IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 29, 2020, accepted May 10, 2020, date of publication May 20, 2020, date of current version June 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2995887

Dynamic Analysis for IoT Malware Detection With Convolution Neural Network Model

JUEUN JEON[1], JONG HYUK PARK[2], (Member, IEEE), AND YOUNG-SIK JEONG[1], (Member, IEEE)

[1]Department of Multimedia Engineering, Dongguk University, Seoul 04620, South Korea
[2]Department of Computer Science and Engineering, SeoulTech University, Seoul 01811, South Korea

Corresponding author: Young-Sik Jeong (ysjeong@dongguk.edu)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2019R1A2C1088383).

ABSTRACT Internet of Things (IoT) technology provides the basic infrastructure for a hyper connected society where all things are connected and exchange information through the Internet. IoT technology is fused with 5G and artificial intelligence (AI) technologies for use various fields such as the smart city and smart factory. As the demand for IoT technology increases, security threats against IoT infrastructure, applications, and devices have also increased. A variety of studies have been conducted on the detection of IoT malware to avoid the threats posed by malicious code. While existing models may accurately detect malicious IoT code identified through static analysis, detecting the new and variant IoT malware quickly being generated may become challenging. This paper proposes a dynamic analysis for IoT malware detection (DAIMD) to reduce damage to IoT devices by detecting both well-known IoT malware and new and variant IoT malware evolved intelligently. The DAIMD scheme learns IoT malware using the convolution neural network (CNN) model and analyzes IoT malware dynamically in nested cloud environment. DAIMD performs dynamic analysis on IoT malware in a nested cloud environment to extract behaviors related to memory, network, virtual file system, process, and system call. By converting the extracted and analyzed behavior data into images, the behavior images of IoT malware are classified and trained in the Convolution Neural Network (CNN). DAIMD can minimize the infection damage of IoT devices from malware by visualizing and learning the vast amount of behavior data generated through dynamic analysis.

INDEX TERMS Cloud-based malware detection, convolution neural network, dynamic analysis, IoT malware, malware detection.

I. INTRODUCTION

Recently, artificial intelligence (AI), virtual reality (VR), big data, 5G, and Internet of Things (IoT), which are the core technologies of the Fourth Industrial Revolution, have been used in various fields by integrating them across the industry. Especially, as IoT technologies and core technologies such as AI and 5G are converging, various IoT industries such as smart cars, smart factories, and smart cities are rapidly being activated. The scale of the IoT market continues to increase, and IoT devices, infrastructure, and applications significantly affect not only industrial fields but also daily living [1]–[7].

In the IoT environment, devices are connected with each other and exchange information. Because of this characteristic, the number of attacks such as distributed denial of service (DDoS), cryptocurrency malicious mining, and botnet activities are expanding at a fast pace [4]–[6], [8]–[14]. In addition, to cope with the rapidly increasing demand for IoT devices, some manufacturers are mass producing IoT devices that are vulnerable to security breaches and are providing them to users. If vulnerable IoT devices are distributed in the market, they will be a main target for malware makers. Malware could not only leak user information collected by IoT devices but also penetrate major networks, resulting in its rapid expansion to other networks [4], [9], [11], [12]. Kaspersky Lab, a cybersecurity product developer in Russia, collected 121,588 IoT malware samples in 2018, approximately quadruple the 32,614 samples it collected in 2017; more than 120,000 variant IoT malware samples whose attack methods were evolved intelligently were discovered [15].

To reduce damage from malware infection by protecting IoT devices from new and variant malware attacks, studies on the detection of IoT malware through feature learning and classification have been conducted [1]–[3], [5], [8], [16]–[27].

The associate editor coordinating the review of this manuscript and approving it for publication was Ana Lucila Sandoval Orozco.

Generally, studies divide the malware detection phase into analysis and detection phases. In the analysis phase, to extract a malware feature, the following methods are available: static analysis, dynamic analysis, hybrid analysis, and memory analysis. Static analysis analyzes information about binary files without directly executing malware, whereas dynamic analysis executes malware in a controlled environment such as a virtual machine (VM) or sandbox to analyze how the malware operates [2], [3], [5], [16]–[27]. Hybrid analysis utilizes both static and dynamic analyses [3], [17], [18], [20], [22]–[26]. Memory analysis is a comprehensive analysis method for malware in memory [22], [23]. Once the malware analysis is complete, the detection phase is performed to detect malware in the analyzed content. The detection phase can be done using signature-based, heuristic-based, specification-based, and cloud-based malware detection techniques. The signature-based malware detection technique detects malware by comparing signatures, which are characteristics of malware [1], [17], [18], [20]–[22], [24], [25], [27]. The heuristic-based malware detection technique predicts new and variant malware in advance based on malware features and patterns [1], [17], [18], [20]–[22], [24], [25], [27]. The specification-based malware detection technique determines malware based on specific rules, whereas the cloud-based malware detection technique detects malware through a cloud-server mode [1], [2], [17], [18], [24], [25]. By detecting known and new and variant IoT malware through the utilization of malware analysis and detection techniques, the propagation of malware into other IoT devices can be prevented.

However, IoT devices have limited hardware resources with systems optimized to perform specific purposes. Detecting new and variant IoT malware that is evolved intelligently and at a rapidly increasing pace in such devices is difficult. In addition, many constraints are followed to analyze the vast amount of behavior data generated by IoT malware in IoT devices and to detect them after training [2], [7], [11], [16], [23].

Thus, this paper proposes a dynamic analysis for IoT malware detection (DAIMD) that performs dynamic analysis on IoT malware in nested cloud-based VM environment and learns behavior images compressed with a vast amount of behavior data based on a convolution neural network (CNN) model. DAIMD performs dynamic analysis on malware in a nested virtual environment rather than an IoT device, so it is possible to accurately analyze and detect variant IoT malware that is obfuscated or whose code value is changed without limitation of hardware resources. In addition, various actions in memory, network, process, system call, and virtual file system are extracted to detect malware that perform malicious actions on embedded Linux-based IoT devices. Utilizing the vast amount of extracted action data to the maximum, it converts behavior data into images to analyze and detect malware. DAIMD, which detects IoT malware by training the generated behavior image on ZFNet, one of the CNN models, shows that it can accurately classify and detect IoT malware.

The remainder of this paper is organized as follows. Section 2 reviews existing research methods for analyzing and detecting IoT malware. Section 3 describes the DAIMD proposed in this paper. Section 4 builds a DAIMD model and Section 5 describes performance evaluation for the DAIMD model. Finally, Section 6 presents the conclusions of this paper and describes future research directions.

II. RELATED WORKS

Various studies utilizing static, dynamic, hybrid, and memory analysis methods have been conducted to analyze how malware works and how code flows prior to its detection. Most studies have employed a static analysis method that can check the overall malware structure without executing the malware. However, static analysis has difficulty detecting obfuscated malware using packing and identifying the overall functions of malware, which are drawbacks [16], [17], [19], [21]–[23], [26], [27].

To solve this, studies on dynamic analysis methods have been conducted to analyze overall functions of malware and to detect obfuscated malware as well as new and variant malware by executing it [28]–[30]. In addition, studies on techniques to transform feature data into images for malware detection by utilizing a large amount of feature data generated by malware have been conducted [31]–[34].

A. MALWARE DETECTION UTILIZING DYNAMIC ANALYSIS TECHINIQUE

Mohaisen *et al.* [28] proposed an automated and behavior-based malware analysis and labeling (AMAL) system to automatically analyze and classify malware behaviors. AMAL largely consists of AutoMal, which monitors behaviors of the file system, network, and registry, and MaLabel, which classifies similar malware by family based on the monitoring of extracted behaviors. MaLabel classifies specific malware families using the machine learning techniques support vector machine (SVM), decision tree (DT), and K-nearest neighbor (KNN) algorithms. However, the AMAL proposed by Mohaisen [28] has the difficulty of manually verifying by the malware analyst in the process of selecting and labeling the representative behavior of the malware.

Galal *et al.* [29] proposed a behavior analysis method of malware that collects information from application programming interface (API) calls and parameters used by malware through an API hooking technique. It infers unique malware behaviors in the API sequence generated from the extracted API calls and parameters. Although machine learning techniques such as DT, random forest (RF), and SVM algorithms were used to classify malware based on the inferred behaviors, the method had difficulty detecting malware, because the inference of malware behaviors involved the subjective intervention of analyzers.

Phode *et al.* [30] proposed a model to predict malware in execution files by setting the file execution time to a sec unit. The behavior data used to classify and detect malware were continuous data such as the total number of processes,

TABLE 1. Comparison between previous studies and DAIMD.

Related works	Environment	Analysis techniques	Machine learning / Deep learning	Visualization	Behavior features
Mohaisen *et al.* [28]	Windows	Dynamic analysis	Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbor (KNN)	X	System, network, registry
Galal *et al.* [29]	Windows	Dynamic analysis	DT, Random Forest (RF), SVM	X	API call
Phode *et al.* [30]	Windows	Dynamic analysis	Recurrent Neural Network (RNN)	X	No. of processes, maximum number of allocated process IDs, CPU usage, transmitted and received packet size, memory usage
Shaid *et al.* [31]	Windows	Dynamic analysis	-	O	API call
Trinius *et al.* [32]	Windows	Dynamic analysis	-	O	API call, process
Han *et al.* [32]	Windows	Dynamic analysis, static analysis	-	O	Opcode
Cui *et al.* [34]	Windows	Static analysis	Convolution Neural Network (CNN)	O	Binary code
DAIMD	Embedded Linux	Dynamic analysis	CNN	O	Memory, network, system call, virtual file system, process

the maximum number of allocated process IDs, or memory usage; which were trained by a recurrent neural network (RNN) to determine the presence of malware before the malware executed the payload, thereby protecting the system from malicious attacks.

B. MALWARE FEATURE DATA VISUALIZATION TECHNIQUE

Shaid *et al.* [31] proposed a malware behavior image technique that visualizes malware by mapping a color according to the malware intensity of API calls after capturing the calls from behavior data to emphasize the malicious acts of the variant malware. When the malware intensity of API calls is higher, warmer colors are used; cooler colors represent a lower malware intensity of calls.

Trinius *et al.* [32] proposed a treemaps and thread graphs to image malware behaviors to summarize and represent a large number of behavior record reports extracted from CWSandbox. The treemaps extract data about the frequency of API calls and operations performed by malware, conducting the visualization. By contrast, the thread graph converts the behavior data, where individual thread operations of processes are sequentially listed by time into images.

Han *et al.* [33] proposed a method to create images based on the opcode sequence extracted from the execution results of static and dynamic analyses on binary files. The method can measure the similarity between variant malware by comparing the RGB pixel information between the images generated from the binary files.

Cui [34] proposed a method to quickly detect variant malicious code by visualizing the malware through image processing technology. First, after converting the binary file for malware into a gray scale image, CNN was used to automatically extract the features of the generated image. In addition, a data equalization method was applied to the malware image by applying the bat algorithm to solve the overfitting problem caused by the number of different malware families. This malware detection method showed an excellent detection speed, and the accuracy was 94.5%.

The DAIMD proposed in this paper analyzes the overall functions of malware through dynamic analysis to detect well-known IoT malware as well as new and variant IoT malware and compresses and represents feature data by visualizing a large amount of this data. It selects representative features in images through a CNN model and trains them to analyze and detect malware, thereby avoiding the need for the subjective intervention of malware analyzers.

Table 1 summarizes the comparison of the proposed DAIMD with other models. The comparison items are set as follows: system environment constructed to detect malware, analysis technique used to extract behavior features, machine learning and deep learning used to classify malware, whether visualization is done to summarize and represent behavior features, and behavior feature type used to detect malware.

III. SCHEME OF DAIMD

The DAIMD scheme dynamically analyzes malware, which is a threat to IoT devices equipped with limited hardware

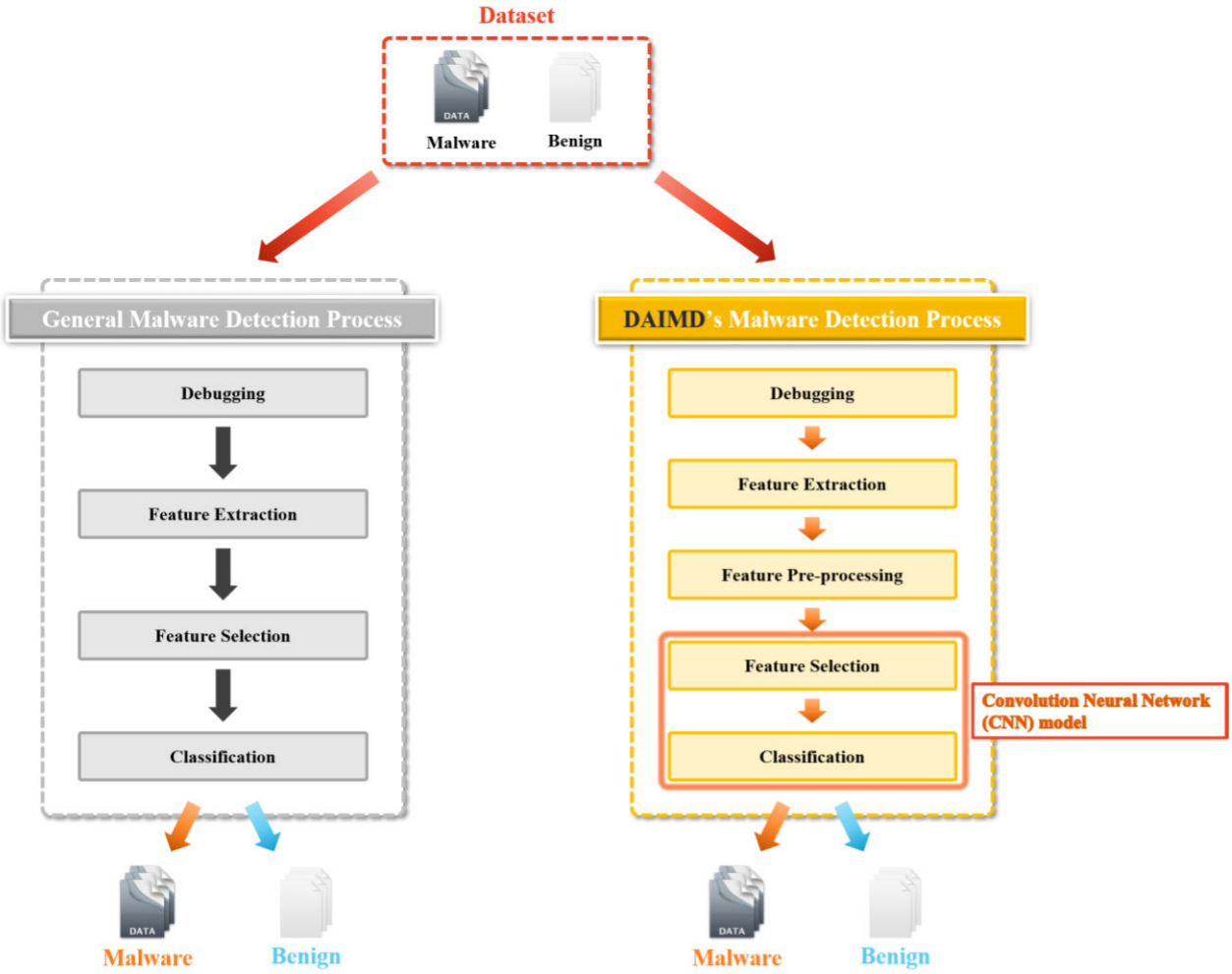


FIGURE 1. Comparison of the malware detection process in a general malware detection system and DAIMD.

resources, in a cloud-based nested virtual environment and trains them using the CNN model following specific analysis and detection processes similar to those used in general malware detection systems. The malware detection processes in a general malware detection system and DAIMD are shown in Fig. 1.

The process of a general malware detection system largely consists of debugging, feature extraction, feature selection, and classification. Debugging is performed with datasets consisting of malware and benign to generate a log file, from which behavior features are extracted. From among the extracted features, representative behaviors are selected; this is followed by a classification process based on representative behaviors. Then, malware features are trained to classify the files as containing malware or benign code [3], [17], [18], [20], [24].

By contrast, the process of DAIMD consists of debugging, feature extraction, feature pre-processing, feature selection, and classification, which are identical to the phase of debugging and feature extraction in a general malware detection system, but it also performs feature pre-processing to convert a vast amount of behavior features created after feature extraction into images. When the representative features are selected from the image generated through the feature pre-processing step and the learning process is performed individually, the subjective thinking of the malware analyst is involved. To prevent this, DAIMD uses the CNN model to perform feature selection and classification steps.

A. DEBUGGING

In a virtual environment where IoT malware and benign files are executed, code in assembly language is generated from binary files utilizing Interactive Disassembler (IDA) Pro analysis tools, and debugging is remotely performed on target files to identify how the code works and flows [35].

B. FEATURE EXTRACTION

The feature extraction phase extracts signatures, which are features of execution files from file behaviors and internal structures analyzed through debugging, from IoT malware

TABLE 2. Feature data types and configuration extracted through debugging.

Feature data type	Configuration	Description
Memory	PID	Process ID
	R/W	Whether read/write is performed on memory
	Byte	Memory size
	Address	Memory address value
	Data	Data stored in memory
Network	Index	Called execution order
	PID	Process ID
	S/R	Whether data are transmitted or received through network
	Target IP	Target IP address that receives data through network
	Local IP	Local IP address that transmits data through network
	Protocol	Network protocol used to transmit data
	Data	Content to be delivered through network
System call	Index	Called execution order
	PID	Process ID
	Type	Called system call type
	Name	Called system call name
	Data type	Data type stored in system call
	Data name	Data name stored in system call
	Data value	Data value stored in system call
Virtual file system (VFS)	Index	Called execution order
	PID	Process ID
	UID	User ID
	GID	Group ID
	R/W	Whether file read or write
	File name	File name
	File size	File size
	Path_data	Path where file is stored
Process	Index	Called execution order
	Process name	Called process name
	PID	Process ID
	PPID	Parent process ID
	Child PID	Child process ID

and benign files. The features extracted as signatures of files by DAIMD are memory, network, system call, virtual file system (VFS), and process, which are extracted as a.csv file format and stored in Excel. The feature data types and configuration that represent behaviors of execution files are presented in Table 2.

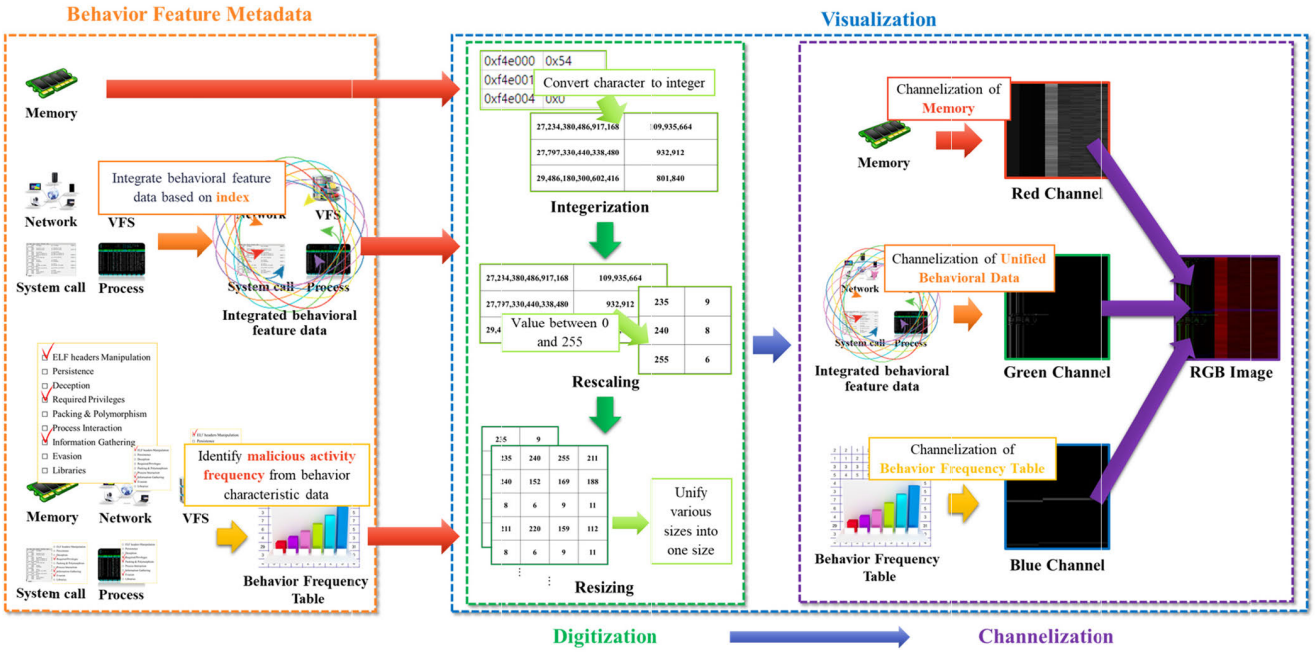


FIGURE 2. Feature pre-processing phase of DAIMD to visualize behavior data.

C. FEATURE PRE-PROCESSING

To represent the vast amount of feature data extracted from IoT malware and benign files during the feature extraction phase, the feature pre-processing phase compresses the feature data and converts it into an image type as shown in Fig. 2, which largely consists of behavior features metadata and visualization.

The behavior feature metadata phase processes feature data that represent execution file behaviors prior to converting the extracted feature data into an image, in which network, system call, VFS, and process that have an index of called order among the feature data are integrated into a single behavior datum, and a behavior frequency table that records the frequency of malicious activities from memory, network, system call, VFS, and process is created. Here, malicious activity refers to nine typical malicious activities that occur in Linux: Executable and Linkable Format (ELF) header manipulation, persistence mechanism, deception, required privilege, packing and polymorphism, information gathering, process interaction, library, and evasion [36].

In the final phase, visualization is performed using memory feature data, integrated behavior feature data, and a behavior frequency table generated from behavior feature metadata. The visualization phase is then divided into a digitization phase, in which strings in the behavior feature data are converted into vectors, and a channelization phase in which RGB channels are created based on the vectors. The digitization is performed in the following order: Integerization, which converts all strings in behavior feature data into integers; Rescaling, which adjusts the converted integer values into a value between 0 and 255, the range of the image; and Resizing, which generalizes a vector of behavior data whose size varies into a single size.

The rescaling process is adjusted to a value between 0 and 255 through (1). Here, V_k denotes a vector element in the kth behavior feature log data, and R_k denotes a vector element in the rescaled kth behavior feature log data.

$$R_k = Int\left(\frac{V_k}{Max(V_1, V_2, \cdots, V_k)}\right) \times 255 \qquad (1)$$

Vectors of the memory, integrated behavior feature data, and behavior frequency table generated through digitization are matched to red, green, and blue channels, respectively, during the channelization phase, and the channels are combined into a single RGB channel, thereby creating images about behaviors that are then stored.

D. FEATURE SELECTION AND CLASSIFICATION

DAIMD detects IoT malware by integrating feature selection and classification phases into a single phase using ZFNet, a CNN model, to select and train representative features of behaviors without human intervention. ZFNet is the CNN model that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2013 with an image recognition error rate of 11.2% [37].

The input datum in the CNN model is an RGB image generated through feature pre-processing, from which behavior features are detected and calculated in a matrix called a feature map. It performs a feature selection phase that extracts only behavior feature values whose size are larger than that of neighbors' behavior feature values by applying the max pooling technique to reduce the generated feature

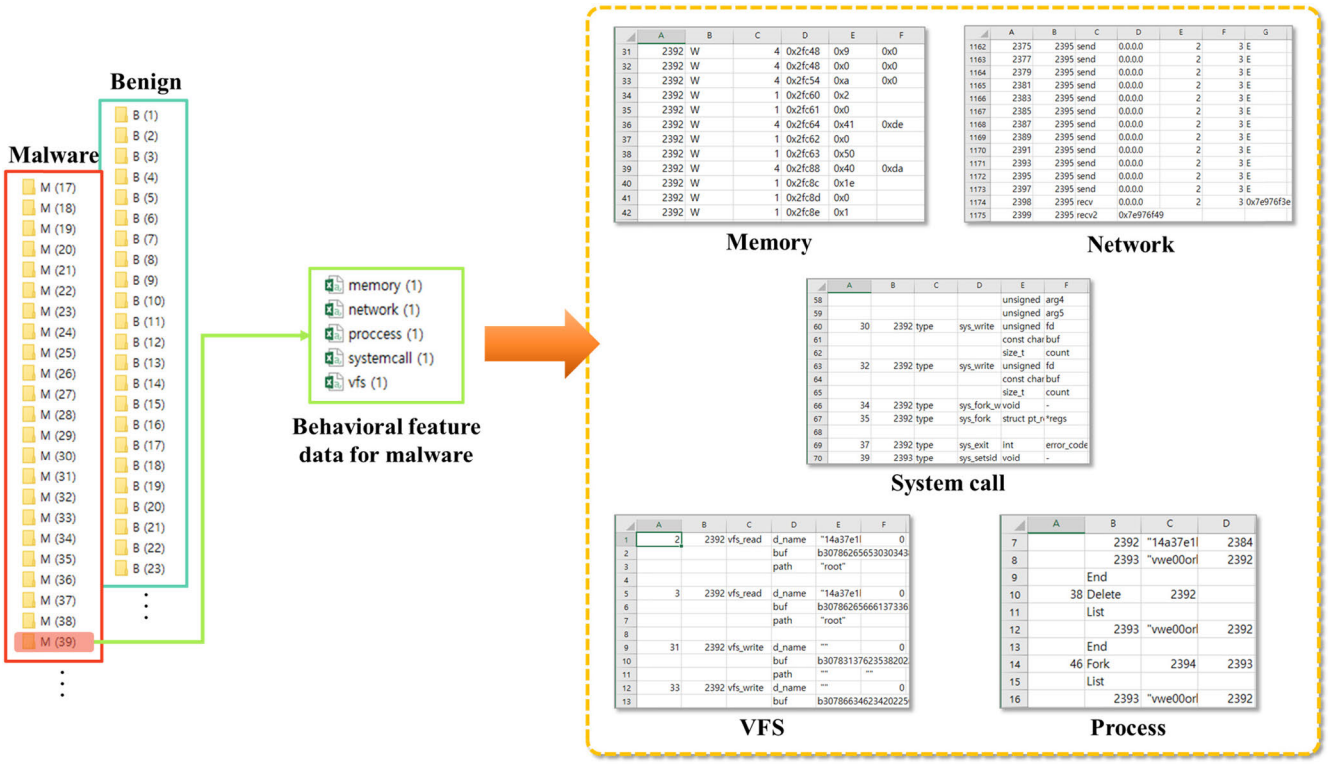


FIGURE 3. Five types of feature data extracted for DAIMD.

map dimensions. Finally, representative behavior features are trained through classification in the CNN model to perform classification and determination whether the dynamically analyzed execution file is malware or benign in nested virtual environment.

IV. DAIMD IMPLEMENTATION

To train and build the DAIMD model proposed in this paper to protect IoT devices from infection of IoT malware, we first built a cloud-based nested virtual environment equipped with Intel Core i7-9700K and GeForce RTX 2070. The analysis of execution files and malware detection were performed in a cloud environment to detect IoT malware that threaten intelligent attacks against IoT devices equipped with limited hardware resources. A VM was created in a cloud to prevent IoT malware from being propagated to major networks. An embedded Linux system with Advanced RISC Machines (ARM) processors was developed by utilizing the VM-based embedded software development verification solution (Imperas) in the VM.

A. DEBUGGING

The dataset was collected from IoT devices. 1,000 new and variant IoT malware samples and 401 benign files were run in the embedded system. From among them, a total of 840 files were used for a training dataset, to develop the DAIMD model by analyzing and training the behavior features of IoT malware. A total of 561 malware and benign files were used as a test dataset to test the DAIMD model.

To analyze and detect IoT malware based on 1,401 IoT malware and benign files (including both the training and test datasets), the files were executed for five min in nested cloud environment, and debugging on the executed files were conducted remotely.

B. FEATURE EXTRACTION

The feature data for memory, network, system call, VFS, and process were extracted as an Excel file format (Fig. 3) after monitoring the flow and how the code worked in the execution files through debugging.

C. FEATURE PRE-PROCESISNG

To convert the vast amount of behavior feature data generated during the feature extraction phase into images, feature pre-processing phase processed feature data by integrating behavioral feature data feature data into one based on the index and recording frequency of malicious activities to create integrated behavioral feature data and behavior frequency table. To efficiently analyze IoT malware, feature data for network, system call, VFS, and process were integrated into one integrated data based on an index that represented the call order, read as an Excel file format with up to 500 records and stored as shown in Fig. 4.

Digitization and channelization were performed to convert integrated behavioral feature data and the behavior frequency

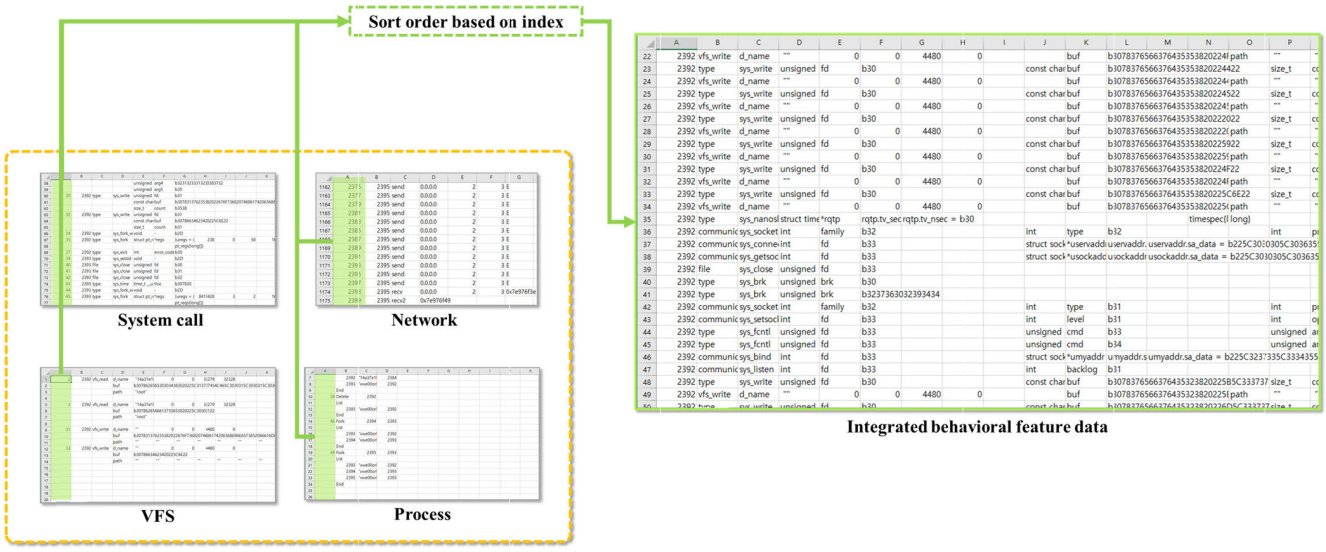


FIGURE 4. Feature data of integrated behaviors based on index.

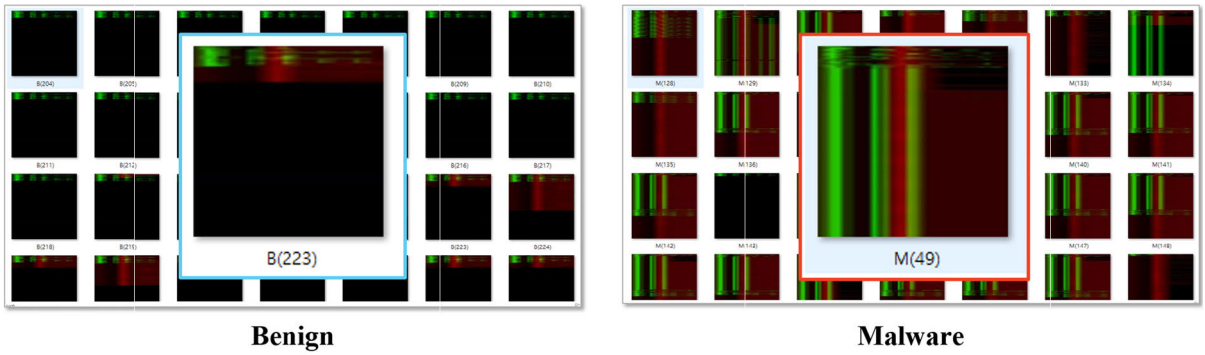


FIGURE 5. Benign and malware images created through feature pre-processing.

table generated from behavior feature metadata and memory, previously feature data, into images. To convert all strings in the behavior data into integers without crossing the range of int representation, up to nine characters were read and converted into an integer in the integerization process. In addition, a size of behavior data converted into a vector whose value ranged between 0 and 255 was generalized to 512 × 512, the maximum allowable size of cloud-based nested virtual environment's memory. To consider as many neighbor feature values in the vector as possible, bilinear interpolation was employed using a re-sampler.

Through the channelization process, a 512 × 512 sized vector created from the digitization of memory feature data, integrated behavioral feature data, and the behavior frequency table were matched to a red channel, green channel, and blue channel, respectively. Each channel, with its reduced dimensions of the feature data, was integrated into a single RGB image and stored. Fig. 5 shows the results when the feature pre-processing phase is complete, with the training and test datasets.

D. FEATURE SELECTION AND CLASSIFICATION

Table 3 below shows the ZFNet layer name, tensor size, and parameters used in this paper to train and build IoT malware detection model.

The DAIMD model detected IoT malware by training with feature images, using a total of 840 images in the training dataset using the ZFNet model. IoT malware detection accuracy was 99.87%, and the difference between actual data and output through the model was 0.0047.

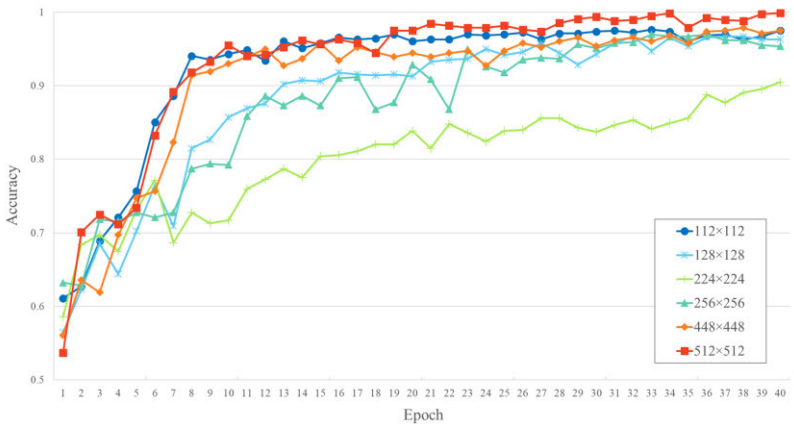
To verify whether the trained DAIMD model could accurately detect IoT malware, tests were conducted using 561 files in the test dataset, and detection accuracy was 99.28%.

V. PERORMANCE EVALUATION

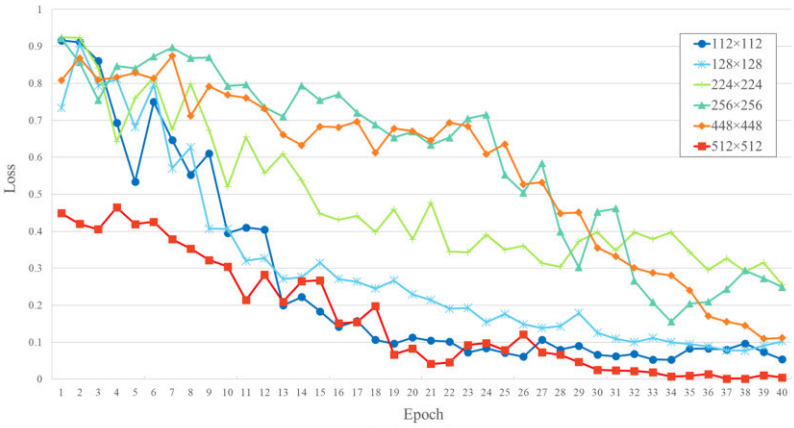
A performance evaluation was conducted to check whether the DAIMD proposed in this paper could accurately analyze and detect both well-known IoT malware as well as intelligently evolving new and variant IoT malware. The indices used to evaluate the performance of the implemented model

TABLE 3. Structure of ZFNet used to build IoT malware detection model.

Layer name	Tensor size	Parameter
Input Image	512×512×3	0
Conv2d	256×256×96	14208
MaxPool-1	128×128×96	0
Conv2d-2	32×32×256	614656
MaxPool-2	15×15×256	0
Conv2d-3	15×15×512	1180160
Conv2d-4	15×15×1024	4719616
Conv2d-5	15×15×512	4719104
MaxPool-3	8×8×512	0
Flatten	32768×1	0
Dense-1	4096×1	134221824
Dropout-1	4096×1	0
Dense-2	4096×1	16781312
Dropout-2	4096×1	0
Dense-3	1×1	4097
Activation	1×1	0



(a) Train accuracy



(b) Train loss

FIGURE 6. Train accuracy and loss for various vector sizes.

were false positive rate (FPR), false negative rate (FNR), and accuracy (ACY).

FPR refers to the rate that IoT malware is falsely classified as benign. It can be calculated using (2). Here, *false positive (FP)* means the number of cases that are classified as benign files, even though they are IoT malware, and *true negative (TN)* means the number of cases correctly classified as benign.

$$FPR = \frac{FP}{(FP + TN)} \tag{2}$$

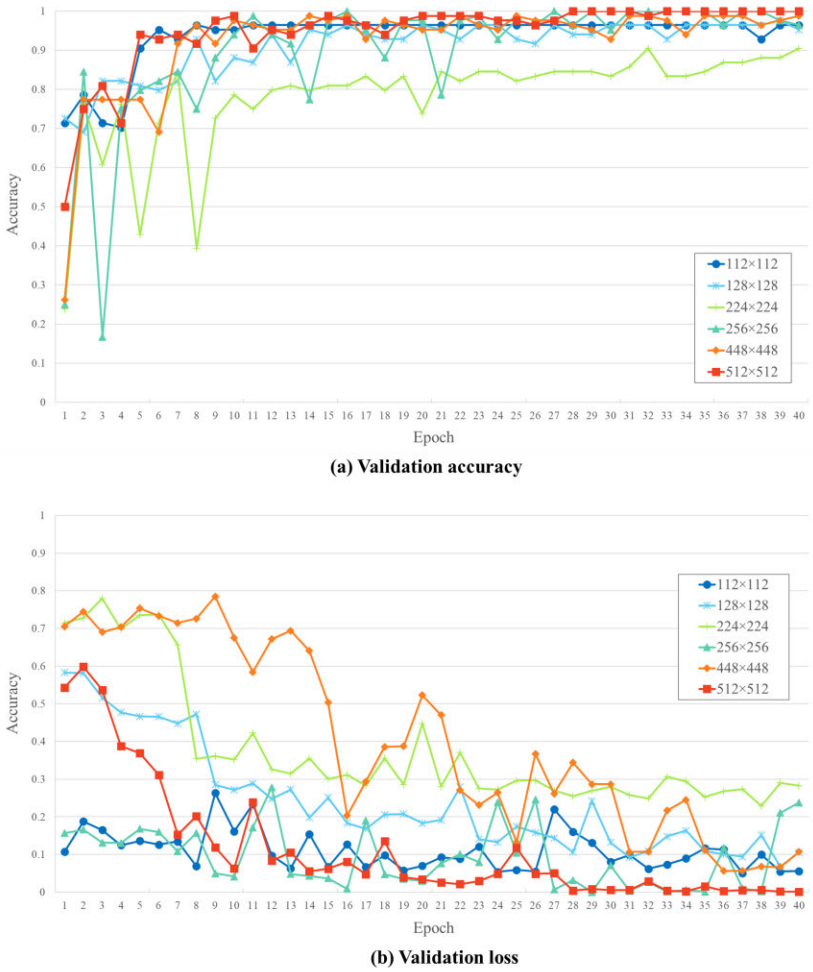


FIGURE 7. Validation accuracy and loss for various vector sizes.

FNR refers to the rate that benign files are classified as malware. It can be calculated using (3). *False negative (FN)* means the number of cases that are incorrectly classified as IoT malware, even if they are benign files, and *true positive (TP)* means the number of cases correctly classified as malware.

$$FNR = \frac{\text{FN}}{(TP + FN)} \tag{3}$$

Finally, *ACY* refers to how accurately malware and benign files are classified; it can be calculated using (4).

$$ACY = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{4}$$

Fig. 6 shows the change in training accuracy and loss values of the DAIMD model, in which feature data of behaviors according to various vector sizes are trained. In training behavior images in the ZFNet model for detecting IoT malware based on feature vectors with optimum size, the Adam optimizer algorithm was used to perform optimization; training of the model was conducted by fixing the epoch to 40. Fig. 6 (a) shows the accuracy measured as a result of training on the DAIMD model. As the epoch increases, the accuracy gradually increases. When vector size is 224 × 224, it shows relatively low accuracy compared to other vector sizes. Fig. 6 (b) shows the loss value, and the loss curve decreases relatively slowly when it has a vector size of 512 × 512.

To conduct a validation of DAIMD, 84 records in the training dataset, which accounted for 10% of the total number of records, were used. Whether an implemented model can exhibit optimal performance can be verified through validation. Fig. 7 shows the accuracy and loss values when validation was performed on DAIMD models with various vector sizes. In Fig. 7 (a), as the epoch increases, the validation accuracy of a vector with a size of 512 × 512 is gradually increasing compared to other sized vectors, whereas a vector with a validation accuracy at a size of 224 × 224 is significantly lower than the field. In addition, overfitting occurred in a specific section of vectors with sizes of 224 × 224 and 256 × 256. Fig. 7 (b) shows the measured loss values when validation was performed for various vector sizes. While the loss curve is decreasing when the size is 512 × 512, the vector of 448 × 448 size vibrates greatly within a specific range, indicating that overfitting occurs.

TABLE 4. Comparison of test accuracy, FPR, and FNR results for various vector sizes.

Vector size	Test accuracy (%)	FPR (%)	FNR (%)
112×112	96.96	1.27	3.71
128×128	93.4	15.38	2.37
224×224	90.01	23.73	2.47
256×256	94.47	2.58	6.65
448×448	98.03	5.05	0.52
512×512	99.28	0.63	0.74

TABLE 5. Performance comparison between DAIMD and other malware detection models.

Related works	Environment	Analysis techniques	Accuracy (ACY, %)	FPR (%)
Shaid *et al.* [31]	Windows	Dynamic analysis	98.4	3
Yang *et al.* [38]	Android	Static analysis	95.51	4.6
Zhou [39]	Windows	Hybrid analysis	97.3	-
Kumar *et al.* [40]	Android	Android	98	1.8
DAIMD	Embedded Linux	Dynamic analysis	99.28	0.63

When testing is performed on the optimal DAIMD model selected through training and validation, accuracy, FPR, and FNR can be represented as shown in Table 4. When testing the DAIMD model with a feature vector with a size of 512 × 512, the test accuracy was 99.28%, and it was confirmed that the accuracy was higher than when testing with a feature vector of another size. It can be seen that when the size of the vector is 512 × 512, the value of FPR, which is an IoT malicious code, but is classified as a file that is not malicious, is significantly lower than that of other sizes. On the other hand, FNR was 0.52% when the vector size was 448 × 448, lower than that with the 512 × 512 vector size. Because the most important index in detecting IoT malware is FPR, this paper also places great emphasis on the performance evaluation of the developed model with FPR.

In Fig. 6, 7, and Table 4, the performance and accuracy of FPR values when training, validation, and testing of the DAIMD model with 512 × 512 size vectors showed better performance than those performed with vectors of different sizes.

Table 5 presents a comparison between the proposed model and other malware detection models. To evaluate the performance of the models, the following comparative items were designated: under which environment malware was detected; what analysis technique was used; and test accuracy and FPR values. The FPR value for Zhou's model [39] was not included, because it was not measured.

All five types of malware detection models analyzed and detected malware using various analysis techniques in different environments. Overall, the accuracy values of the models showed excellent detection performance above 95%. Among them, the accuracy of the DAIMD model (99.28%) was best, and the probability of it incorrectly classifying malware as benign (0.63%) was the lowest. This means that the DAIMD model can accurately detect variant malware that threatens IoT devices.

VI. CONCLUSION

IoT devices in embedded Linux environments configured with various architectures and libraries are particularly targeted by malware authors, because their attack points are wide, and many vulnerable products are distributed in the market. Most IoT devices are equipped with ARM processors, so the number of IoT malware samples targeting IoT devices with them has increased. To address this, numerous studies have been conducted to analyze and detect IoT malware. However, most studies have involved subjective intervention by malware analyzers to select and classify

representative malware behaviors. This approach makes it potentially difficult to accurately detect new and variant IoT malware as it is intelligently evolved. In addition, the vast amount of behavior data that must be extracted to detect IoT malware in embedded Linux-based IoT devices, equipped with limited hardware resources designed for specific purposes, are difficult to accumulate and store for long periods of time.

Thus, this paper proposed a dynamic analysis for IoT malware detection (DAIMD) scheme that analyzed IoT malware dynamically and trained for and classified IoT malware using the CNN model in a cloud environment under a virtual embedded system. First, cloud-based nested virtual environment was designed and implemented to analyze and detect IoT malware in a safe environment. Then, the DAIMD model was created by performing training, validation, and testing according to the following phases: debugging, feature extraction, feature pre-processing, feature selection, and classification in the cloud environment. Since the feature data of the behaviors extracted through the detection process were numerous, they were converted to images to prevent a complex computation problem for training and classification of the feature data in the classification phase, reducing the number of dimensions of the data. In addition, the features of IoT malware and benign files were comprehensively represented through the DAIMD visualization technique.

The infection of IoT devices or the propagation of IoT malware to other IoT devices connected through the Internet can be prevented using DAIMD. Furthermore, because the DAIMD selects and classifies behavior features using the CNN model without human subjective intervention, new and variant IoT malware with various intelligent attack techniques can be accurately detected.

The DAIMD proposed in this paper analyzed behavior features by executing IoT malware using a dynamic analysis technique. Because some IoT malware can easily recognize that they are executed in a limited environment such as a VM, they may avoid malware analysis and detection systems that use the dynamic analysis technique. Thus, a study on the implementation of a model that can detect IoT malware using the hybrid analysis technique, which analyzes malware by utilizing both static and dynamic techniques, will be conducted in the future.

REFERENCES

[1] H. Sun, X. Wang, R. Buyya, and J. Su, "CloudEyes: Cloud-based malware detection with reversible sketch for resource-constrained Internet of Things (IoT) devices," *Softw., Pract. Exper.*, vol. 47, no. 3, pp. 421–441, Mar. 2017.

[2] M. Noor, H. Abbas, and W. B. Shahid, "Countering cyber threats for industrial applications: An automated approach for malware evasion detection and analysis," *J. Netw. Comput. Appl.*, vol. 103, pp. 249–261, Feb. 2018.

[3] S. Sharmeen, S. Huda, J. H. Abawajy, W. N. Ismail, and M. M. Hassan, "Malware threats and detection for industrial mobile-IoT networks," *IEEE Access*, vol. 6, pp. 15941–15957, 2018.

[4] O. A. Waraga, M. Bettayeb, Q. Nasir, and M. A. Talib, "Design and implementation of automated IoT security testbed," *Comput. Secur.*, vol. 88, pp. 1–17, Jan. 2020.

[5] R. Kumar, X. Zhang, R. U. Khan, and A. Sharif, "Research on data mining of permission-induced risk for Android IoT devices," *Appl. Sci.*, vol. 9, no. 2, pp. 1–22, Jan. 2019.

[6] P. K. Sharma, J. H. Park, Y.-S. Jeong, and J. H. Park, "SHSec: SDN based secure smart home network architecture for Internet of Things," *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 913–924, Jun. 2019.

[7] Y.-S. Jeong and J. H. Park, "IoT and smart city technology: Challenges, opportunities, and solutions," *J. Inf. Process. Syst.*, vol. 15, no. 2, pp. 233–238, Apr. 2019.

[8] T. Lei, Z. Qin, Z. Wang, Q. Li, and D. Ye, "EveDroid: Event-aware Android malware detection against model degrading for IoT devices," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6668–6680, Aug. 2019.

[9] K. Gafurov and T.-M. Chung, "Comprehensive survey on Internet of Things, architecture, security aspects, applications, related technologies, economic perspective, and future directions," *J. Inf. Process. Syst.*, vol. 15, no. 4, pp. 797–819, Aug. 2019.

[10] S.-Y. Choi, C. G. Lim, and Y.-M. Kim, "Automated link tracing for classification of malicious Websites in malware distribution networks," *J. Inf. Process. Syst.*, vol. 15, no. 1, pp. 100–115, Feb. 2019.

[11] N. Y. Kim, S. Rathore, J. H. Ryu, J. H. Park, and J. H. Park, "A survey on cyber physical system security for IoT: Issues, challenges, threats, solutions," *J. Inf. Process. Syst.*, vol. 14, no. 6, pp. 1361–1384, Dec. 2018.

[12] A. Nieto and R. Rios, "Cybersecurity profiles based on human-centric IoT devices," *Hum.-Centric Comput. Inf. Sci.*, vol. 9, no. 1, pp. 1–23, Nov. 2019.

[13] T. A. Alghamdi, "Convolutional technique for enhancing security in wireless sensor networks against malicious nodes," *Hum.-Centric Comput. Inf. Sci.*, vol. 9, no. 1, pp. 1–10, Oct. 2019.

[14] P. K. Sharma, J. H. Ryu, K. Y. Park, J. H. Park, and J. H. Park, "Li-Fi based on security cloud framework for future IT environment," *Hum.-Centric Comput. Inf. Sci.*, vol. 8, no. 1, pp. 1–13, Aug. 2018.

[15] M. Kuzin, Y. Shmelev, and V. Kuskov. (Sep. 18, 2018). New trends in the world of IoT threats. Kaspersky. [Online]. Available: https://securelist.com/new-trends-in-the-world-of-iot-threats/87991/

[16] J. Kang, S. Jang, S. Li, Y.-S. Jeong, and Y. Sung, "Long short-term memory-based malware classification method for information security," *Comput. Electr. Eng.*, vol. 77, pp. 366–375, Jul. 2019.

[17] 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, Jan. 2018.

[18] R. Tahir, "A study on malware and malware detection techniques," *Int. J. Eng. Educ.*, vol. 8, no. 2, pp. 20–30, Mar. 2018.

[19] B. Yu, Y. Fang, Q. Yang, Y. Tang, and L. Liu, "A survey of malware behavior description and analysis," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 5, pp. 583–603, May 2018.

[20] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Comput. Surv.*, vol. 44, no. 2, pp. 1–49, Feb. 2012.

[21] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," presented at the 5th Conf. Inf. Knowl. Technol., Shiraz, Iran, May 2013.

[22] R. Sihwail, K. Omar, and K. A. Z. Ariffin, "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, nos. 2–4, pp. 1662–1671, 2018.

[23] C.-W. Tien, J.-W. Liao, S.-C. Chang, and S.-Y. Kuo, "Memory forensics using virtual machine introspection for malware analysis," presented at the IEEE Conf. Depend. Secure Comput., Taipei, Taiwan, Aug. 2017.

[24] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 1–40, Oct. 2017.

[25] J. Landage and M. P. Wankhade, "Malware and malware detection techniques: A survey," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 61–68, Dec. 2013.

[26] S. Wu, P. Wang, X. Li, and Y. Zhang, "Effective detection of Android malware based on the usage of data flow APIs and machine learning," *Inf. Softw. Technol.*, vol. 75, pp. 17–25, Jul. 2016.

[27] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Comput. Secur.*, vol. 81, pp. 123–147, Mar. 2019.

[28] A. Mohaisen, O. Alrawi, and M. Mohaisen, "AMAL: High-fidelity, behavior-based automated malware analysis and classification," *Comput. Secur.*, vol. 52, pp. 251–266, Jul. 2015.

[29] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection," *J. Comput. Virol. Hacking Techn.*, vol. 12, no. 2, pp. 59–67, May 2016.
[30] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *Comput. Secur.*, vol. 77, pp. 578–594, Aug. 2018.
[31] S. Z. M. Shaid and M. A. Maarof, "Malware behaviour visualization," *J. Teknol.*, vol. 70, no. 5, pp. 25–33, Sep. 2014.
[32] P. Trinius, T. Holz, J. Gobel, and F. C. Freiling, "Visual analysis of malware behavior using treemaps and thread graphs," presented at the 6th Int. Workshop Vis. Cyber Secur., Atlantic City, NJ, USA, Oct. 2009.
[33] K. Han, B. Kang, and E. G. Im, "Malware analysis using visualized image matrices," *Sci. World J.*, vol. 2014, pp. 1–15, Jul. 2014.
[34] 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.
[35] IDA: About, Liege, Belgium. *Hex-Rays*. Accessed: Dec. 13, 2019. [Online]. Available: https://www.hex-rays.com/products/ida/
[36] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding Linux malware," presented at the 39th IEEE Symp. Secur. Privacy, San Francisco, CA, USA, May 2018.
[37] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," presented at the 13th Eur. Conf. Comput. Vis., Zurich, Switzerland, Sep. 2014.
[38] M. Yang and Q. Wen, "Detecting Android malware by applying classification techniques on images patterns," presented at the IEEE 2nd Int. Cloud Comput. Big Data Anal., Chengdu, China, Apr. 2017.
[39] H. Zhou, "Malware detection with neural network using combined features," presented at the 15th Int. Annu. Conf. Cyber Secur., Beijing, China, Aug. 2018.
[40] R. Kumar, X. Zhang, W. Wang, R. U. Khan, J. Kumar, and A. Sharif, "A multimodal malware detection technique for Android IoT devices using various features," *IEEE Access*, vol. 7, pp. 64411–64430, 2019.

JUEUN JEON received the B.S. and M.S. degrees in multimedia engineering from Dongguk University, South Korea, in 2018 and 2020, respectively, where she is currently pursuing the Ph.D. degree with the Department of Multimedia Engineering. Her current research interests include information security for cloud computing and the Internet of Things (IoT).

JONG HYUK (JAMES J.) PARK (Member, IEEE) received the Ph.D. degree from the Graduate School of Information Security, Korea University, South Korea, and the Ph.D. degree from the Graduate School of Human Sciences, Waseda University, Japan. He has served as a Research Scientist at the Research and Development Institute, Hanwha S&C Company, Ltd., South Korea, from December 2002 to July 2007. He was a Professor with the Department of Computer Science and Engineering, Kyungnam University, South Korea, from September 2007 to August 2009. He is currently employed as a Professor with the Department of Computer Science and Engineering and the Department of Interdisciplinary Bio IT Materials, Seoul National University of Science and Technology (SeoulTech), South Korea. has published about 200 research articles in international journals and conferences. His research interests include security and digital forensics, human-centric ubiquitous computing, context awareness, and multimedia services. Also, his research interests include human-centric ubiquitous computing, vehicular cloud computing, information security, digital forensics, secure communications, multimedia computing, and so on. Dr. Park is a member of the IEEE Computer Society, KIPS, and KMMS. He received best paper awards from the ISA08 and ITCS-11 conferences and outstanding leadership awards from IEEE HPCC09, ICA3PP-10, IEE ISPA-11, and PDCAT-11. Furthermore, he received an outstanding research award from SeoulTech, in 2014. He has also served as the chair, the program committee chair or organizing committee chair at many international conferences and workshops. He is a Founding Steering Chair of various international conferences, including MUE, FutureTech, CSA, UCAWSN, and so on. He is employed as the Editor-in-Chief of *Human-Centric Computing and Information Sciences* (HCIS) by Springer, the *Journal of Information Processing Systems* (JIPS) by KIPS, and the *Journal of Convergence* (JoC) by KIPS CSWRG. He is also the Associate Editor or Editor of 14 international journals, including eight journals indexed by SCI(E). In addition, he has been employed as a Guest Editor for various international journals by such publishers as Springer, Elsevier, Wiley, Oxford University Press, Hindawi, Emerald, and Inderscience.

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. He was a Professor with the Department of Computer Engineering, Wonkwang University, South Korea, from 1993 to 2012. He worked and conducted research with the Michigan State University and Wayne State University in his capacity as a Visiting Professor, in 1997 and 2004, respectively. He is currently with the Department of Multimedia Engineering, Dongguk University, South Korea. His research interests include multimedia cloud computing, information security for cloud computing, mobile computing, the Internet of Things (IoT), and wireless sensor network applications. Prof. Jeong is also an Executive Editor of the *Journal of Information Processing Systems*, an Associate Editor of the *Journal of Supercomputing* (JoS), an Editor of the *Journal of Internet Technology* (JIT), and an Associate Editor of the *Journal of Human-centric Computing* (HCIS). He has been employed as a Guest Editor for various international journals by publishers, including Springer, Elsevier, John Wiley, Oxford University Press, Hindawi, Emerald, and Inderscience.