Network Malware Detection Using Deep Learning Network Analysis

Peng Xiao

Information Center of Yunnan Power Grid Co., Ltd., Kunming, 650000, Yunnan, China
E-mail: xiaopeng202112@163.com

Received 02 December 2022; Accepted 08 February 2023;
Publication 05 December 2023

Abstract

Malware, short for malicious software, is designed for harmful purposes and threatens network security because it can propagate without human interaction by exploiting user's vulnerabilities and carelessness. Having your system regularly scanned for malicious software is essential for keeping hackers at bay and avoiding the disclosure of sensitive data. The major drawbacks are the rapid creation of new malware variants, and it may become difficult to detect existing threats. With the ever-increasing volume of Android malware, the sophistication with which it can hide, and the potentially enormous value of data assets stored on Android devices, detecting or classifying Android malware is a big data problem. Security researchers have developed various malware detection and prevention programs for servers, gateways, user workstations, and mobile devices. Some offer centralized monitoring for malware detection software deployed on many systems or computers. The purpose of this essay is to critically examine the research that has been done specifically on malware detection. This paper proposes the Anti-Virus Software Detection for Malware with Deep Learning Network (AVSD-MDLN) framework to explore the possible threats. The two methods help in finding the threats.

Journal of Cyber Security and Mobility, Vol. 13_1, 27–52.
doi: 10.13052/jcsm2245-1439.1312

Dynamic Analysis for the Detection of Spyware (DA-DS) framework is framed to detect malicious malware, while the other is for classifying Android malware which is helped out through the Category in an Ensemble (CE) method. Prior malware detection methods are compared with the results of the proposed method. According to the research findings, the proposed approach achieves a higher projected time (0.5 sec) and detection accuracy (97.47%) than the existing situation machine learning and deep learning methodologies. Performance, correlation coefficient, and recall rate all improved in the suggested framework. Likewise, the negative rate (MPR) and the positive rate (PPR) also improved.

Keywords: Malware detection, spyware, antivirus, deep learning, dynamic analysis.

1 An Overview of Malware Detection

Devices are increasingly interconnected and sharing data; however, some companies mass-produce insecure devices and sell them to end users [1]. Thus, millions of Android apps can be obtained by users from numerous app stores and installed on their devices [2]. Malware authors will likely focus on IoT devices if they become widely available on the market [3]. It is possible for malware to rapidly spread to other networks by leaking user information collected by devices and penetrating significant networks [4]. The number of malicious Android apps has increased dramatically alongside the popularity of Android devices and operating systems [5]. With the development of new computer systems, cyberattacks have progressed [6]. These threats include novel malware attacks that compromise Android device security and are difficult to detect [7].

The Android operating system has a built-in module that can grant the necessary permissions to Android apps if there has been no security policy violation [8]. The dataset includes four types of malware: adware, ransomware, scareware, and SMS [9]. Because malware is constantly growing and changing, many methods for finding and stopping it have been proposed [10]. Researchers have reported two methods for spotting malware [11]. There is Static analysis, in which applications are examined without running them, and Dynamic analysis, in which malware behavior is reviewed in a sandbox after it has run [12].

Furthermore, the sophistication and multi-stage nature of modern cyberattacks has increased [13]. Large-scale property damage and monetary

losses resulted from these kinds of attacks [14]. As most services have moved to online and remote mode, the risk of cyberattacks and viruses has increased, and the COVID-19 pandemic has offered various cybersecurity challenges [15]. Research on detecting IoT malware by feature learning and categorization has been spurred by a need to shield IoT devices from novel and variant malware assaults [16].

Most research separates the process of finding malware into analysis and detection phases [17]. Static analysis, dynamic analysis, hybrid analysis, and memory analysis are some methods that can detect malicious software [18]. Unlike dynamic analysis, which runs malware in a sandbox or virtual computer to observe its behavior, static analysis examines the information of binary files [19]. Although static and dynamic analyses are used in hybrid analysis, memory analysis is the most thorough method for analyzing malware in memory [20]. Using the document tree as input, user agents may use the Visual Formatting Model to format the data appropriately for visual presentation. File names, hashes, strings like IP addresses, domain names, and file header data may all be utilized as technical indications to detect whether or not a file is malicious. Statistical study of system modifications or probabilistic explanation of an executable being malware based on the occurrence of literals is the foundations of traditional malware analysis techniques. However, this probabilistic or statistical technique only approximates a small subset of malware characteristics and has trouble handling disguised threats. The dataset was short, and the treatment of packed executables was disregarded, both of which increase the unpredictability of the outcomes should the system be deployed in real-time. a detection system that doesn't care about where the executable was called from or how it was called, and hence misses suspicious behaviour.

After malware analysis is complete, malware detection can begin. Several strategies, such as biometrics, memory-based and model, specification-based, and internet methods, can be employed at the detection stage [21]. Signature-based malware detection identifies malicious software by comparing known malware samples' signatures or characteristics [22]. There is growing consensus that Android malware detection and classification could benefit greatly from machine learning (ML) applications [23]. Many ML techniques have been proposed, human ingenuity is still essential in developing these tools and methods [24]. However, data integrity might be compromised throughout the process of choosing useful data for a study. The main goal of data selection is to find the best combination of data type, source, and instrument to answer research questions.

The significant contributions to this paper are as follows:

- A framework called Anti-Virus Software Malware Detection with a Deep Learning Network (AVSD-MDLN) is presented to reveal the current threats in the environment.
- One framework designed to identify spyware is the Dynamic Analysis for the Detection of Spyware (DA-DS) framework.
- Category in an Ensemble (CE) is a strategy that helps classify Android malware.

Precisely, this document will follow the following structure: Section 2 discusses the research in the broader context, and Section 3 provides a model for foreseeing the current threat and malware detection. The findings and discussion can be found in Section 4, while the endnotes and recommendations can be found in Section 5.

2 Background Literature

Phone conversations are just one function of smartphones; they may be used as portable data banks for text messages, photos, videos, documents, and more. Because of their popularity and the sensitive information they store, fraudsters see Android smartphones as a prime opportunity to steal personal information. Research on the visualization of malware's features is ongoing to improve classification precision, performance, and efficiency while minimizing expenses.

Imtiaz, S. I et al. [25]. Proposed DeepAMD [DAMD], a game-changing method that employs deep Artificial Neural Networks (ANNs) to defend against actual Android malware. It is done in need of effective means of identifying malicious software before exploiting sensitive information. It compares favorably to traditional machine learning classifiers and new research that pushes the boundaries of existing knowledge regarding efficiency measures like recall, f-score, and precision.

Damaševičius, R et al. [26]. Suggest a method for detecting malware based on Ensemble Classification [EC]. A stacked ensemble of dense (fully connected) and convolutional neural networks (CNN) do the initial classification step, while a meta-learner handles the final stage. An ensemble of five dense and CNN neural networks, with the extra trees classifier acting as a meta-learner, produces the best results.

Naeem H et al. [27]. Developed a framework for Malware Detection in the IIoT (MD-IIOT). For this reason, protecting the millions of users of

IIoT from cybercriminals necessitates a clever protection strategy. Malware analysis is made more accessible with the help of a proposed method that uses color picture visualization and a deep convolution neural network. Past malware detection methods are compared with the results of the suggested approach.

Qiu, J et al. [28]. Introduced DL-based Android Malware Detection and Classification (DL-AMDC), which seeks to solve the problems by analyzing how far it is with deep learning methods for identifying and categorizing Android malware. Choice of DL architecture, feature extraction and processing, performance measurement, and even acquiring appropriate high-quality data provide difficulties for academics and practitioners. This research aims to expose state of the art in malware detection on Android by modeling code semantics. Consequently, offer our perspectives on the potential and threats facing this new area of study and our predictions for where it will go in the future.

Jeon, J et al. [29]. Framed Dynamic Analysis for IoT Malware Detection (DA-IMD), harm done to the Internet of Things gadgets can be mitigated with the use of a technique that detects both well-known and newly-evolved IoT malware. DAIMD can dynamically evaluate IoT malware in a nested cloud environment because of the CNN model it employs. DAIMD performs dynamic analysis of IoT malware in a nested cloud, extracting features such as memory, network, virtual file system, process, and system call.

DAMD, EC, MD-IIOT, DL-AMDC, and DA-IMD are just a few of the well-known models that could benefit from refinement. One solution to address the need for better models is using a deep learning network to detect malware through anti-virus software. The other two approaches to identifying spyware and aiding in the classification of Android malware make the development of valuable methods far more efficient, accurate, predictable, and successful than is currently possible. When used to scan all incoming data, AVSD-MDLN may assist stop malware infections before it start. In addition to identifying sophisticated malware, AVSD-MDLN software can defend users against ransomware. A database server may not detect file corruption until it attempts to retrieve corrupted data. Be sure that database is still accurate every so often. Data may be checked for accuracy in a number of ways, including checksums, index data, and the presence of objects that should be associated with each page of a table. To further boost efficiency, one may also use an express validation that skips referential integrity checks. There has been no validation of the current CPS on the dataset.

3 Anti-Virus Software Detection for Malware with Deep Learning Network

This paper uses the help of a Deep Neural Network to detect malware through anti-virus software. The Anti-Virus Software Detection for Malware dynamically analyzes malware using nested virtualization in the cloud environments and then trains them using the deep learning network following detailed examination and detection techniques comparable to those employed by generic malware detection systems.

The above Figure 1 shows the system for detecting malware through a deep learning network. The dataset includes malware and benign code, a kind of executable file. Debugging, feature extraction, feature pre-processing, feature selection, and classification are the primary operations of a malware detection system's workflow. Consequently, debugging is performed on datasets that include both malicious and benign files to produce a log file from which behavioral data can be retrieved. Hence to classify the extracted features, expected behaviors are chosen from among them. The files are then classified as either malicious or benign using malware features that have been trained and employ a model based on a deep neural network to carry out tasks such as feature selection and classification. CNNs can reliably identify and categorise malware by examining a training set of PE files. Bytes sequence, grayscale images, API call sequence, structural entropy, and HTTP traffic are just some of the data types that CNNs can be used to analyse in malware.

Debugging:

Analyzing tools translate binary files into assembly language, and remote debugging helps understand the code's logic and flow. It is done in a simulated setting where both malicious and safe files can be run.

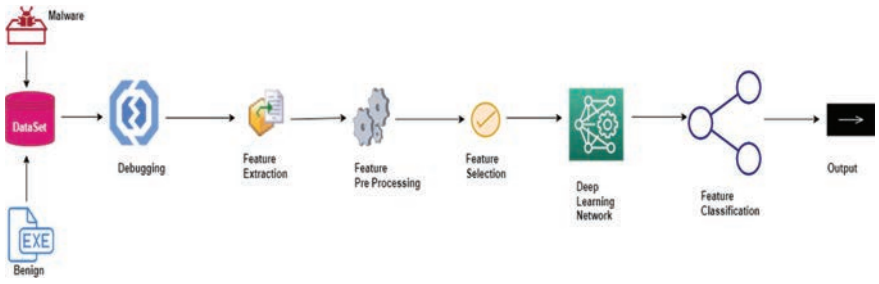

Figure 1 System for detecting malware.

Feature Extraction:

Malware memory, network, system call, VFS, and process signatures are extracted as.csv files and stored in Excel during the feature extraction phase. A file's signature is a characteristic explored about its execution by analyzing its behavior and internal structure during debugging. Through the use of feature selection methodologies during training, the suggested framework was able to effectively identify malware in real-world applications. In addition, the model construction is aided by the specified characteristics when several machine learning techniques are taken into account. The AVSD-MDLN framework is educated using various machine learning methods that are LSTM and SVM, and it operates on the idea of feature selection techniques. We take into account two different feature selection methodologies, the feature ranking strategy and the feature subset selection approach, to choose the optimal features for training.

Feature Pre-Processing:

The term "data preprocessing" refers to the process of transforming a messy data collection into a usable one. The data on malicious and safe files' features are compressed and converted during the feature pre-processing stage into an image type for more accessible representation, most of which is made up of metadata concerning behavior and visual terms of that data.

Feature Selection:

Feature Selection is a technique for streamlining the model's inputs by keeping just the most pertinent information and discarding irrelevant details, or noise. The ZFNet CNN model automatically picks and trains representative aspects of behavior; AVSD-MDLN can detect malware without human intervention in either the feature selection or classification phases.

Deep Learning Network:

For this, a deep neural network (DNN) or deep net will refer to any neural network with two or more hidden layers. Complex mathematical modeling is at the heart of how deep neural networks process data in various ways.

$$D = \emptyset(y_1, \ldots, y_n) \tag{1}$$

where D is the output $y_1 \ldots y_n$ are the network's input. $\emptyset$ is the precision factor. The network learns a map of input space y to space D as output in

Equation (1).

$$y_i = A\left[\sum_{i=1}^{N} y_j * k_{ij} + b_j\right] \tag{2}$$

The above Equation (2) shows the activation map of the network where k is the kernel, j is the size of the kernel, N is the inputs count, b, y_j is the bias kernel, A is the activation function.

$$\emptyset_j^l = \emptyset_j^{l+1} W_j^{l+1} of'(u^l) = \alpha_j^{l+1} up(\emptyset_j^{l+1}) of'(u^l) \tag{3}$$

where $l+1$ shows the pooling layer, W represents kernel and $\alpha_j^{l+1} up$ do up sample are shown in Equation (3).

$$\frac{\partial E}{\partial b_j} = \sum_{s,t} (\emptyset_j^l) u, v \tag{4}$$

$$\frac{\partial E}{\partial b_j} = \sum_{s,t} (\emptyset_j^{l+1}) u, v \tag{5}$$

where $p(\emptyset_j^{l+1})$ shows the patch for the deep network of y_j, k_{ij} denotes the patch center in samples of datasets, and the partial derivative of the error function is mentioned in the above Equations (4) and (5).

$$y_j^l = A(down(y_j^{l-1}) + b_j \tag{6}$$

In the above Equation (6), $down$ connotes a process involving collecting and combining resources, and b shows value bias.

$$\emptyset_j^l = \emptyset_j^{l+1} W_j^{l+1} of'(u^l) \tag{7}$$

Precision is obtained from the above Equation (7).

Feature Classification:

The trained CNN is then put to use in the validation step, where the probability of malware attack is determined. The CNN takes in a feature picture from the validation process and spits out a two-dimensional vector whose elements reflect varying degrees of harmful or benign behavior. The harmful value exceeds the benign value if the input picture is determined to be malicious. In order to determine the probability that a piece of malware will infect a

system, we apply the following sigmoid function in Equation (8) on the value of the harmful class:

$$P(X) = \frac{1}{1 + e^{(-X)}} \tag{8}$$

Where, X is the malicious class.

Digitizing and channelizing integrated behavioral feature data and the behavior frequency table retrieved from behavior feature metadata and memory produces these images. The proposed model can identify malicious code created in the training dataset with the ZFNet model by training on display pictures. A performance evaluation is carried out to determine if the model this work proposes a method that can accurately analyze and detect malware, from the most prevalent forms to the constantly evolving new and variant forms, and then generate an output that cleanly separates the malware from the input data.

The architecture of threat detection is shown in Figure 2. The storage infrastructure comprises three databases: traffic, behavior, and log. The raw Android files from devices using the control unit are collected and stored in the traffic database. The history of a network is stored in the database as a collection of datasets. The behavior database is kept up-to-date by constantly collecting new malware fingerprints from the log database.

Processing a large amount of data requires more time and human resources. As a result, incorporate a Deep Neural Network (DNN) and a color image transformation technique into the design of the analyzer unit.

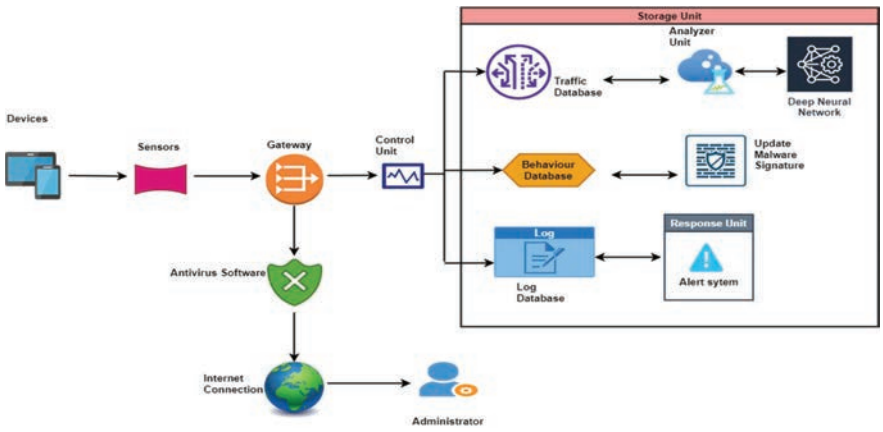

Figure 2 Architecture for threat detection.

The DNN model is fed a color image previously converted from the raw Android file. File fingerprints are compared to a behavior database, and the analyzer decides whether or not the file is malicious. If malicious behavior is exploreed on the network, the response unit delivers a final warning to the system administrator, who can then take appropriate action. Then Antivirus software connected to the gateway helps detect the threat recognized and will be displayed for the administrator through the anti-virus software. Malware with more temporal variability in time has been shown to be more consistently classified as harmful by antivirus engines. Figure 2 compares the distributions of malicious samples with low temporal variability (lower than the 25th percentile) to those with high variability (on the top of the 75th percentile). In November of 2019 (exactly one year after the first observation of the samples), we will check Virus Total for the total number of detections for each sample. The samples that were subsequently picked up by AV software exhibited more temporal variation, as was seen. This demonstrates that time tends to have a more profound influence on variability for samples that are simpler to identify, even if temporal variability of time is low across the board generally. It has been shown that the primary benefit of Deep Learning algorithms is their attempt to learn abstract characteristics from data in a gradual fashion. Thus, neither domain knowledge nor rigorous feature extraction are required. In deep learning, a computer model is trained to make inferences about the world around it without being explicitly programmed to do so. When it comes to accuracy, deep learning algorithms may sometimes even outperform humans.

Detection of Spyware:

Spyware is malicious software that sneaks into a user's device without their knowledge, monitors their online activities without their consent, and sends this data to third parties without their knowledge or consent. Spyware, by definition, is a form of malicious software that gains access to and disrupts a user's device without the latter's knowledge or permission. The raw file is crucial to the success of any android project while using android studio for development. Audio and video files (in formats like MP3 and MP4) and other file types can be stored in Android's raw folder. Specifically, the raw folder will be created in the res folder, at raw. To detect threats, one must conduct a thorough investigation of the complete security infrastructure in order to identify any malicious actions that might compromise the system. Once a threat has been identified, appropriate countermeasures must be taken to prevent it from exploiting any existing vulnerabilities in the system.

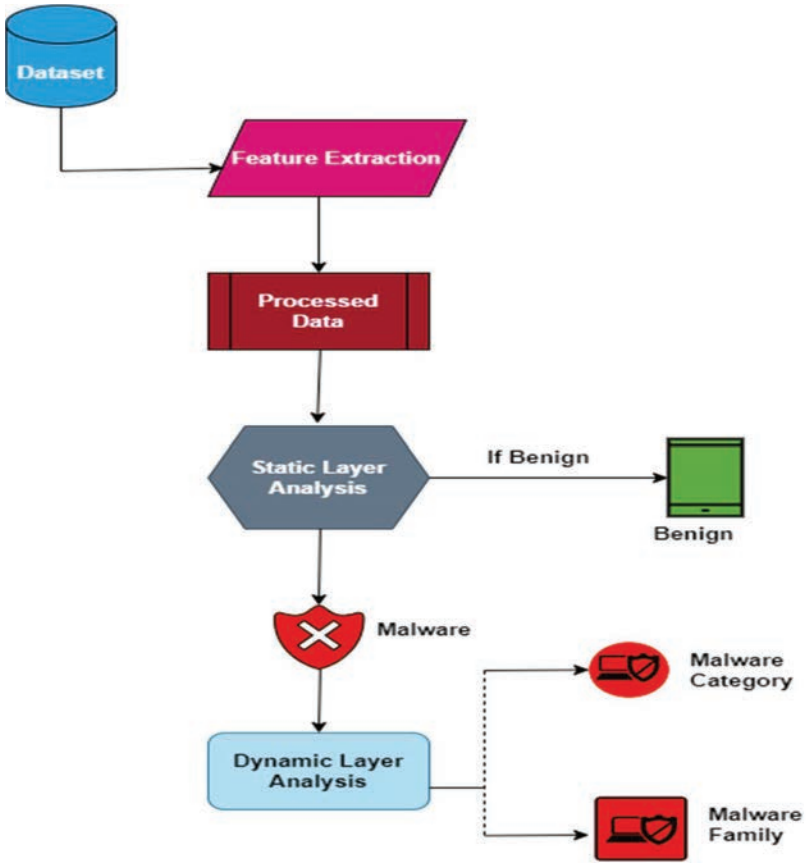

Figure 3 Detection of malicious android application in a visual format.

Figure 3 gives a visual format for the detection of malware. It summarizes the proposed method, which consists of Static binary classification and Malware detection classification. Therefore, samples from the dataset are sorted into malware and benign categories in the Static layer. After being identified as malicious at the Static layer, models are further categorized at the Dynamic layer into one of four types and 39 families. There are four types of malware which are as follows:

1. Ransomware
2. Scareware
3. SMS Malware
4. Adware

1. Ransomware:

Ransomware is malicious software that encrypts user data and locks the infected device before demanding payment from the victim. Once files have been encrypted, even if the ransomware is removed, the decryption key will not work to restore the files. The android system is vulnerable to ransomware because it handles permissions and intents.

In exchange for decrypting the victim's files, ransomware demands payment from the infected user. Both lock-screen and crypto-ransomware are common on Android. In the lock-screen context, a photograph that completely covers a smartphone's display is used as a security measure. Second, ransomware encrypts the client's vital data, as in the crypto class. The definition of Android ransomware often mirrors that of a Trojan horse. To further confuse victims, ransomware can assume the appearance of a legitimate law enforcement agency or label.

2. Scareware:

Scareware is malicious software designed to trick users into downloading and installing unnecessary programs. Scareware is software intended to intimidate or trick users into providing personal information on a phishing website. Scareware is software designed to intimidate its target audience by masquerading as a helpful program that usually protects the user's computer from malicious software. Fake filtering exchanges, fake progress bars, and fake alarms are all features designed for scareware.

3. SMS Malware:

Malicious text messages can be sent from an infected Android device if the malware is allowed to take over the device's messaging system. The owner uses the malware to have the infected phones send messages in his direction. Malware explicitly designed to target a compromised mobile device is often used in SMS attacks. Without the client's knowledge or permission, the Trojan can be programmed to initiate unauthorized communications such as phone calls or text messages. These communications are then routed to premium-rate or chargeable SMS content services, providing the cybercriminal with massive revenue streams.

4. Adware:

Adware is a malicious program that compromises computer systems and the personal information of its users. Adware can harm the client in several

ways, including stealing his data and sending it to a remote server, stealing his screen real estate, or displaying notifications for unrelated ads usually reserved for major structural events. An adware program can hijack a mobile device's volume in rare instances. Adware is software whose primary function is to force users to view or click on many advertisements, banners, and posts without their knowledge or consent. Any software that consistently displays advertisements based on a user's browsing or application activity history is considered adware. One aspect of this is data collection, which is often done for legitimate reasons can be used maliciously.

Category in an Ensemble:

An innovative method for malware identification is proposed in this study; it employs both fully connected and convolutional neural networks are used as the underlying learning mechanism and is founded on a category in an ensemble learning approach. Cybercriminals and hackers often target critical computer systems and networks to steal information, exact financial compensation, or demonstrate their skills. When identifying previously explored malware, the tried and true methods performed admirably. Log data is the most basic data type created by any and all computers, services, and network nodes, and it may be collected at the host level. Log data is a useful information source for detecting attacks in IT networks because it allows for monitoring of the current state of a computer system or network. Syslog, journald, and the Windows event log are all well-known services for logging data and making it accessible.

Managing the storage infrastructure includes a wide range of tasks, such as ensuring its availability, enough capacity, optimal performance, and flawless security. These components are interdependent and must operate in conjunction to get optimal ROI.

A SAN (storage area network) is a network of storage devices that can be accessed by multiple servers or computers, providing a shared pool of storage space. Each computer on the network can access storage on the SAN as though they were local discs connected directly to the computer.

Data selection is the first step in the system methodologies, as shown in Figure 4, and progresses to the model evaluation stage. Machine learning techniques can detect malware in two phases: feature extraction (to determine which data characteristics should be used to make predictions) and classification/clustering. The proposed technology centers on machine learning, which can be trained to identify malicious from safe files and make accurate predictions about files that have never been seen.

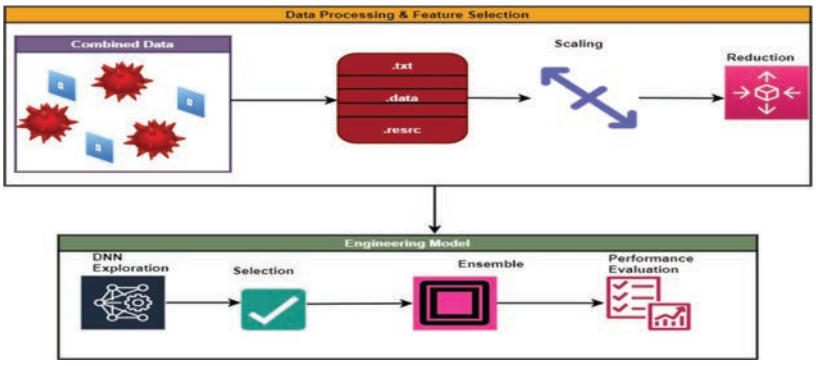

Figure 4 Malware detection procedures.

The following steps, which are part of the data processing, feature selection, and engineering model, ultimately lead to the desired result. They are processing, exploration, and choice of data-neural-network models; construction, testing, and evaluation of ensemble models; assessment of model performance.

Ensemble:

Ensemble techniques assume that a base classifier can be applied to many differently rearranged training datasets (through resampling or reweighting) to generate an ensemble of base classifiers. By merging the prediction impacts of these individual base classifiers, a new ensemble classifier is caused by the stacked ensemble approach, with the latter learning how to integrate the predictions of the former more effectively. Here employed a two-step stacking procedure.

Initially, several models are trained using the same dataset. The results from each model are then combined to form a fresh set of information. Each item in the current data set connects to the actual value it is trying to approximate. Secondly, the output is derived from the dataset used to train the meta-learning.

Figure 5 depicts a stacking model architecture and a conceptual overview of ensemble classification in which level-0 models serve as the foundation upon which a meta-learner (or generalizer) that combines the projections of multiple level-0 models is built. Multiple atmospheric dispersion models are used for each simulation in this multi-model ensemble. Typically, data from many meteorological models is combined with the various atmospheric dispersion models. The metrics, optimizer, and loss function are all defined

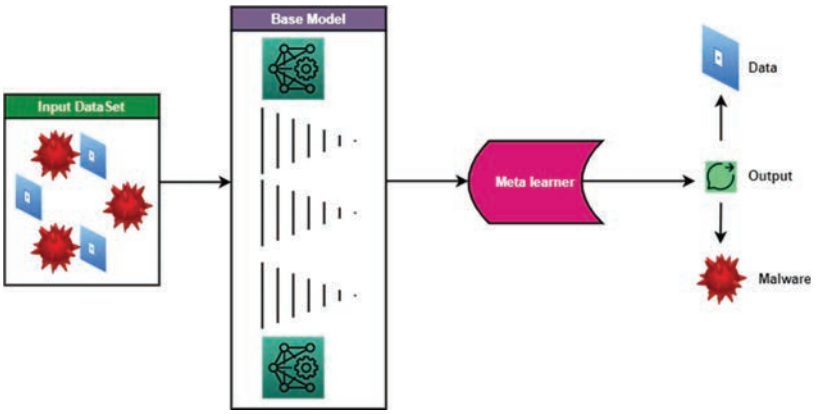

Figure 5 A conceptual overview of ensemble classification.

during compilation. Compiling a model several times will not affect the quality of the pretrained weights, and it has nothing to do with the weights either. It's impossible to train without a constructed model (because training uses the loss function and the optimizer). Learning to learn, or meta-learning, is the process of methodically examining the results of various machine learning methods across a broad variety of learning tasks and then using that information to accelerate the acquisition of new knowledge. Typically, metadata is embedded directly into an image file, along with the data that defines the image. Metadata text can be extracted from binary data with a program like Hex Fiend.

Base models are compiled with forecasts and can fit into the training data. The goal of training a meta-learner (level-1 model) is to merge the model's underlying predictions. The meta-learner makes use of straightforward models to inform their decision-making. To prepare the meta-learner, feed it a dataset consisting of input-output value pairs and the predictions of the underlying models. The data and malware are separated in the outcome section.

The suggested ensemble learning model is tested for its classification accuracy. True positive (TP), true negative (TN), false positive (FP), and false negative (FN) values were determined by comparing the actual labels to the predicted labels. Here presumed a two-category problem, where one class is labeled positive, denoted by $+1$ and the other negative, represented by -1.

$$\mathrm{MPR} = \frac{\sum_{i=1}^{N} [b(y_i) = +1][x_i = -1]}{\sum_{i=1}^{N} [x_i = -1]} \tag{9}$$

Where, N is the inputs count and The above Equation (9) shows the minus positive rate (MPR)

$$\mathrm{PPR} = \frac{\sum_{i=1}^{N} [b(y_i) = +1][x_i = +1]}{\sum_{i=1}^{N} [x_i = +1]} \tag{10}$$

The above Equation (10) shows the plus positive rate (PPR)

$$\mathrm{MNR} = \frac{\sum_{i=1}^{N} [b(y_i) = -1][x_i = +1]}{\sum_{i=1}^{N} [x_i = +1]} \tag{11}$$

The above Equation (11) shows the minus negative rate (MNR) where $b(y)$ is the input classifier with input y_i and output x_i. The number of times malicious files are mistakenly identified as benign is the minus positive (MP). Still, the frequency with which they are accurately classified as harmless is the plus negative (PN).

$$\mathrm{Accuracy} = \frac{\mathrm{PPR}}{\mathrm{PPR} + \mathrm{MPR}} \tag{12}$$

Equation (12) above helps in calculating the accuracy.

$$\mathrm{Performance} = 2 * \frac{\mathrm{Accuracy} * \mathrm{Recall}}{\mathrm{Accuracy} + \mathrm{Recall}} \tag{13}$$

The above Equation (13) shows the performance of the proposed system.

$$\mathrm{CC} = \frac{\mathrm{PP} * \mathrm{PN} - \mathrm{MP} * \mathrm{MN}}{(\mathrm{PP} + \mathrm{MP})(\mathrm{PP} + \mathrm{MN})(\mathrm{PN} + \mathrm{MP})(\mathrm{PN} + \mathrm{MN})} \tag{14}$$

where CC represents the correlation coefficient of the sample data taken and correlation is achieved from the above Equation (14).

$$\mathrm{Recall} = 1 - \frac{1 - \partial_0}{1 - \partial_e} \tag{15}$$

where ∂_0 denoted the proportion of valid responses in the evaluation data, and ∂_e is the ratio of the percentage breakdown of correlation that can be explained by chance alone in Equation (15). Recall Rate is high in performance for the proposed model, and it's achieved by the above Equation (15).

Table 1 above exhibits accuracy variation, showing how each of the techniques above can record data as per Equation (12). Since it optimizes gains while minimizing losses, AVSD-MDLN is more accurate than its rivals.

Table 1 Accuracy variation

Epoch	DAMD	EC	MD-IIOT	DL-AMDC	DA-IMD	AVSD-MDLN
0	85.12	88.12	86.2	87.12	89.12	90.3
100	85.78	89.78	87.34	88.45	90.1	91.3
200	86.3	90.1	88.1	90.3	90.9	92.3
300	88.4	90.8	88.9	92.5	91.5	93.9
400	89.12	91.5	89	93	92.1	95.6
500	90.1	92.1	89.3	93.4	92.4	97.5

Table 2 Recall rate analysis

Epoch	DAMD	EC	MD-IIOT	DL-AMDC	DA-IMD	AVSD-MDLN
0	86.12	89.12	87.4	88.45	90.1	91.4
100	87.23	89.78	87.9	89.45	91.3	92.3
200	88.56	90.1	88.34	91.2	93.4	93.5
300	89.23	90.8	88.9	92.5	94.1	94.6
400	89.98	91.5	90.3	93.6	94.5	96.7
500	90.2	91.2	93	94.5	95	98.5

Accuracy is defined as the degree to which a calculated value is within a certain tolerance of the actual value. The term "precision" refers to the consistency with which a certain instrument or procedure returns the same result. Thus, precision refers to easily an outcome may be replicated, whereas accuracy measures how closely an outcome matches the original. Precision, precision, and calibration are all phrases that seem similar but have quite different meanings. Being precise is essential to being accurate. Unfortunately, accuracy is not a guarantee of precision. Accuracy is achieved by calibration and precise measurement.

The above methods can all record information according to the Equation, as shown in Table 2, which displays recall rate variation per Equation (15). AVSD-MDLN excels compared to its competitors because it maximizes profits while achieving a high recall rate.

Here, introducing Anti-Virus Software Detection for Malware with Deep Learning Network is a strategy for enhancing the accuracy and precision with which a procedure is typically performed. Two more methods have been developed to evaluate the previously stated approach, with the overarching intention of detecting data from malware through an assessment of precision, recall and accuracy, and other factors.

4 Results and Discussion

The research recommends defining and running a Deep Learning Network in Anti-Virus Software for Malware Detection and simulations to achieve optimal precision, performance, and accuracy. Different models are compared with one another in terms of recall rate, technical performance, precision, and accuracy (such as DAMD, EC, MD-IIOT, DL-AMDC, and DA-IMD). To evaluate the performance of the proposed AVSD-MDLN, here engage in two primary functions: (1) detection and identification and (2) attribution determining if an app is malicious, the purpose of which is to identify the type of malware that has been found.

The dataset is sourced from the cited website.

https://www.qub.ac.uk/ecit/CSIT/Research/SecurityIntelligence/DeepAndroidMalwareDetection/ [30].

Table 3 of the abovementioned data set is used for this analysis, and the problems that the proposed model solves are outlined, and how all the systems are compared.

4.1 Accuracy Analysis

The testing procedure for the accuracy analysis ratio is depicted in Figure 6. Change over time is measured along the X-axis, while accuracy is examined along the Y-axis. Time variation factor performance versus accuracy is crucial regarding malware analysis and detection. To help meet these requirements, the accuracy factor, represented by Equation (12), is provided.

4.2 Precision Analysis

The analytical precision outcome is shown in Figure 7. The optimum precision rate is determined by plugging Equation (7) into a graph where the Y-axis represents the analysis ratio of precision, and the X-axis represents time.

Table 3

S.No	Classification System	Benign	Malware	Access
1.	DAMD	741	1260	0.90
2.	EC	834	1000	0.92
3.	MD-IIOT	912	2367	0.89
4.	DL-AMDC	1005	2890	0.93
5.	DA-IMD	1034	3023	0.92
6.	AVSD-MDLN	2345	5634	0.97

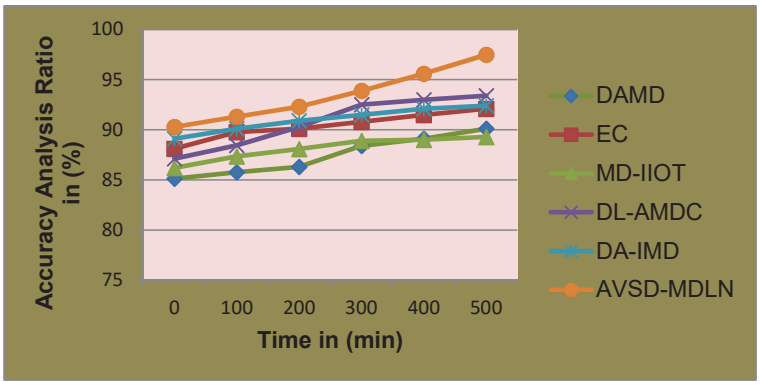

Figure 6 Accuracy analysis.

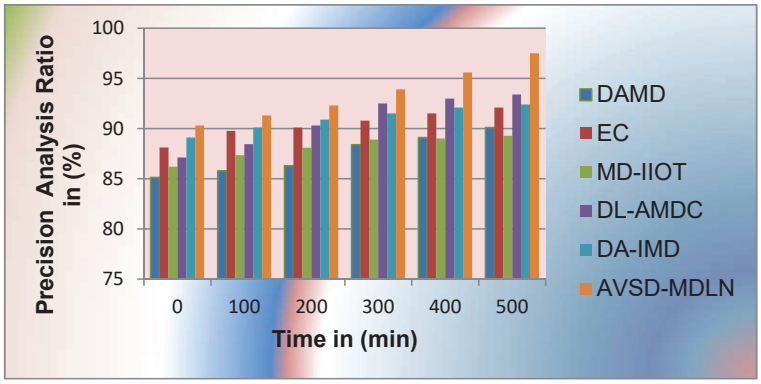

Figure 7 Precision analysis.

When it comes to analyzing and predicting malware detection, AVSD-MDLN is superior to all other models. Static features, such as a new feature based on control statement act of trying, are employed for sample classification in the process of static analysis. While it takes less time to train than s-value in Equation (16), its accuracy is only 97.3%, a difference of 1.6%. It's possible that the actual rate of mistakes is much greater in practice. In this way, s-value improved training speed marginally while also adding characteristics beyond those that are static. In order to get higher precision, it is preferable to sacrifice some time and effort. Static analysis cannot compete with s-value in either direction because of the time it takes to make projections.

$$s\text{-}value = (subsets\ of\ samples)/(malware\ count\ in\ train\ set) \quad (16)$$

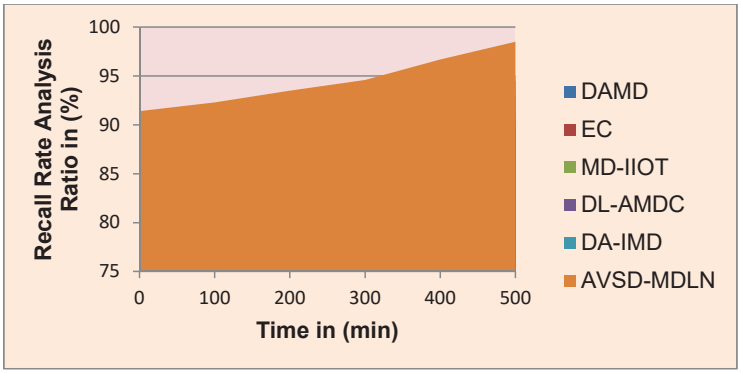

Figure 8 Recall rate analysis.

4.3 Recall Rate Analysis

Figure 8 displays the observed range in relay rates. The optimal recall rate is found by plotting the analysis ratio of recall rate versus time using Equation (15). There is no better model than AVSD-MDLN for analyzing and forecasting malware detection. When the dynamic nature of Android's malware landscape is taken into consideration, as demonstrated in figure, both detection techniques demonstrate strong performance in the identification of previously identified malware. Our system has an average F-measure of 0.94 when testing samples from the same year as the training, and F-measures of 0.84 and 0.72 when testing samples from one and two years following the training, respectively. Our average F-Measure is 0.92, whereas AVSD-MDLN's is 0.84 after one year of training and 0.70 after two years of training, therefore the two are similar. 14 out of 15 findings suggest that our system performs better when utilising newer samples for training and older for testing. In a worldwide comparison of the two methods, 25 of our findings are on par with or better than the AVSD-MDLN's, while 13 of the AVSD-MDLN's results are superior. Because of this, the approach can outperform AVSD-MDLN in identifying new forms of malware.

4.4 Performance Analysis

Figure 9 depicts the testing for the Performance evaluation. The Y axis represents the elapsed time, while the X axis displays the Performance analysis ratio. Equation (13) helps maximize the performance index compared to other models; the AVSD-performance MDLN is at its highest, making it ideal for analyzing and predicting malware.

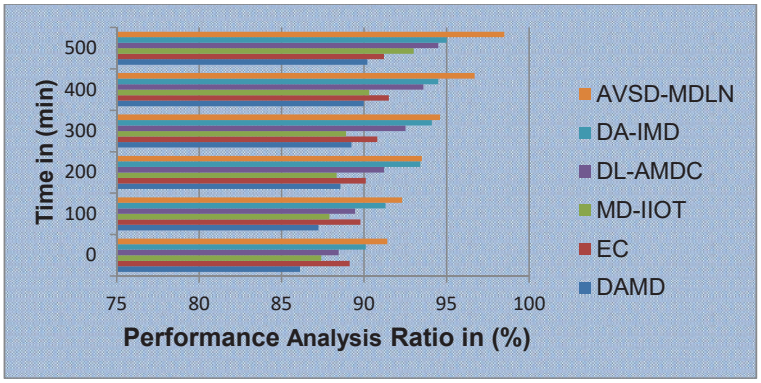

Figure 9 Performance analysis.

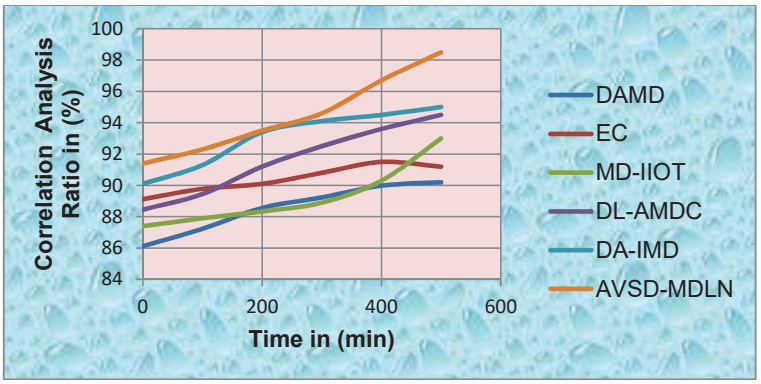

Figure 10 Correlation analysis.

4.5 Correlation Analysis

Examination for correlation analysis is shown in Figure 10. The time variation is plotted along the X axis, while the correlation is shown along the Y axis. Compared to other models, the AVSD- MDLN performance has the highest correlation index achieved from Equation (14), making it ideally suited for analyzing and predicting malware.

Existing models such as DAMD, EC, MD-IIOT, DL-AMDC, and DA-IMD have been compared to the proposed AVSD- MDLN -model, which has been shown to perform better than all of them. In this research, the authors conclude that these issues can be overcome with the help of virus detection in anti-virus software.

5 Endnotes

The need for innovative approaches to identify new forms of malware is growing since the current procedures are laborious and prone to numerous mistakes. Malware for Android devices is evolving, and new strains are being explored. Several investigations into the analysis and detection of malware have been conducted to respond directly to this problem. Protecting Android devices from malicious software, this paper proposed a new method based on AVSD- MDLN. This method may hamper our ability to identify evolving forms of malware as they emerge.

Moreover, the large amounts of behavior data required to detect Malware in E-devices are challenging to collect and store for extended periods due to the devices' limited hardware resources and purpose-built hardware. The remaining steps of model development – classification, feature selection, pre-processing, feature extraction, and bug fixing – were carried out on the cloud. The proposed model can stop malware from infecting or spreading to other devices over the Internet. The proposed model in this paper can analyze the behavioral features of malware in action through a dynamic analysis technique. Malware aware of its limited execution environment can potentially avoid dynamic analysis-based malware analysis and detection systems. Yet, the suggested malware detection model achieved a projected time (0.5 sec) and detection accuracy of 97.47% on the dataset. The Detection of Spyware and Category in an Ensemble are the two methods to accomplish the other models' precision, recall rate, and performance. Correlation coefficient is improved in the suggested framework. Likewise, the negative rate (MPR) and the positive rate (PPR) also improved. Traditional malware detection methods cannot be directly applied due to the computational complexity and limited resources (time, money, and memory) of devices. So, they hope to eventually develop a model for detecting malware that operates entirely in the block chain and relies on machine learning rather than on any persistent storage. There is no mention of the deep learning architectures' resilience in the paper offered. Since malware defection is a crucial application in mission-critical settings, this is a promising area for further research. Misclassification is a serious problem that may have far-reaching consequences for businesses. The future direction of the proposed research are these selected features help to build a model by considering different machine learning algorithms that is MLP and PCA.

References

[1] Sarker, I. H. (2021). Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective. *SN Computer Science*, *2*(3), 1–16.

[2] Huang, X., Ma, L., Yang, W., and Zhong, Y. (2021). A method for windows malware detection based on deep learning. *Journal of Signal Processing Systems*, *93*(2), 265–273.

[3] Catak, F. O., Ahmed, J., Sahinbas, K., and Khand, Z. H. (2021). Data augmentation based malware detection using convolutional neural networks. *PeerJ Computer Science*, *7*, e346.

[4] Nisa, M., Shah, J. H., Kanwal, S., Raza, M., Khan, M. A., Damaševičius, R., and Blažauskas, T. (2020). Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. *Applied Sciences*, *10*(14), 4966.

[5] Usman, N., Usman, S., Khan, F., Jan, M. A., Sajid, A., Alazab, M., and Watters, P. (2021). Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics. *Future Generation Computer Systems*, *118*, 124–141.

[6] Amin, M., Tanveer, T. A., Tehseen, M., Khan, M., Khan, F. A., and Anwar, S. (2020). Static malware detection and attribution in android byte-code through an end-to-end deep system. *Future generation computer systems*, *102*, 112–126.

[7] Mahindru, A., and Sangal, A. L. (2021). MLDroid – Framework for Android malware detection using machine learning techniques. *Neural Computing and Applications*, *33*(10), 5183–5240.

[8] Ren, Z., Wu, H., Ning, Q., Hussain, I., and Chen, B. (2020). End-to-end malware detection for android IoT devices using deep learning. *Ad Hoc Networks*, *101*, 102098.

[9] Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B., and Zheng, Q. (2020). IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*, *171*, 107138.

[10] Alzaylaee, M. K., Yerima, S. Y., and Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. *Computers & Security*, *89*, 101663.

[11] Gibert, D., Mateu, C., and Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments,

trends and challenges. *Journal of Network and Computer Applications*, *153*, 102526.

[12] Feng, R., Chen, S., Xie, X., Meng, G., Lin, S. W., and Liu, Y. (2020). A performance-sensitive malware detection system using deep learning on mobile devices. *IEEE Transactions on Information Forensics and Security*, *16*, 1563–1578.

[13] Pei, X., Yu, L., and Tian, S. (2020). AMalNet: A deep learning framework based on graph convolutional networks for malware detection. *Computers & Security*, *93*, 101792.

[14] Hwang, J., Kim, J., Lee, S., and Kim, K. (2020). Two-stage ransomware detection using dynamic analysis and machine learning techniques. *Wireless Personal Communications*, *112*(4), 2597–2609.

[15] Xiao, G., Li, J., Chen, Y., and Li, K. (2020). MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks. *Journal of Parallel and Distributed Computing*, *141*, 49–58.

[16] D'Angelo, G., Palmieri, F., Robustelli, A., and Castiglione, A. (2021). Effective classification of android malware families through dynamic features and neural networks. *Connection Science*, *33*(3), 786–801.

[17] Wu, Y., Wei, D., and Feng, J. (2020). Network attacks detection methods based on deep learning techniques: a survey. *Security and Communication Networks*, *2020*.

[18] Jha, S., Prashar, D., Long, H. V., and Taniar, D. (2020). Recurrent neural network for detecting malware. *computers & security*, *99*, 102037.

[19] Vasan, D., Alazab, M., Venkatraman, S., Akram, J., and Qin, Z. (2020). MTHAEL: Cross-architecture IoT malware detection based on neural network advanced ensemble learning. *IEEE Transactions on Computers*, *69*(11), 1654–1667.

[20] Jeon, S., and Moon, J. (2020). Malware-detection method with a convolutional recurrent neural network using opcode sequences. *Information Sciences*, *535*, 1–15.

[21] Park, S., and Choi, J. Y. (2020). Malware detection in self-driving vehicles using machine learning algorithms. *Journal of advanced transportation*, *2020*.

[22] D'Angelo, G., Ficco, M., and Palmieri, F. (2020). Malware detection in mobile environments based on Autoencoders and API-images. *Journal of Parallel and Distributed Computing*, *137*, 26–33.

[23] Pektaş, A., and Acarman, T. (2020). Deep learning for effective Android malware detection using API call graph embeddings. *Soft Computing*, *24*(2), 1027–1043.

[24] Darabian, H., Homayounoot, S., Dehghantanha, A., Hashemi, S., Karimipour, H., Parizi, R. M., and Choo, K. K. R. (2020). Detecting cryptomining malware: a deep learning approach for static and dynamic analysis. *Journal of Grid Computing*, *18*(2), 293–303.

[25] Imtiaz, S. I., ur Rehman, S., Javed, A. R., Jalil, Z., Liu, X., and Alnumay, W. S. (2021). DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network. *Future Generation computer systems*, *115*, 844–856.

[26] Damaševičius, R., Venčkauskas, A., Toldinas, J., and Grigaliūnas, Š. (2021). Ensemble-based classification using neural networks and machine learning models for windows pe malware detection. *Electronics*, *10*(4), 485.

[27] Naeem, H., Ullah, F., Naeem, M. R., Khalid, S., Vasan, D., Jabbar, S., and Saeed, S. (2020). Malware detection in industrial Internet of things based on hybrid image visualization and deep learning model. *Ad Hoc Networks*, *105*, 102154.

[28] Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., and Xiang, Y. (2020). A survey of android malware detection with deep neural models. *ACM Computing Surveys (CSUR)*, *53*(6), 1–36.

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

[30] https://www.qub.ac.uk/ecit/CSIT/Research/SecurityIntelligence/DeepAndroidMalwareDetection/

Biography

Peng Xiao was born in Kunming, Yunnan, P.R. China, in 1988. He received the bachelor's degree from Yunnan University Dianchi College, P.R. China in 2012. Now, he works in Information Center of Yunnan Power Grid Co., Ltd, Kunming, Yunnan, China. His research interests is mainly information security evaluation technology, include network attack and defense technology, network security management, enterprise security system construction, etc.