devices with outdated operating systems.

The rest of this article is organized as follows. Section II describes related works on malware detection. Section III presents the overall scheme of the HyMalD model proposed in this study. Section IV designs the HyMalD model and Section V implements the HyMalD model. Section VI presents the model's performance evaluation results. Finally, Section VI concludes this article.

II. Related Works

This section discusses studies focusing on the static, dynamic, or hybrid analysis techniques of traditional malware detection. Although various methods based on these analysis techniques are available, this article describes methods that use the typical analysis techniques based on deep and machine learning [11]–[13].

A. Static Analysis for Malware Detection

Cui *et al.* [14] proposed a technique that classifies and detects variant malware by training the convolutional neural network (CNN) model with grayscale images generated from binary code. This technique solved the data imbalance problem between malware families by dynamically resampling images using the bat algorithm.

Zhu *et al.* [15] proposed DroidDet for detecting Android malware using various types of static information. DroidDet first extracted four types of static information and then determined the top features by applying the terms frequency-inverse document frequency (TF-IDF) and cosine similarity. The selected features were then used to classify and detect Android malware through the rotation forest classifier.

B. Dynamic Analysis for Malware Detection

Jeon *et al.* [6] proposed DAIMD, which detected malware by performing dynamic analysis in a cloud-based nested virtual environment. DAIMD treated behaviors in memory, the virtual file system, processes, the network, and system calls as dynamic features and created images through integerization, rescaling, and resizing. The generated images were used to train the CNN model to detect and classify malware.

Tang *et al.* [16] proposed a malware classification method that applied color-mapping rules to the API call sequence extracted through dynamic analysis. It singled out API calls belonging to 14 categories consisting of eight functions and six malicious intentions. It created 16 × 16-sized images by designating a color value according to the number of times the API category was called per unit time. The created feature images are classified into nine malware families through the CNN model.

C. Hybrid Analysis for Malware Detection

Eskandari *et al.* [17] proposed the HDM-analyzer, a malware detection model based on hybrid analysis, to improve the detection speed and accuracy. The HDM-analyzer created an enriched control flow graph (ECFG) through the API call sequence extracted from the hybrid analysis and detected malware using the ECFG as the input data in the decision stump, sequential minimal optimization, naïve Bayes, random tree, lazy KStar, and random forest classifiers. The HDM-analyzer clarified the ambiguous point that occurs in the static analysis using the API call sequence extracted from the dynamic analysis.

Han *et al.* [18] proposed MalDAE to detect malware by considering the correlation of the API call sequence extracted from the hybrid analysis. MalDAE fused the static and dynamic API call sequences into a single hybrid API call sequence using semantic mapping. The hybrid feature vector space was configured from the generated hybrid-fused API call sequence and applied to the random forest classifier for malware classification.

Martín *et al.* [19] proposed a malware detection approach that classified hybrid features using a combined ensemble classifier. AndroPyTool, which extracts hybrid features, performed prestatic, static, and dynamic analysis by applying Android application analysis tools, such as VirusTotal, and then generated features in the JavaScript object notation (JSON) and CSV formats. The features were classified by applying random forest and bagging to the created feature dataset. The results of classification were combined using the voting classifier.

Because our HyMalD model proposed in this study performs static and dynamic analyses simultaneously depending on whether the files to be analyzed are obfuscated, it improves the detection accuracy by reducing the false-negative rate (FNR) for obfuscated malware, the principal drawback of static analysis.

III. HyMalD Model

This article proposes the HyMalD model, which performs static- and dynamic-analysis-based malware detection logically simultaneously to detect obfuscated malware. The overall scheme of the HyMalD model is shown in Fig. 1.

Malware authors like to avoid detection by using obfuscation techniques such as compression, encryption, or packing. The obfuscated malware increases randomness, thereby increasing the file entropy [20]. HyMalD uses the Shannon entropy as an indicator of obfuscated malware. The Shannon entropy for files configured with 8-bit levels is expressed as presented in the following equation:

$$H(x) = -\sum_{i=1}^{n} P(i) \times \log_2 P(i) \tag{1}$$

where *P(i)* refers to the probability of information occurring in the *i*th block. The calculated entropy *H(x)* has a value between 0 and 8.

Authorized licensed use limited to: Milner Library Illinois State University. Downloaded on September 30,2024 at 23:45:52 UTC from IEEE Xplore. Restrictions apply.

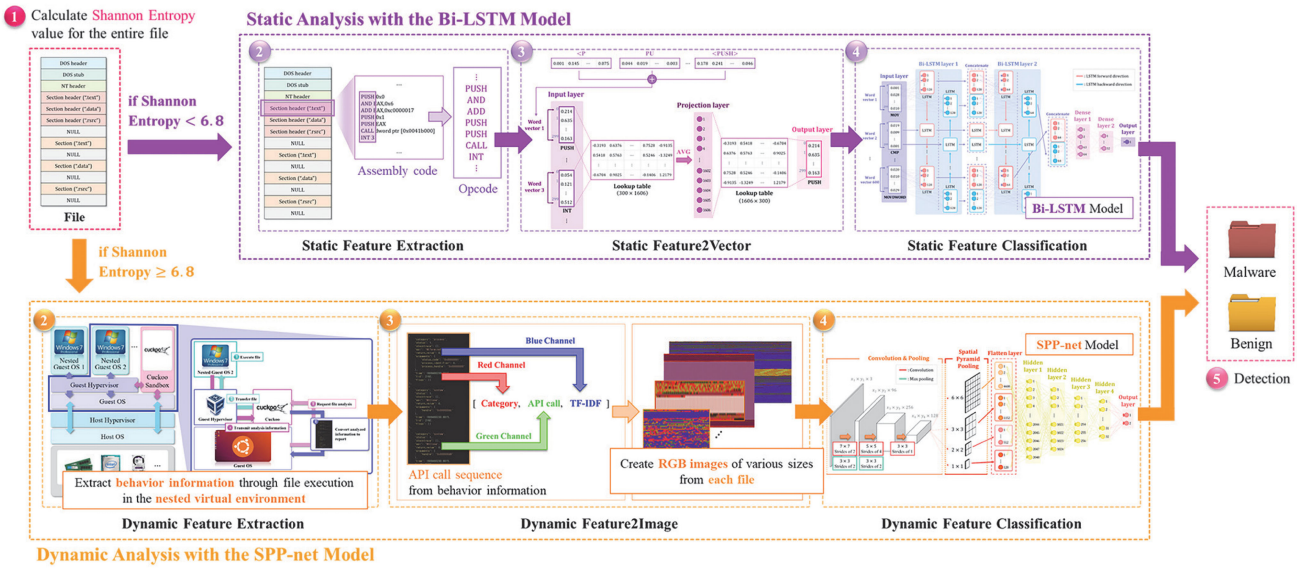


Fig. 1. Overall scheme of the HyMalD.

Lyda and Hamrock [20] generalized the range of entropy such that the average entropy of packed executables, for example, was around 6.8, and the highest entropy was around 7.2. Accordingly, for dataset reconfiguration, if the Shannon entropy of the file in HyMalD is more than 6.8, it is determined that the file is obfuscated. Otherwise, the file is considered not obfuscated.

Using the reconfigured dataset, static analysis with the Bi-LSTM model and dynamic analysis with the SPP-Net model are performed logically and simultaneously.

A. Static Analysis With the Bi-LSTM Model

Static analysis with the Bi-LSTM model analyzes and trains the code semantics and file type without an execution in three steps: static-feature extraction, static feature2vector, and static-feature classification.

Initially, the static-feature extraction step performs disassembly to convert the machine code into the assembly code. The converted assembly code consists of an opcode and one or more operands used by the opcode. HyMalD employs the opcode as a feature.

The static feature2vector step performs word embedding that converts the opcode sequence into a vector. It uses the FastText technique, which uses a word's internal structure to improve the vector representation. In the FastText technique, if there are opcodes with the same radical, such as MOV, MOVSB, and MOVSW, they have a similar vector, which is unlikely to result in much different learning. This improves the efficiency of learning [21].

Finally, the opcode sequence is learned using the Bi-LSTM model in the static-feature classification step. Bi-LSTM is a deep-learning model that performs bidirectional sequence analysis based on contexts. The learning speed is fast, and it solves the shortage problem of contextual connections in current LSTM models [22].

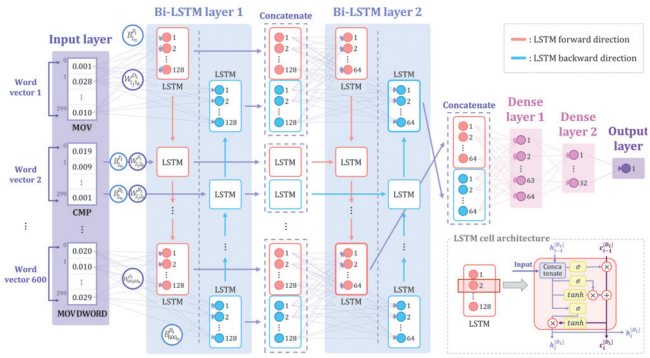


Fig. 2. Architecture of the Bi-LSTM model.

The architecture of Bi-LSTM is shown in Fig. 2, in which the static features of vectorized opcode sequences are used as inputs. In the Bi-LSTM layer, features with high importance are selected bidirectionally. The output layer determines whether the file is malware. Here, a weight $W_{i_j i_k}^{D_L}$ is used to express the importance assigned to each value in the input, Bi-LSTM, and output layers. Furthermore, bias $b_{i_n}^{D_L}$ is used to prevent an output value in a specific cell from being biased to 0. The value $h_{i-1}^{D_L}$ produced by the LSTM at the previous time and cell state value $c_{i-1}^{D_L}$ at the previous time are used. In D_L, D refers to the network direction in Bi-LSTM, in which the forward and backward directions are marked F and D. L refers to the layer index. In addition, i is the current time, j is the index at the previous layer, and k and n are the indices of the next layer.

B. Dynamic Analysis With the SPP-Net Model

Dynamic analysis with the SPP-Net model consists of dynamic-feature extraction, dynamic feature2image, and dynamic-feature classification. It executes obfuscated files, analyzes, and trains their behaviors.

Authorized licensed use limited to: Milner Library Illinois State University. Downloaded on September 30,2024 at 23:45:52 UTC from IEEE Xplore. Restrictions apply.

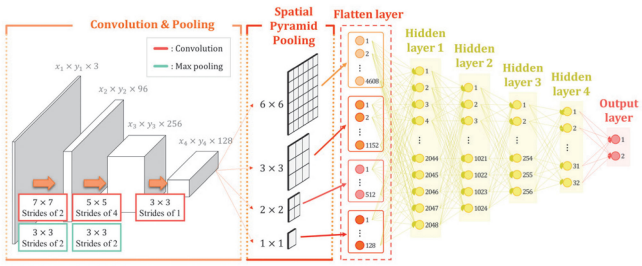


Fig. 3. Architecture of the SPP-Net model.

The dynamic-feature extraction step executes files in a nested virtual environment. It then monitors the behaviors in networks, processes, files, and the registry in real time through the Cuckoo Sandbox. Next, it generates a report in the JSON format and extracts the API call sequence and information about the API call.

The dynamic feature2image step channelizes the API call sequence and its information to convert the features into red–green–blue (RGB) images. The height of each channel is determined according to the file size with a fixed width [23].

The category, one of the pieces of the API-call-related information, is matched to the red channel to convert dynamic features into images. There are 17 categories, including _notification_, certificate, crypto, file, network, process, and registry, and they are adjusted into a pixel value within 0–255.

API calls are normalized to values between 0 and 255 based on a specific section calculated from the number of API calls in the same category, and then matched to the green channel.

Finally, the TF-IDF is calculated to represent the importance of the API call in each file as a numerical value and is matched to the blue channel. The TF-IDF is obtained by the following equation:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \tag{2}$$

where $w_{i,j}$ refers to the importance of the *i*th API call in the *j*th file and $tf_{i,j}$ is the frequency of the *i*th API call in the *j*th file. In addition, *N* denotes the total number of files, and df_i refers to the number of files where the *i*th API call is included.

The red, green, and blue channels generated through the channelization process are converted into RGB images.

The dynamic-feature classification step utilizes the SPP-Net model to train the RGB images, thereby detecting obfuscated malware. The architecture of an SPP-Net model is shown in Fig. 3. The convolution feature map for an image is calculated in the convolution layer. The SPP layer performs pooling of various sizes, creating a single feature vector [24]. The files that execute malicious behaviors are classified and detected as malware using the generated feature vector.

The SPP-Net model trains images consisting of features of various sizes, thereby preventing the loss of important features in malware [25], [26].

The detection results of malware using HyMalD are predicted by integrating the results of the static analysis with the Bi-LSTM model and results of the dynamic analysis with the SPP-Net model.

IV. HyMalD Design

This article uses the auxiliary, machinery, processing, signature, and reporting modules provided by the Cuckoo Sandbox to analyze and extract the real-time behavior of malware in a virtual environment. Moreover, the HyMalD model is added to analyze and detect malware using hybrid analysis and deep-learning techniques.

The auxiliary module in the Sandbox plays an auxiliary role in clearing the virtual environment before virtual machine execution or capturing the network traffic while it is being executed. The machinery module manages the execution and termination of all virtual machines while Cuckoo is being operated via the cuckoo.conf file. The processing module analyzes the behavior results collected from the virtual machine using 23 modules (including static, strings, procmon, debug, and network). It structures them into usable data in the signature and reporting modules. The signature module provides the behavior patterns of various operating systems and matches a pattern with the content analyzed in the processing module. Finally, the reporting module stores the analysis results generated through the processing and signature modules in JSON file format and provides them to users [27].

HyMalD consists of a nested virtual environment, static analysis with Bi-LSTM, dynamic analysis with SPP-Net, and a database. The nested virtual environment provides a secure virtual environment for dynamic analysis. The static analysis of Bi-LSTM consists of three operations: static-feature extraction, which extracts the opcode sequence from the assembly file generated in the static analysis; static feature2vector, which converts the extracted opcode sequence into a vector; and static-feature classification, which builds a malware detection model by training the vectorized opcode sequence in the Bi-LSTM model.

Dynamic-feature extraction in dynamic analysis with SPP-Net extracts the API call sequence from the generated report. The dynamic feature2image converts a large number of API call sequences into images. Then, the dynamic-feature classification builds an obfuscated malware detection model by training the generated images on the SPP-Net model.

The HyMalD database consists of a file database that contains datasets, a feature database that collects features extracted by the static and dynamic analyses, a classification result database with the integrated detection results of the static and dynamic analysis, and a classifier model that stores the models trained and built using the analysis techniques.

The basic components provided by Cuckoo Sandbox and HyMalD are connected to each other. The HyMalD database transfers obfuscated malware to the Cuckoo Sandbox through the command-line interface (CLI). The nested virtual environment also exchanges analysis information and command about the Cuckoo Sandbox and virtual environment management through the CLI. In addition, dynamic analysis with the SPP-Net

Authorized licensed use limited to: Milner Library Illinois State University. Downloaded on September 30,2024 at 23:45:52 UTC from IEEE Xplore. Restrictions apply.

receives the generated report from the reporting module inside the Cuckoo Sandbox and then extracts the API call sequence.

V. HyMalD Implementation

To detect malware that attacks vulnerable IoT devices in a mobile environment, the proposed HyMalD model was implemented in a system with a GeForce RTX 2070 and an Intel Core i7-9700K. In addition, a nested virtual environment was implemented to prevent the proliferation of malware to other devices connected over the network while dynamic analysis was being run. The implemented nested virtual environment was configured with a guest operating system in Ubuntu 18.04 and the nested guest operating system in Windows 7.

HyMalD trained and built the model using a dataset consisting of malware and benign files to detect malware that attacks IoT devices with outdated operating systems. The dataset used was KISA-data challenge 2019-Malware.04, provided by the Korea Internet & Security Agency, wherein 38 166 files were employed, except for data whose file format was not portable executable (PE). There were 23 634 files that were classified as not obfuscated based on a Shannon entropy of 6.8 and 14 532 files that were determined to be obfuscated.

A. Static Analysis With the Bi-LSTM Model

Static analysis with the Bi-LSTM model used 18 908 training datasets to build a static analysis-based malware detection model. A total of 2363 validation datasets were used to verify the built model. In addition, a total of 2363 test datasets were used to test the model. To perform static-feature extraction, PE files were analyzed using Pydasm, a Python-based static analysis tool, and the opcode was extracted. Duplicate opcodes in the extracted opcode sequence were removed (a meaningless opcode such as "add" was duplicated to avoid the analysis in some malware).

Depending on the computer specifications used in the study, a 1607-word vocabulary, including <PAD>, from the opcode sequence was generated for all the datasets by performing static feature2vector after fixing the length of the opcode sequence for a single file to 600. In this case, the FastText structure with an input layer composed of ten opcodes, a 1606-dimensional projection layer, and a 300-dimensional output layer was used.

Finally, the Bi-LSTM model used for malware detection in the static-feature classification step had one 300-dimensional embedded layer, two 256- and 128-D Bi-LSTM layers, two dense layers, and one output layer.

The training result of the Bi-LSTM model showed 94.12% average detection accuracy for nonobfuscated malware. The accuracy when verifying the built Bi-LSTM model was 91.90%, and the test accuracy was 91.59%.

B. Dynamic Analysis With the SPP-Net Model

The dynamic analysis with the SPP-Net model used a dataset with 14 743 samples to build the dynamic-analysis-based malware detection model, which consisted of 14 532 samples of obfuscated dataset classified based on the Shannon entropy and of a dataset with 211 samples whose opcode sequence was 0 in the static analysis with the Bi-LSTM model. A dataset with an opcode sequence of 0 was added because some of the obfuscated malware use the adversary technique, which reduces the entropy by taking an action such as adding strings of benign files or reducing the amount of packed data.

The reconstructed dataset was executed for 2 min in a virtual environment, and the behaviors were monitored through the Cuckoo Sandbox and a report was generated. Some malware recognizes that it is being analyzed in a virtual environment, so the behavior is not recorded in the report. There were 2623 of malware with such an anti-debugging function, and they were removed from the dataset.

Thus, the final dataset used to build the obfuscated-malware detection model has 12 120 samples, 8231 of which were malware samples and 3889 of which were benign samples. The dynamic analysis with the SPP-Net employed 9698 samples in the training dataset for malware detection, 1211 samples in the validation dataset, and 1211 samples in the test dataset.

Dynamic-feature extraction obtained the API call sequence that exhibited the behavior of the file from the report. There were 303 API call types for entire datasets, and the file with the longest API call sequence had 2 818 616 API calls.

The dynamic feature2image created an RGB image by channelizing the API call sequence and information on the API call. The generated images were of various sizes, the largest being 1024×1325 and the smallest being 32×5.

In dynamic-feature classification, the SPP-Net model had three convolutional layers, two pooling layers, four pyramids, four fully connected layers, and one output layer, and detected obfuscated malware using images.

As a result of training the SPP-Net model, the average accuracy was 91.23%; when the built model was verified, the average accuracy was 89.93%. Conducting the test resulted in detecting obfuscated malware with 93.31% accuracy.

The results of building the HyMalD model through static analysis with the Bi-LSTM model and dynamic analysis with the SPP-Net model showed that malware was detected with 92.50% accuracy. In addition, the HyMalD model had a 7.6% probability of falsely detecting IoT malware as benign and a 7.2% probability of detecting a benign sample as malware.

VI. Performance Evaluation

In this article, we conducted a performance evaluation to determine that HyMalD accurately detects obfuscated malware by performing static and dynamic analyses logically and simultaneously. The following evaluation indices were used to conduct the performance evaluation of the built HyMalD model: accuracy, FNR, and false-positive rate (FPR).

These three indices employed a confusion matrix in the evaluation. The confusion matrix contained four types of information: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP, TN, FP, and FN, respectively, pertain to the number of actual malware samples predicted as malware, the number of benign samples predicted as benign, the number of benign samples falsely predicted as malware, and the number of malware samples predicted as benign.

Authorized licensed use limited to: Milner Library Illinois State University. Downloaded on September 30,2024 at 23:45:52 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISON OF PERFORMANCES OF STATIC ANALYSIS AND HYMALD

Analysis technique	Accuracy (%)	FNR (%)	FPR (%)
Static analysis	92.09	9.97	5.54
HyMalD	92.50	7.67	7.25

Accuracy is the index used to evaluate whether the trained model could accurately classify malware; it was measured using the following equation:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}) . \quad (3)$$

FNR is the ratio of malware samples falsely predicted as benign, which is expressed by the following equation:

$$\text{FNR} = \text{FN} / (\text{TP} + \text{FN}) . \quad (4)$$

FPR is the ratio of benign samples falsely predicted as malware, which is expressed by the following equation:

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN}) . \quad (5)$$

The performance of the HyMalD model was measured with Shannon entropies of 6.8, 7.0, and 7.2. The malware detection accuracy of the HyMalD model based on 6.8 was 92.50%, which is higher than 91.46% at 7.0 and 91.76% at 7.2.

Table I compares the static analysis and HyMalD performance results for a Shannon entropy of 6.8. The detection accuracy was 92.50%, which was higher than that of static analysis (92.09%). In addition, the ratio of false negatives was 7.67%, which was significantly lower than that of static analysis (9.97%).

Fig. 4 shows the results of training and validating Bi-LSTM and LSTM models according to the Shannon entropy of 6.8 criteria. Fig. 4(a) shows the training accuracy of the Bi-LSTM and LSTM models measured when the epoch was 100. Although both models showed an increasing trend in training accuracy as the number of epochs increased, the accuracy of malware detection was higher when the Bi-LSTM model was used.

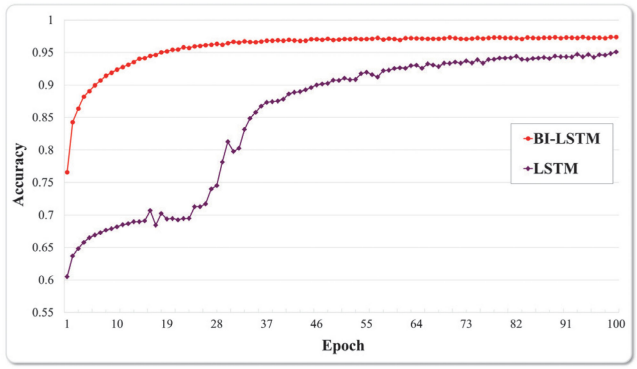
Fig. 4(b) shows the validation accuracies of the built Bi-LSTM and LSTM models. The validation accuracy of the Bi-LSTM model was higher than that of the LSTM model.

The test results of the Bi-LSTM and LSTM models revealed 92.09% and 87.09% test accuracies, respectively, indicating that the Bi-LSTM model more accurately detects nonobfuscated malware.

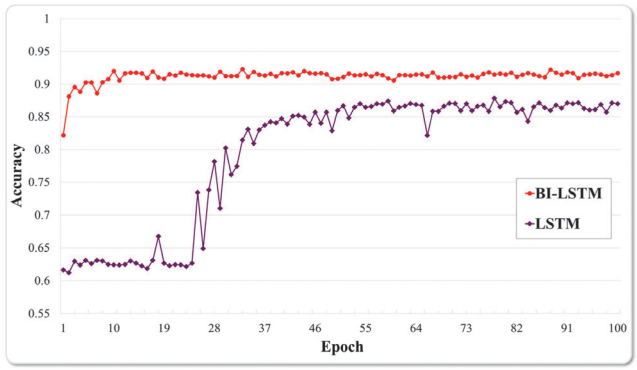
Fig. 5 shows the training and validation accuracies measured when the obfuscated malware detection model was built using the SPP-Net and CNN models. The SPP-Net model has various-sized images as inputs. The CNN model uses only images of a single size as inputs. Thus, to compare the performances of the SPP-Net and CNN models, the sizes of the images input to the CNN model were 64 × 64, 112 × 112, 128 × 128, 256 × 256, 384 × 384, and 512 × 512.

Fig. 5(a) shows the training accuracy of the SPP-Net and CNN models. Although the accuracy of the CNN and SPP-Net models increased as the number of epochs increased, the SPP-Net model's training accuracy was lower than that of the CNN model.

Fig. 5(b) shows the validation accuracy of the built SPP-Net and CNN models. When images of size 64 × 64, 128 × 128, and

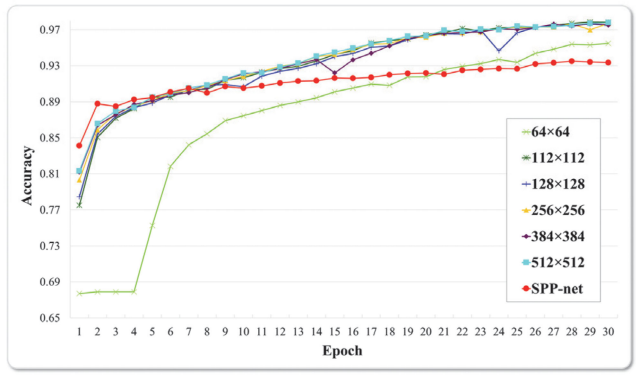


(a)

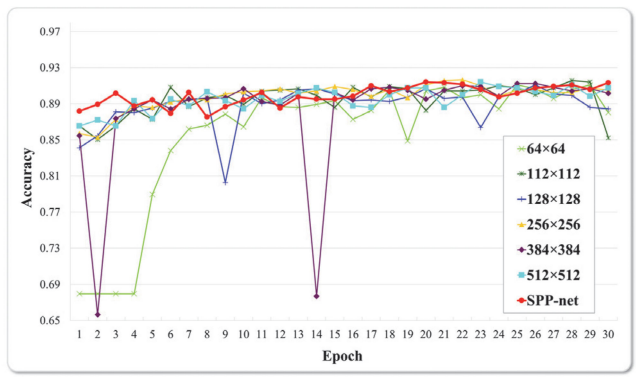


(b)

Fig. 4. Accuracy levels of the Bi-LSTM and LSTM models. (a) Training. (b) Validation.



(a)



(b)

Fig. 5. Accuracy of the SPP-Net and CNN models. (a) Training. (b) Validation.

Authorized licensed use limited to: Milner Library Illinois State University. Downloaded on September 30,2024 at 23:45:52 UTC from IEEE Xplore. Restrictions apply.

TABLE II
COMPARISON OF PERFORMANCES OF STATIC ANALYSIS AND HYMALD

Related work	Environment	Analysis technique	Accuracy (%)
Zhu *et al.* [15]	Android	Static analysis	88.26
Andrade *et al.* [29]	Windows	Static analysis	90.63
Han *et al.* [18]	Windows	Hybrid analysis	97.89
HyMalD	Windows	Hybrid analysis	94.85

384 × 384 were input to the CNN model, the accuracy significantly fluctuated in some sections because the model overfitted to the training dataset did not accurately predict the validation dataset. On the other hand, a nearly constant validation accuracy was measured for the SPP-Net model even for an increasing number of epochs. The gap between the training and validation accuracies for the CNN model became wider with an increasing number of epochs.

The test accuracies of the CNN model were measured at 87.28%, 84.39%, 90.1%, 90.3%, 90.7%, and 91.24% for 64 × 64, 112 × 112, 128 × 128, 256 × 256, 384 × 384, and 512 × 512 input image sizes, respectively. On the other hand, the test accuracy of the SPP-Net model was 93.3%, showing that detection of its obfuscated malware was superior without fixing the size of the feature image.

Table II compares the malware detection models of other studies and the HyMalD model proposed in this study. The four malware detection models collected datasets from VirusShare [28], a malware sample repository. The following points of comparison were used: the operating systems where malware was run; the technique used to analyze malware; and the accuracy of detecting malware.

To test the built HyMalD model, 1000 malware and benign were collected from VirusShare [28] and Windows environments, respectively. As a result, the accuracy of the HyMalD model was 94.85%, which showed higher accuracy than the study that detected malware using the existing static analysis techniques. This means that the HyMalD model detects obfuscated malware with a relatively high probability compared to the static analysis-based malware detection model. On the other hand, the accuracy of the HyMalD model was measured low compared to the existing hybrid analysis model that performed both static and dynamic analysis on a single file.

VII. CONCLUSION

This article proposed a HyMalD scheme that logically performed static and dynamic analyses simultaneously to detect malware in IoT devices used in the healthcare industry. HyMalD checked whether sample files were obfuscated utilizing their Shannon entropy. A dynamic analysis was performed for obfuscated files; for non-obfuscated files, a static analysis was performed. The extracted features were trained in the Bi-LSTM and SPP-Net models. The performance evaluation of HyMalD was 92.5% accuracy and 7.67% FNR, which is less likely to incorrectly predict malware as benign files compared to the 92.09% accuracy and 9.67% FNR that was the performance when static analysis was used to detect IoT malware.

Some malware reduces the entropy, which is usually detected as high due to obfuscation, by taking actions such as adding strings used in benign samples or reducing the amount of encrypted data. When the Shannon entropy is applied to such malware, the malware is falsely classified due to the obfuscated files. Another type of intelligent malware recognizes when it is running on a virtual machine and then performs anti-debugging, such as hiding the process or causing a crash to prevent the virtual machine from operating, thus escaping detection. Future study will improve our HyMalD model by further analyzing obfuscation and anti-debugging techniques to detect currently undetectable malware.

REFERENCES

[1] D. Kim, Y. Pan, and J. H. Park, "A study on the digital forensic investigation method of clever malware in IoT devices," *IEEE Access*, vol. 8, pp. 224487–224499, 2020.
[2] A. Makkar, S. Garg, N. Kumar, M. S. Hossain, A. Ghoneim, and M. Alrashoud, "An efficient spam detection technique for IoT devices using machine learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 903–912, Feb. 2021.
[3] Y. Pan, X. Ge, C. Fang, and Y. Fan, "A systematic literature review of android malware detection using static analysis," *IEEE Access*, vol. 8, pp. 116363–116379, 2020.
[4] G. Sun and Q. Qian, "Deep learning and visualization for identifying malware families," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 283–295, Jan./Feb. 2021.
[5] 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.
[6] 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.
[7] W. Zhang, H. Wang, H. He, and P. Liu, "DAMBA: Detecting android malware by ORGB analysis," *IEEE Trans. Rel.*, vol. 69, no. 1, pp. 55–69, Mar. 2020.
[8] X. Wang, Y. Yang, and S. Zhu, "Automated hybrid analysis of android malware through augmenting fuzzing with forced execution," *IEEE Trans. Mobile Comput.*, vol. 18, no. 12, pp. 2768–2782, Dec. 2019.
[9] H. V. Le and Q. D. Ngo, "V-Sandbox for dynamic analysis IoT Botnet," *IEEE Access*, vol. 8, pp. 145768–145786, 2020.
[10] F. S. Ahmed, N. Mustapha, A. Mustapha, M. Kakavand, and C. F. M. Foozy, "Preliminary analysis of malware detection in opcode sequences within IoT environment," *J. Comput. Sci.*, vol. 16, no. 9, pp. 1306–1318, Jun. 2020.
[11] Q. D. Ngo, H. T. Nguyen, V. H. Le, and D. H. Nguyen, "A survey of IoT malware and detection methods based on static features," *ICT Exp.*, vol. 6, no. 4, pp. 280–286, Dec. 2020.
[12] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Hum.-Centric Comput. Inf. Sci.*, vol. 8, no. 1, pp. 1–22, Dec. 2018.
[13] J. Kang and Y. Won, "A study on variant malware detection techniques using static and dynamic features," *J. Inf. Process. Syst.*, vol. 16, no. 4, pp. 882–895, Aug. 2020.
[14] Z. Cui, F. Xue, X. Cai, Y. Cao, G. G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018.
[15] H. J. Zhu, Z. H. You, Z. X. Zhu, W. L. Shi, X. Chen, and L. Cheng, "DroidDet: Effective and robust detection of android malware using static analysis along with rotation forest model," *Neurocomputing*, vol. 272, pp. 638–646, Jan. 2018.
[16] M. Tang and Q. Qian, "Dynamic API call sequence visualisation for malware classification," *IET Inf. Secur.*, vol. 13, no. 4, pp. 367–377, Jul. 2019.
[17] M. Eskandari, Z. Khorshidpour, and S. Hashemi, "HDM-Analyser: A hybrid analysis approach based on data mining techniques for malware detection," *J. Comput. Virol. Hacking Techn.*, vol. 9, no. 2, pp. 77–93, Feb. 2013.

Authorized licensed use limited to: Milner Library Illinois State University. Downloaded on September 30,2024 at 23:45:52 UTC from IEEE Xplore. Restrictions apply.

[18] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, and L. Mao, "MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics," *Comput. Secur.*, vol. 83, pp. 208–233, Jun. 2019.
[19] A. Martín, R. Lara-Cabrera, and D. Camacho, "Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset," *Inf. Fusion*, vol. 52, pp. 128–142, Dec. 2019.
[20] R. Lyda and J. Hamrock, "Using entropy analysis to find encrypted and packed malware," *IEEE Secur. Privacy*, vol. 5, no. 2, pp. 40–45, Mar./Apr. 2007.
[21] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, Jun. 2017.
[22] G. Wang, T. Lu, and H. Yin, "Detection technology of malicious code family based on BiLSTM-CNN," *J. Phys., Conf. Ser.*, vol. 1650, no. 3, pp. 1–7, Oct. 2020.
[23] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. Int. Symp. Vis. Cyber Secur.*, 2011, pp. 1–7.
[24] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
[25] S. Sriram, R. Vinayakumar, V. Sowmya, M. Alazab, and K. P. Soman, "Multi-scale learning based malware variant detection using spatial pyramid pooling network," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2020, pp. 740–745.
[26] P. Zhang, B. Sun, R. Ma, and A. Li, "A novel visualization malware detection method based on Spp-Net," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2019, pp. 510–514.
[27] R. V. Zutphen, "Cuckoo sandbox architecture," *eForensics Mag.*, Accessed: Jan. 30, 2021. Apr. 2019. [Online]. Available: https://eforensicsmag.com/cuckoo-sandbox-architecture/
[28] VirusShare, Accessed: May 27, 2021. [Online]. Available: https://virusshare.com/
[29] E. de O. Andrade, J. Viterbo, C. N. Vasconcelos, J. Guérin, and F. C. Bernardini, "A model based on LSTM neural networks to identify five different types of malware," *Procedia Comput. Sci.*, vol. 159, pp. 182–191, Jan. 2019.

Jueun Jeon was born in Seoul, South Korea, in 1995. She received the B.S. and M.S. degrees in multimedia engineering in 2018 and 2020, respectively, from Dongguk University, Seoul, South Korea, where she is currently working toward the Ph.D. degree in information security with the Department of Multimedia Engineering.

She is the first author of "Resource utilization scheme of idle virtual machines for multiple large-scale jobs based on OpenStack" published in *Applied Sciences* in 2019, and "Dynamic analysis for IoT malware detection with convolution neural network model" published in IEEE ACCESS in 2020. Her current research interests include information security for cloud computing and the Internet of Things.

Byeonghui Jeong was born in Suwon, South Korea, in 1996. He received the B.S. degree in computer science and engineering from Kongju National University, Cheonan, South Korea, in 2021. He is currently working toward the M.S. degree in resource management in cloud computing with the Department of Multimedia Engineering, Dongguk University, Seoul, South Korea.

He is the co-author of "Two-stage hybrid malware detection using deep learning" published in *Human-Centric Computing and Information Sciences* in 2021. He is conducting research on efficient resource management techniques in container-based cloud computing using deep learning. His current research interests include cloud computing, the Internet of Things, and information security for cloud computing.

Seungyeon Baek was born in Seoul, South Korea, in 1996. He received the B.S. degree in multimedia engineering in 2021 from Dongguk University, Seoul, South Korea, where he is currently working toward the M.S. degree in information security with the Department of Multimedia Engineering.

He is the first author of "Two-stage hybrid malware detection using deep learning" published in *Human-centric Computing and Information Sciences* in 2021. He is conducting research on solving problems in various fields, such as music and cloud computing, using deep learning. His current research interests include information security for IoT devices and malware detection using deep learning.

Young-Sik Jeong (Member, IEEE) received the B.S. degree in mathematics and the M.S. and Ph.D. degrees in computer science and engineering from Korea University, Seoul, South Korea, in 1987, 1989, and 1993, respectively.

From 1993 to 2012, he was a Professor with the Department of Computer Engineering, Wonkwang University, Iksan, South Korea. He worked and conducted research with Michigan State University, East Lansing, MI, USA, and Wayne State University, Detroit, MI, as a Visiting Professor, in 1997 and 2004, respectively. He is currently working with the Department of Multimedia Engineering, Dongguk University, Seoul. His research interests include multimedia cloud computing and information security.

Prof. Jeong is also an Executive Editor of the *Journal of Information Processing Systems*, an Associate Editor for the *Journal of Supercomputing* and *Journal of Human-Centric Computing*, and an Editor of the *Journal of Internet Technology*.

Authorized licensed use limited to: Milner Library Illinois State University. Downloaded on September 30,2024 at 23:45:52 UTC from IEEE Xplore. Restrictions apply.