Applied Information Technology And Computer Science Vol. 3 No. 2 (2022) 032-049
© UniversitiTun Hussein Onn Malaysia Publisher's Office

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Comparison of Convolutional Neural Network and Artificial Neural Network for Android Botnet Attack Detection

Selvatarasi Balasunthar[1], Zubaile Abdullah[1]*

[1]Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2022.03.02.003
Received 24 July 2022; Accepted 06 October 2022; Available online 30 November 2022

Abstract: Mobile devices, such as Androids, are now widely used. Androids are used for making phone calls, sending text messages, web browsing, social networking, and online banking transactions. The Android operating system's global popularity makes it a more appealing target for cyber criminals to gains access on Android device, to steal valuable data by installing an Android botnet attack. Thus, this research presents the Android botnet attack detection using deep learning algorithms, Convolutional Neural Network (CNN) and Artificial Neural Network (ANN). The experiment was carried out and tested on 1929 botnet dataset and 4873 benign applications using different categories of permission features. The research covers several performance metrics like accuracy, precision, recall, f1-score, true-positive and false-positive in identifying the best performance classifiers. At the end of the study, the ANN classifier was identified to be best classifiers for Android Botnet attack detection with the highest detection accuracy 96.35% whereas the detection accuracy obtained by CNN is 95.44%. In addition, the performance metrics derived from Android botnet attack detection using CNN and ANN were better than those obtained from prior studies that employed machine learning algorithms for Android botnet attack detection.

Keywords: Android Botnet Detection, Deep Learning, Permission Feature, CNN, ANN

1. Introduction

A botnet is a network of devices that are controlled by a botmaster, a malevolent user, or a group of malicious attackers. The botnet comes with a Command and Control (C&C) infrastructure, which allows malicious actors to send commands, updates, and status information to the bots [1]. Android botnet attacks are more dangerous than other mobile malware such as mobile phishing attacks, ransomware, spyware and Trojans. This is because they pose dangerous threats to Android devices and networks [2]. The botmaster can then use the infected Android devices to commit cybercrimes or cyber-attacks, such as sending spam messages, disrupting networks, launching distributed denial-of-service (DDoS) attacks, and collecting sensitive data for illegal purposes.

**Corresponding author: zubaile@uthm.edu.my*

Due to Android botnet attacks malicious action towards the android users, many researchers have extensively done research to detect the Android Botnet attack [3]. To detect Android botnet attacks, the researchers used well-known machine learning methods, but these traditional methods are incapable of detecting new sophisticated Android Botnet attack. According to research findings, the majority of the machine learning techniques used in the classification models achieved an overall classification accuracy of over 85% [4]. As a consequence, attacks and threats of android botnets have increased in recent days. According to a survey conducted by Azure DDoS protection, the number of Android botnet threat families has increased significantly in both volume and complexity. In the first half of 2021, the average daily number of attacks increased by 25% when compared to the fourth quarter of 2020 [2].

Many studies have been conducted to detect Android botnet attacks, but their classification accuracy can still be improved [5]. Lower accuracy is caused by use of insufficient data or smaller data in the experiments. Since, machine learning typically requires structured data and implements traditional algorithms, it is incapable of handling large amounts of unstructured data [6]. The small size of a dataset is also responsible for poor performances of Android botnet detection [4]. This has an effect on detection accuracy because as the size of the sample data collection is limited, the confidence in the estimate reduces and the uncertainty rises, resulting in lower precision. When it comes to achieving high efficacy of Android botnet detection, developing more data is always a good idea. Furthermore, the use of untrained data has an impact on the Android botnet detection result [7]. Trained data is the primary and most important data that machines use to learn and predict. Increasing the amount training data provides more information and assist in better user fit.

These findings indicate that the proposed method for detecting Android botnet attacks could be improved further. Many studies have shown that using deep learning algorithms can improve the detection of android botnet attacks. As a result, this study was proposed to detect android botnet attacks in order to improve detection accuracy, precision, recall, f1-score, true-positive and false-positive results. Due to the organized hierarchy of increasing abstraction and complexity, deep learning algorithm has used in this study. The total 130 permission features extracted from 1,929 botnet dataset, and 4873 benign applications. These features were then used to perform feature extraction and feature selection. The Convolutional Neural Networks (CNNs) and Artificial Neural Networks (ANNs) algorithms used to test and train the datasets to distinguish between botnet and benign applications. The following are the objectives of this project:

- To study the permission features for Android botnet attack detection.
- To propose Android botnet detection using Convolutional Neural Network and Artificial Neural Network algorithm.
- To test and validate the proposed model using Accuracy, Recall, Precision, F1- score, True-Positive and False-Positive.

2. Related Work

2.1 Android Botnet Architecture

The Android botnet is examined using the botmaster's architectural design. Botmaster builds robust and complex botnets using a variety of topologies and tools. As a result, it is more difficult to detect the botnet. A botnet can be divided into two parts based on its architectural design which are centralized and decentralized [4].

Centralized generally employs Hypertext Transfer Protocol (HTTP) and Internet Relay Chat (IRC) protocols. IRC operates in real time on internet text messages. Due to the flexibility and simple architecture, IRC bots are popular among botnet owners. The limitation of the IRC botnet is that it can be easily detected by identifying IRC traffic for regular traffic. HTTP botnet traffic can easily be disguised as normal traffic. When the HTTP protocol is used for communication, detection becomes more difficult. Some well-known botnets that use an HTTP-based model are Click Bot, Bobox, and Rustock [2].

The vulnerability of the Command-and-Control channel is overcome by decentralized models. In the decentralized model, a large number of bots can be created in a single botnet, making the detection extremely difficult. The decentralized model employs P2P protocols that connect all bots. These protocols are primarily concerned with concealing a Command-and-Control channel. When new commands are issued, botmaster employs a variety of bots. Bots are reliant on previous bots or any other bots that are linked to it. P2P architecture is extremely complex and difficult to detect. Slapper, Phatbot, Sinit, and Nugache are some of the examples for P2P architecture. The hybrid model is a combination of decentralized and centralized models. Botnet traffic can be hidden using an encryption key in the hybridized model [2].

2.2 Android Botnet Analysis Techniques

Two common techniques used for Android botnet detection can be categorized into static and dynamic analysis.

Static analysis is the examination of an application without running it. Static analysis can be performed directly on the source code of the application or the corresponding binary file, using reverse engineering techniques to extract specific features and methods invoked from the source code. Manifest files can be used to investigate features and methods in Android apps. In addition to detecting malicious payload, extracted features or methods can be used to profile and weigh malware threats [8]. In contrast, some of the features and methods that are typically extracted from application source code are Requested Permission, Imported Package, API Calls, Instructions or Operation Code (Opcode), Data Flow, and Control Flow [9]. Static analysis is simple and effective at detecting and classifying known Android botnet attacks; however, due to obfuscation and encryption techniques used by Android botnet authors, it is incapable of detecting unknown or modified Android botnet attacks. Dynamic analysis is used to detect Android botnet attacks in order to circumvent this limitation.

Rather than examining the source code, dynamic analysis examines the application sample while it is running in a controlled environment. According to recent research, the application's behaviour can be monitored using Logged Behavior Sequence, System Calls and Dynamic Tainting, Data Flow and Control Flow [9], since this statement is prepared for detection analysis by monitoring and logging every relevant execution operation.

2.3 Deep Learning Algorithm

A deep learning algorithm is a subset of a machine learning algorithm that performs data processing and calculations on massive amounts of data using multiple layers of neural networks. The deep learning system can learn from both structured and unstructured data that does not require human involvement. The way the human brain operates, and functions inspires deep learning algorithms as the deep learning algorithms rely on neural networks in the same way as the human brain uses millions of neurons to compute information [1]. There are three different types of layers in deep learning algorithms which are known as input layer, hidden layer, and output layer. The input features and a well-known dataset are included in the input layer. Hidden neurons which are known as hidden layers, must be educated in order for the brain to function properly. The output layer generates the value to be categorized. Deep learning algorithms can be classified into two categories which are the supervised and unsupervised [3].

Deep learning algorithms that require external assistance are known as supervised deep learning algorithms. Regression and classification problems are the two most common types of supervised learning. The goal of classification is to predict a label or class. Classification algorithms are used to predict a categorical variable, whereas regression algorithms are used to predict a continuous quantity. When training materials are not classified or labeled, unsupervised deep learning methods are used.

Unsupervised learning aims to pre-train a model called discriminator or encoder network to be used for other tasks. Auto-encoder, clustering, learning and generative models are some of the categories from unsupervised deep learning algorithms.

2.3.1 Convolutional Neural Network (CNN)

The most promising deep learning model development method is the Convolutional Neural Network (CNN), also known as ConvNets. CNN is skilled at detecting simple patterns in data that are then used to create more complex patterns in higher layers. Convolutional layer, pooling layer, and fully linked layer are the three layers that make up a CNN algorithm. The CNN's performance can be enhanced depending on the dataset's features, the number of layers, the number of filters (kernels), and the size of the filters [3]. The required number of layers is determined by the sophistication and non-linearity of the data, as deeper layers of the CNN recover more abstract features [3]. As the number of layers and filters increases, so does the computational complexity. Furthermore, with more complex architectures, the risk of overfitting for the classification algorithm model increased and resulting in poor prediction accuracy on the testing sets. During model training, techniques like 'dropout' and 'batch regularization' are used to reduce overfitting.

The CNN classification model as shown in Figure 1 is intended to classify the Android botnet as well as benign samples. Convolutional layer generates the feature vectors from the pre-processed data. The pooling layer used to reduce the data dimension by deleting irrelevant data as it has been converted into feature vectors. Following the pooling layer, the feature vector is flattened by a multidimensional array. The flattened array is then loaded into CNN algorithm in order to train the model [1]. Accordingly, the final layer of CNN is the data link layer, used to classify the android botnet samples and benign samples after the CNN model has been trained. A one-dimensional Convolutional Neural Network can be used to process datasets with a one-dimensional structure (1D CNN). The dimensionality of the input data and how the filter (feature detector) slides across the data are the main differences between a 1D and a 2D or 3D CNN. The filters in a 1D CNN only move in one direction across the input data.

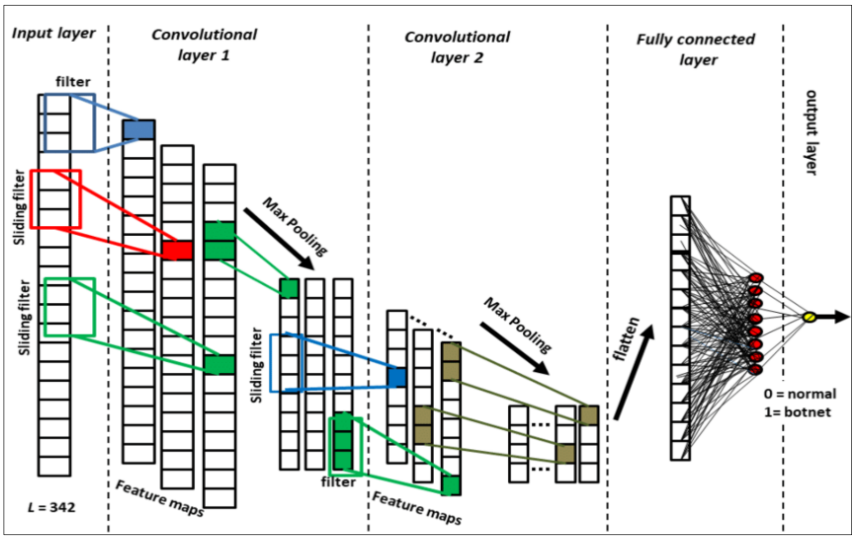


Figure 1: The 1D CNN model to classify Android Botnet attacks [3]

The sigmoid activation function used in 1D CNN model and will be calculated as follows:

$$S = \frac{1}{1+e^{-x}} \qquad Eq. 1$$

The final classification layer generates results that correspond to the 'botnet' or 'benign' classes. The activation function for ReLU (Rectified Linear Units) is f(x) = max(0, x), which is implemented in the convolutional layers. ReLU can help prevent vanishing and exploding gradients. ReLU has been found to be more efficient in terms of time and cost for training large data sets than traditional non-linear activation functions such as Sigmoid or Tangent functions.

2.3.2 Artificial Neural Network (ANN)

Artificial Neural Networks (ANN) is a deep learning algorithm that are inspired by biological neurons in the brain and central nervous system. The ANN's inputs are fed to the artificial neurons in one or more hidden layers, where they are weighted and processed to determine the next layer's output. Back-propagation of errors in ANN is based on gradient descent, which allows the weights and biases of neurons in the hidden layer and output layer to be adaptively modified [3]. Along with its self-adaptive nature, ANN can capture very complicated and non-linear interactions between dependent and independent variables without the need for prior information. In a variety of applications, ANN have been used to solve a wide range of classification problems.

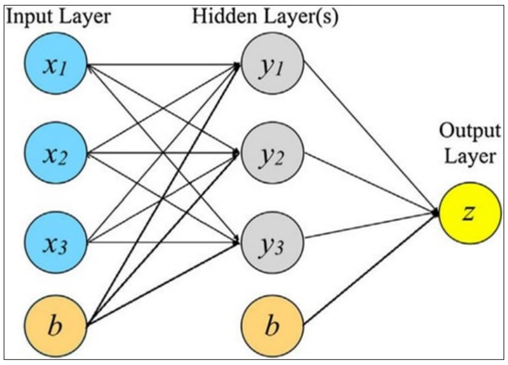


Figure 2: The classification model of ANN [11]

ANN algorithm consists of three layers which are input layer, hidden layer and output layer in Figure 2. In order to consider compelling to (0-1) range, the input layers and output layers must have numeric values. As a result, the data is normalized within the (0-1) range before being passed to the input layer [3]. In the hidden layer, the weight is the set of performance parameters for the feed-forward neural network. Starting with random weights, bestowing the data, instance by instance, modifying the weights, imparting the error for each instance, and continuing until the error is very small, the training method of the ANN is exaggerated. The backpropagation algorithm adjusts the weights for each instance based on the variance of the actual output and function output.

2.4 Existing Android Botnet Detection Techniques

Table 1 shows the comparison of existing Android botnet detection techniques and proposed research

Table 1: The comparison of existing Android botnet detection techniques

Research Paper	Approach	Dataset	Features	Classifier	Strength
[4]	Static analysis	1365 botnet dataset from ISCX Android Botnet and 1960 Benign application from Google Play Store	Requested permission and protection levels	Random Forest, Multilayer Perceptron, J-48 Decision Tree and Naive Bayes	Promising Detection Power
[10]	Static analysis	Benign apps from open-source site and Botnet apps from botnet repositories	Requested permission and used features	Naive Bayes, A Statistical Classifier, Support Vector Machine, Reduced Error Pruning	High accuracy and low positive rate

Table 1: (cont.)

Research Paper	Approach	Dataset	Features	Classifier	Strength
[8]	Static analysis	1400 botnet applications from ISCX Android Botnet and 1400 Benign application from Google Play Store	Permissions, Broadcast Receivers and Background Receivers	Support Vector Machine, Random Forest, Naive Bayes, J-48 Decision Tree, Bagging	Four separate layers and using MD5 signatures
Proposed	Static analysis	1929 botnet applications from ISCX Android Botnet and 4873 Benign application from Google Play Store	Permissions	Convolutional Neural Network, Artificial Neural Network	Expecting high detection accuracy, recall, precision

3. Methodology

The research methodology is perhaps the sequence of measures to be followed during the evaluation of research studies. The detection framework is made up of basic system components and general processing steps for detecting Android botnet attacks using CNN and ANN. There are six phases in total which include raw data, pre-processing, feature extraction and feature selection, classification algorithm, and performance metrics as shown in Figure 3.

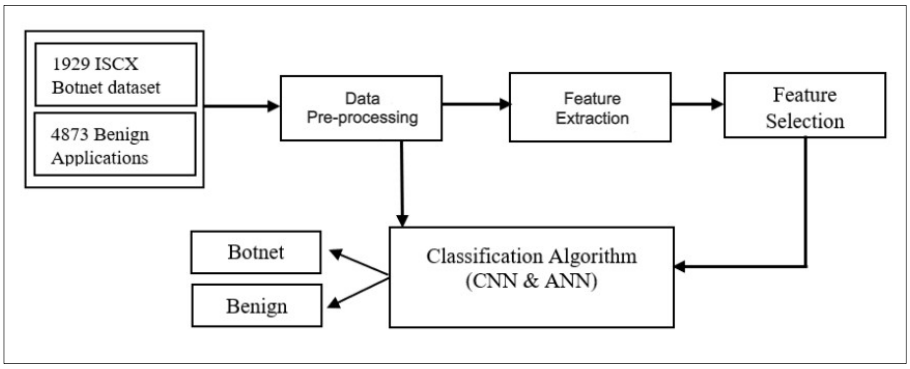


Figure 3: Proposed Android Botnet Attack Detection Methodology

3.1 Raw Data

The term "raw data" refers to the information that has not been filtered or normalized. As a consequence, to perform Android botnet detection using Convolutional Neural Networks and Artificial Neural Networks, the Android dataset, also known as the ISCX dataset, which contains 1,929 botnet datasets from 14 different families, was obtained from an online source (ISCX-AndroidBot). Moreover, this study also used a total of 4873 benign application downloaded from Google Play Store to facilitate supervised learning while training the Convolutional Neural Network and Artificial Neural Network algorithm. The benign applications were found in a variety of categories on the Google Play store. While downloading the dataset from an online source, the botnet datasets were found in form of .apk file whereas for benign applications, the name list of applications only provided. The .apk file for each benign applications were obtained by downloading the specific Google Play applications using the APKPure. In order to decompress the .apk files of the ISCX botnet dataset and benign applications, VirusTotal has been used. VirusTotal decompiles the .apk files to source code folders and declares the benign applications files to be virus-free and identifies the malware percentage of the botnet for botnet dataset files. It also provides the detailed information of each dataset file which will be useful to extract the features.

3.2 Data Pre-Processing

Data pre-processing is the data mining technique which changes the data to an efficient and useful format. The data is preprocessed to identify those features from the data that play a significant role in the classification of dependent features, which enables this study to proceed. The data and 130 extracted permission features from the decompiled .apk file, of 1929 botnet datasets which include 14 types of botnet families and 4873 benign applications, is loaded and saved as a.CSV (comma delimited) file with each feature denoted by a '1' or a '0,'. These properties are represented as a binary number feature vector, with '1' indicating the botnet and '0' indicating the benign [3]. The data file is then preprocessed for training and testing by removing redundant data. Preprocessing of the botnet dataset and benign application is essential in identifying relevant data for the study before it's being used to implement it in the deep learning machine. The RapidMiner and WEKA tools were used to assist with the preprocessing process by removing the redundant data and applying the nominal filters to the datasets.

3.3 Feature Extraction

The process of extracting features from a dataset is known as feature extraction. Since large datasets contain many variables, processing them requires a large amount of computing resources. Therefore, feature extraction involves reducing the number of the variables of the large dataset to get the available variable for the research. Initially, 130 permission features were extracted for this study from the raw data, 1929 ISCX(Android-Bot) and 4873 benign applications. To perform the feature extraction on these 130 permission features, three new features were extracted based on the permission feature's protection level. The protection level of permission feature consists of Dangerous permission feature, Normal permission feature and Signature permission feature as shown in Table 2.

Table 2: The protection level of permission features

Protection Level Permission	Description
Dangerous	A higher-risk permission that grants requesting applications access to isolated application-level features while posing little risk to other apps, the system, or the user.
Normal	A lower-risk permission that would grant a requesting application access to private user data or device control, which could be harmful to the user.
Signature	The system will only grant this permission if the requesting application is registered with the same certificate as the one that declared on the permission.

These features indicate how the application is concerned with gaining privilege on the target mobile's resources and accessing more resources. So, the collected dataset and features are extracted according to the protection level of permission features.

3.4 Feature Selection

Feature selection is a technique for removing redundant and irrelevant features from a dataset. The Information Gain (IG) algorithm is used to implement the feature selection approach. To ensure the selection of the most discriminant features, the obtained feature sets are passed through the information gain (IG) feature selection algorithm. In terms of feature selection and ranking methods, the IG is very accessible. It is defined as the amount of data provided by the feature items for the document category

[11]. In order to measure the importance of features for classification, IG is calculated by how much of a term can be used for classification of information as shown Equation 2.

$$E = -\sum_i^C P_i \log_2 P_i \qquad Eq.2$$

Approaching feature selection, the list of capabilities is chosen by algorithmic processing on the WEKA tool by implementing the Information Gain. The top 10 ranked features and top 20 ranked features were selected for Convolutional Neural Network and Artificial Neural Network model classification by processing the algorithm.

3.5 Classification Algorithm

A classification algorithm is used to categorize each byte of data in a dataset into one of several predefined groups. The performance of Android botnet detection was evaluated on the Convolutional Neural Network and Artificial Neural Network classifier using the extracted features and selected features of the dataset. The detection model will be classified with the assistance of the WEKA tool. 10-Fold Cross Validation was used to test and train the datasets, since this technique divides the dataset into ten parts, referred to as "folds," and holds each part in turn before averaging the results. As a result, each data point in the dataset is tested once and then trained nine times. The goal of cross-validation is to avoid overfitting and make predictions that are more general.

3.6 Performance Metrics

In this phase, the performance of both CNN and ANN compared in terms of accuracy, precision, recall, f1-score, true-positive and false-positive as the following equations:

1. Accuracy: The rate of correctly classified instances of both classes is measured by accuracy in Equation 3. The botnet class is the positive class for Android botnet detection. (TP = True-Positive, TN = True-Negative, FP = False-Positive, FN = False-Negative)

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad Eq.\ 3$$

2. Precision: Measures the percentage of test data that is correctly identified as malicious android bot application from the malicious android botnet application classes in Equation 4. (TP = True-Positive, FP = False-Positive)

$$Precision = \frac{TP}{TP+FP} \qquad Eq.\ 4$$

3. Recall: The percentage of malicious android botnet application classes that were correctly detected is measured by recall in Equation 5. (TP = True-Positive, TN = True-Negative)

$$Recall = \frac{TP}{TP+TN} \qquad Eq.\ 5$$

4. F1-Score: The F1-score is a test accuracy metric that assesses the balance between precision and recall in Equation 6.

$$F1 - Score = \frac{2*RECALL*PRECISION}{RECALL+PRECISION} \qquad Eq.\ 6$$

5. True-Positive: The number of botnet applications that are correctly classified is known as true positive (TP) in Equation 7. (TP = True-Positive, FN = False-Negative)

$$True - Positive = \frac{TP}{TP+FN} \qquad Eq.\ 7$$

6. False-Positive : The number of botnet applications classified incorrectly is known as false positives (FP) in Equation 8. (TN = True-Negative, FP = False-Positive)

$$False - Positive = \frac{FP}{FP+TN} \qquad Eq.\ 8$$

3.7 Software and Hardware

The Waikato Environment for Knowledge Analysis (WEKA) is used since it is a data mining software that uses a collection of deep learning algorithms. WEKA is a collection of tools for regression, association, clustering, data preparation, visualization, and classification that can be used together or separately. The explorer, experimenter, knowledge flow, workbench, and simple CLI are the five applications available on WEKA. The explorer implements the data mining tasks on row data with a graphical interface. Aside from that, Rapid Miner was used to pre-process the datasets in order to remove redundant data. This is due to the fact that Rapid Miner makes ease datasets preparation for predictive modeling. Table 3 shows the hardware requirement to conduct this research.

Table 3: List of Hardware Requirements

Hardware	Description
Swift SF514-54T Processor	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
Windows Edition	Windows 10 Pro
System Type	64-bit operating system, x64-based processor
Installed RAM	8.00 GB (7.78 GB usable)

4. Results and Discussion

4.1 Experiment Setup

While downloading the 1929 botnet dataset from (ISCX-AndroidBot), the botnet dataset was found in form of .apk file whereas the 4873 benign applications list was generated from the Google Play Store. Then, the .apk file for each benign applications were obtained by downloading the specific Google Play Store applications using the APKPure.

In order to decompress the .apk files of the ISCX botnet dataset and benign applications, VirusTotal has been used. By uploading the .apk file into VirusTotal, it decompiles the files to source code folders that provides the detailed information of each dataset file which will be useful to extract the features. The useful information includes basic properties, permissions, activities, receivers, intent filters by action, intent filter by category, interesting strings, warning, contents metadata, contained files by type and contained files by extension. Other than that, VirusTotal also declares the benign applications files to be virus-free and identifies the malware percentage of the botnet dataset files as shown in Figure 4.

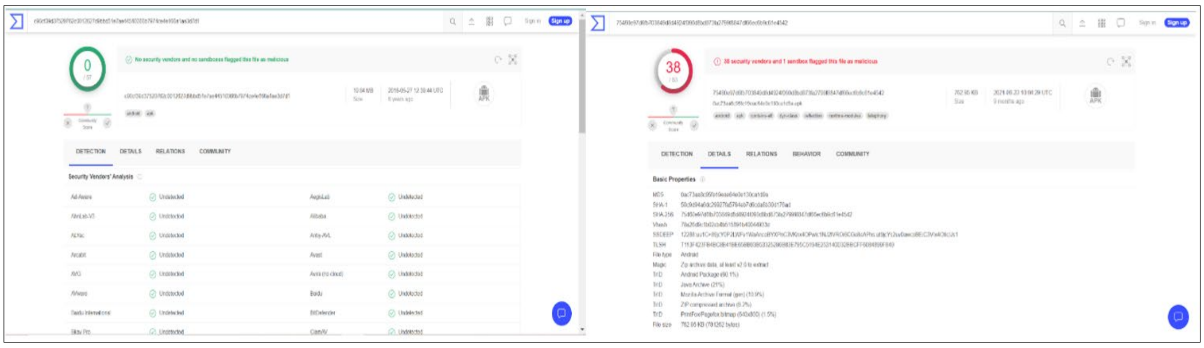
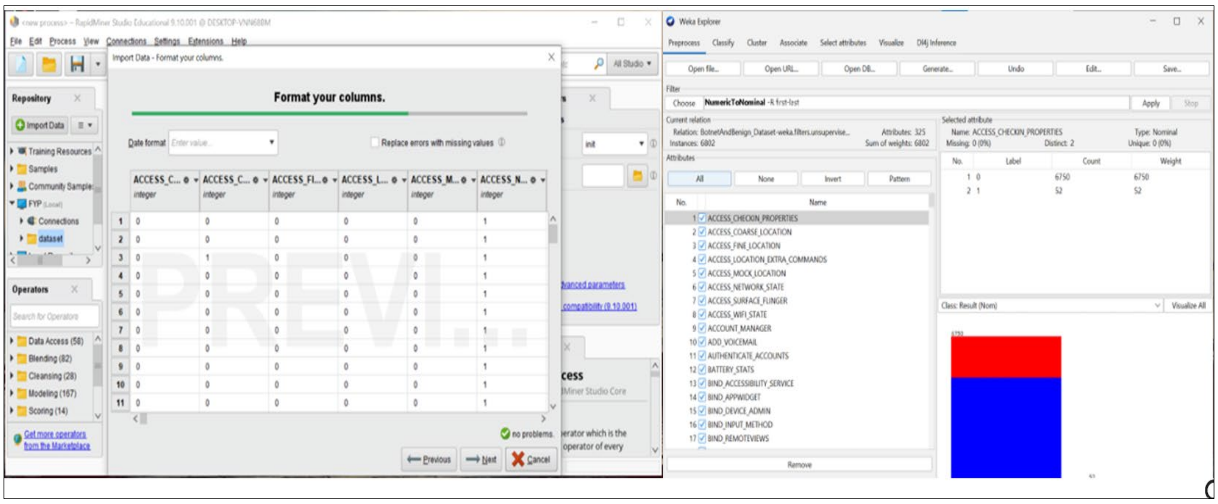

0
No security vendors and no sandboxes flagged this file as malicious
DETECTION
DETAILS
RELATIONS
COMMUNITY
Security Vendors' Analysis
38
38 security vendors and 1 sandbox flagged this file as malicious
BEHAVIOR
Basic Properties
Sign up

Figure 4: VirusTotal also declares the benign applications files to be virus-free and identifies the malware percentage of the botnet dataset files

4.2 Data Pre-Processing

The total of 6802 datasets and 130 permission features that obtained from decompiling the .apk file is loaded and saved as a.CSV (comma delimited) file with each feature denoted by a '1' or a '0,'. These properties are represented as a binary number feature vector, with '1' indicating the botnet and '0' indicating the benign [3]. The data file is then preprocessed for training and testing by removing redundant data. The 1929 botnet dataset and 4873 benign applications were pre-processed in the Rapid Miner to identify those features from the data that play a significant role in the classification of dependent features, making this study possible. As a result, the pre-processing has been completed by declaring that the datasets are free to use.



	ACCESS_C... integer	ACCESS_C... integer	ACCESS_FI... integer	ACCESS_L... integer	ACCESS_M... integer	ACCESS_N... integer
1	0	0	0	0	0	1
2	0	0	0	0	0	1
3	0	1	0	0	0	1
4	0	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	1
7	0	0	0	0	0	1
8	0	0	0	0	0	1
9	0	0	0	0	0	1
10	0	0	0	0	0	1
11	0	0	0	0	0	1

Filter: NumericToNominal -R first-last

Current relation
Relation: BotnetAndBenign_Dataset-weka.filters.unsupervise... Attributes: 325
Instances: 6802 Sum of weights: 6802

Selected attribute
Name: ACCESS_CHECKIN_PROPERTIES Type: Nominal
Missing: 0 (0%) Distinct: 2 Unique: 0 (0%)

No.	Label	Count	Weight
1	0	6750	6750
2	1	52	52

Class: Result (Nom)

No.	Name
1	ACCESS_CHECKIN_PROPERTIES
2	ACCESS_COARSE_LOCATION
3	ACCESS_FINE_LOCATION
4	ACCESS_LOCATION_EXTRA_COMMANDS
5	ACCESS_MOCK_LOCATION
6	ACCESS_NETWORK_STATE
7	ACCESS_SURFACE_FLINGER
8	ACCESS_WIFI_STATE
9	ACCOUNT_MANAGER
10	ADD_VOICEMAIL
11	AUTHENTICATE_ACCOUNTS
12	BATTERY_STATS
13	BIND_ACCESSIBILITY_SERVICE
14	BIND_APPWIDGET
15	BIND_DEVICE_ADMIN
16	BIND_INPUT_METHOD
17	BIND_REMOTEVIEWS

Figure 5: Data pre-processing in RapidMiner Studio and data filtering process in the Weka tool.

The preprocessed dataset file from RapidMiner Studio was then loaded and executed in the Weka tool to apply filters that make data preparation easier as shown in Figure 5. Following that, the nominal filter was applied to the numeric datasets. This is because a nominal rating indicates the filter's ability to prevent a specific percentage of solid particles larger than the nominal rating's stated micron size from passing through. The weight of these particles of each contaminant is then calculated. In addition, the nominal rating also indicates the degree of filtration or efficiency.

4.3 Feature Extraction

Three new features were extracted from the 130-permission which was obtained from the raw data, 1929 botnet dataset and 4873 benign applications by decompiling the .apk files using Virus Total. These three new features were extracted by classifying the permission features protection level which are consist of Dangerous permission feature, Normal permission feature and Signature permission feature. These features indicate how the application is concerned with gaining privilege on the target mobile's resources and accessing more resources. The features were extracted as shown in Table 4.

Table 4: Extracted Features based on the Permission Protection Level

Permission Protection Level	List of Features
Dangerous	ACCESS_COARSE_LOCATION
	ACCESS_FINE_LOCATION
	ACCESS_MOCK_LOCATION
	ADD_VOICEMAIL
	AUTHENTICATE_ACCOUNTS
	CALL_PHONE
	CAMERA
	GET_ACCOUNTS
	PROCESS_OUTGOING_CALLS
	READ_CALENDAR
	READ_CALL_LOG

Table 4: (cont.)

Permission Protection Level	List of Features
Dangerous	READ_CONTACTS
	READ_EXTERNAL_STORAGE
	READ_PHONE_STATE
	READ_SMS
	RECEIVE_MMS
	RECEIVE_SMS
	RECEIVE_WAP_PUSH
	RECORD_AUDIO
	SEND_SMS
	USE_SIP
	WRITE_CALENDAR
	WRITE_CALL_LOG
	WRITE_CONTACTS
	WRITE_EXTERNAL_STORAGE
Normal	ACCESS_LOCATION_EXTRA_COMMANDS
	ACCESS_NETWORK_STATE
	ACCESS_WIFI_STATE
	BLUETOOTH
	BLUETOOTH_ADMIN
	BROADCAST_STICKY
	CHANGE_NETWORK_STATE
	CHANGE_WIFI_MULTICAST_STATE
	CHANGE_WIFI_STATE
	DISABLE_KEYGUARD
	EXPAND_STATUS_BAR
	FLASHLIGHT
	GET_PACKAGE_SIZE
	GET_TASKS
	INTERNET
	KILL_BACKGROUND_PROCESSES
	MANAGE_ACCOUNTS
	MODIFY_AUDIO_SETTINGS
	NFC
	PERSISTENT_ACTIVITY
	READ_HISTORY_BOOKMARKS
	READ_PROFILE
	READ_SOCIAL_STREAM
	READ_SYNC_SETTINGS
	READ_SYNC_STATS
	READ_USER_DICTIONARY
	RECEIVE_BOOT_COMPLETED
	REORDER_TASKS
	RESTART_PACKAGES
	SET_ALARM
	SET_TIME_ZONE
	SET_WALLPAPER
	SET_WALLPAPER_HINTS
	SUBSCRIBED_FEEDS_READ
	SUBSCRIBED_FEEDS_WRITE
	USE_CREDENTIALS

Table 4: (cont.)

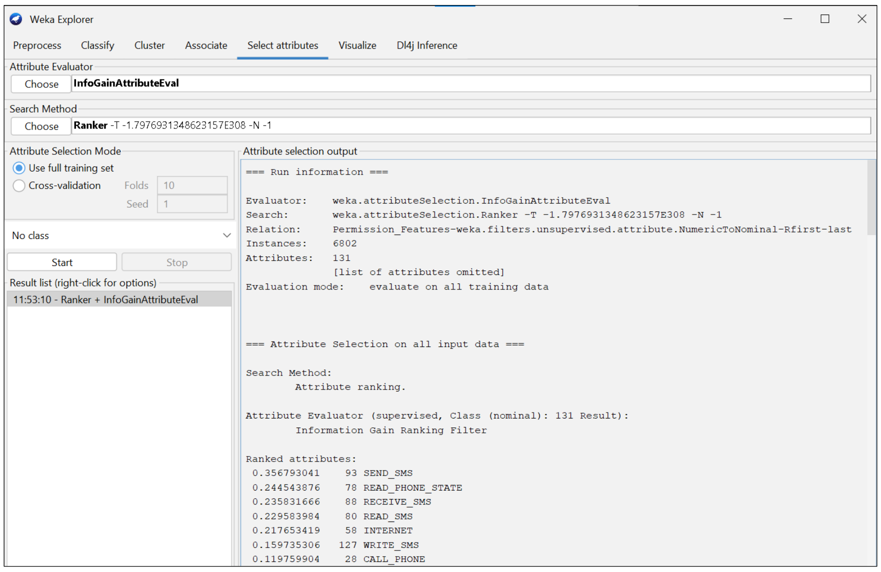
Permission Protection Level	List of Features
Normal	VIBRATE
	WAKE_LOCK
	WRITE_HISTORY_BOOKMARKS
	WRITE_PROFILE
	WRITE_SMS
	WRITE_SOCIAL_STREAM
	WRITE_SYNC_SETTINGS
	WRITE_USER_DICTIONARY
Signature	ACCESS_CHECKIN_PROPERTIES
	ACCESS_SURFACE_FLINGER
	ACOUNT_MANAGER
	BATTERY_STATS
	BIND_ACCESSIBILITY_SERVICE
	BIND_APPWIDGET
	BIND_DEVICE_ADMIN
	BIND_INPUT_METHOD
	BIND_REMOTEVIEWS
	BIND_TEXT_SERVICE
	BIND_VPN_SERVICE
	BIND_WALLPAPER
	BRICK
	BROADCAST_PACKAGE_REMOVED
	BROADCAST_SMS
	BROADCAST_WAP_PUSH
	CALL_PRIVILEGED
	CHANGE_CONFIGURATION
	CHANGE_COMPONENT_ENABLED_STATE
	CLEAR_APP_CACHE
	CLEAR_APP_USER_DATA
	CONTROL_LOCATION_UPDATES
	DELETE_CACHE_FILES
	DELETE_PACKAGES
	DEVICE_POWER
	DIAGNOSTIC
	DUMP
	FACTORY_TEST
	FORCE_BACK
	GLOBAL_SEARCH
	HARDWARE_TEST
	INJECT_EVENTS
	INSTALL_LOCATION_PROVIDER
	INSTALL_PACKAGES
	INTERNAL_SYSTEM_WINDOW
	MANAGE_APP_TOKENS
	MASTER_CLEAR
	MODIFY_PHONE_STATE
	MOUNT_FORMAT_FILESYSTEMS
	MOUNT_UNMOUNT_FILESYSTEMS
	READ_FRAME_BUFFER
	READ_INPUT_STATE

Table 4 : (cont.)

Permission Protection Level	List of Features
Signature	READ_LOGS
	REBOOT
	SET_ACTIVITY_WATCHER
	SET_ALWAYS_FINISH
	SET_ANIMATION_SCALE
	SET_DEBUG_APP
	SET_ORIENTATION
	SET_POINTER_SPEED
	SET_PREFERRED_APPLICATIONS
	SET_PROCESS_LIMIT
	SET_TIME
	SIGNAL_PERSISTENT_PROCESSES
	STATUS_BAR
	SYSTEM_ALERT_WINDOW
	UPDATE_DEVICE_STATS
	WRITE_APN_SETTINGS
	WRITE_GSERVICES
	WRITE_SECURE_SETTINGS
	WRITE_SETTINGS

4.4 Feature Selection

Feature selection processes have been carried out to identify the top ranked 10 features and 20 features from the 130 permission features. To identify the top ranked features using the full training dataset, 130 permission features were executed in the WEKA tool. The 'Select Attributes' function has been chosen for the feature selection process. The Information Gain attribute were applied to ensure that the most discriminating features were selected. Figure 6 shows the feature selection process using the WEKA tool. Once the feature selection process is done, the list of capabilities features is chosen by manual judgment. The top ranked of 10 features and top ranked of 20 features were selected and tabled for Convolutional Neural Networks and Artificial Neural Networks model classification by processing the algorithm as shown in Table 5 and Table 6.



Attribute Evaluator: InfoGainAttributeEval

Search Method: Ranker -T -1.7976931348623157E308 -N -1

Attribute Selection Mode: [✓ Use full training set] Cross-validation Folds 10 Seed 1

No class

Result list (right-click for options): 11:53:10 - Ranker + InfoGainAttributeEval

Attribute selection output

```
=== Run information ===

Evaluator:    weka.attributeSelection.InfoGainAttributeEval
Search:       weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation:     Permission_Features-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last
Instances:    6802
Attributes:   131
              [list of attributes omitted]
Evaluation mode:    evaluate on all training data



=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 131 Result):
        Information Gain Ranking Filter

Ranked attributes:
 0.356793041    93 SEND_SMS
 0.244543876    78 READ_PHONE_STATE
 0.235831666    88 RECEIVE_SMS
 0.229583984    80 READ_SMS
 0.217653419    58 INTERNET
 0.159735306   127 WRITE_SMS
 0.119759904    28 CALL_PHONE
```

Figure 6: The features selection process using WEKA tool

Table 5: Selected Top 10 Ranked Features

Selected Features	Information Gain (IG) Value	Protection Level
SEND_SMS	0.356793041	Dangerous
READ_PHONE_STATE	0.244543876	Dangerous
RECEIVE_SMS	0.235831666	Dangerous
READ_SMS	0.229583984	Dangerous
INTERNET	0.217653419	Normal
WRITE_SMS	0.159735306	Normal
CALL_PHONE	0.119759904	Dangerous
ACCESS_NETWORK_STATE	0.106684474	Normal
RECEIVE_BOOT_COMPLETED	0.104624444	Normal
READ_CONTACTS	0.101539954	Dangerous

Table 6: Selected Top 20 Ranked Features

Selected Features	Information Gain (IG) Value	Protection Level
SEND_SMS	0.356793041	Dangerous
READ_PHONE_STATE	0.244543876	Dangerous
RECEIVE_SMS	0.235831666	Dangerous
READ_SMS	0.229583984	Dangerous
INTERNET	0.217653419	Normal
WRITE_SMS	0.159735306	Normal
CALL_PHONE	0.119759904	Dangerous
ACCESS_NETWORK_STATE	0.106684474	Normal
RECEIVE_BOOT_COMPLETED	0.104624444	Normal
READ_CONTACTS	0.101539954	Dangerous
BIND_DEVICE_ADMIN	0.078817724	Signature
RESTART_PACKAGES	0.075427169	Normal
WRITE_APN_SETTINGS	0.057685673	Signature
WRITE_CONTACTS	0.057490069	Dangerous
WRITE_HISTORY_BOOKMARKS	0.054544365	Normal
READ_HISTORY_BOOKMARKS	0.049460075	Normal
WRITE_EXTERNAL_STORAGE	0.049106294	Dangerous
ACCESS_WIFI_STATE	0.048171406	Normal
MOUNT_UNMOUNT_FILESYSTEMS	0.044971003	Signature
READ_LOGS	0.037499214	Signature

4.5 Classification Algorithm

The classification algorithm is used to categorize each byte of data in a dataset into one of several predefined groups. The performance of Android botnet detection using the extracted features and selected features were evaluated on the Convolutional Neural Network and Artificial Neural Network classifiers. In total there were six types of classification algorithm models with different types of features were trained on each classification algorithm, Convolutional Neural Network and Artificial Neural Network. 10-Fold Cross Validation was used and therefore it divides the dataset into ten parts, known as "folds," and holds each part in turn before averaging the results. As a result, each data point in the dataset is tested and trained nine times. The classification models for Android Botnet Detection using Convolutional Neural Network and Artificial Neural Network are shown in Table 7.

Table 7: The Classification Algorithms Model

Classification Algorithm	Convolutional Neural Network	Artificial Neural Network
Categories of Classification Model using different features	130 Permission Features	130 Permission Features
	Dangerous Permission Feature	Dangerous Permission Feature
	Normal Permission Feature	Normal Permission Feature
	Signature Permission Feature	Signature Permission Feature
	Top 10 ranked features	Top 10 ranked features
	Top 20 ranked features	Top 20 ranked features

4.6 Result and Discussion

Based on the Android Botnet attack detection using Convolutional Neural Network and Artificial Neural Network classifier, the detection results have been obtained for each category of classification model. This section discusses the result of the experiment and performed some analysis for performance metric of the datasets in terms of Accuracy, Precision, Recall, F1-Score, True-Positive and False-Positive.

Table 8: Performance Metrics for CNN and ANN using 130 Permission

	Accuracy	Precision	Recall	F1-Score	True-Positive	False-Positive
CNN	95.44%	0.954	0.954	0.954	0.954	0.071
ANN	96.35%	0.964	0.964	0.963	0.964	0.071

Based on the detection result of Table 8, the ANN classifier has higher accuracy rate with the value 96.35% while CNN classifier has 95.44% accuracy rate. The precision, recall, f1-score and true-positive rate for ANN classifier also achieves higher rate with the value 0.964, 0.964, 0.963 and 0.963 whereas for CNN classifier are 0.954, 0.954, 0.954 and 0.954. The false-positive rate for ANN and CNN classifier is same with the value 0.071.

Table 9: Performance Metrics for CNN and ANN using Dangerous Permission

	Accuracy	Precision	Recall	F1-Score	True-Positive	False-Positive
CNN	90.94%	0.912	0.909	0.906	0.909	0.200
ANN	92.94%	0.930	0.929	0.928	0.929	0.148

Based on the detection result of Table 9, the ANN classifier has higher accuracy rate with the value 92.94% while CNN classifier has 90.94% accuracy rate. The precision, recall, f1-score and true-positive rate for ANN classifier also achieves higher rate with the value 0.930, 0.929, 0.928 and 0.929 whereas for CNN classifier are 0.912, 0.909, 0.906 and 0.909. The false-positive rate for ANN is low with the value 0.148 while for CNN is 0.200.

Table 10: Performance Metrics for CNN and ANN using Normal Permission

	Accuracy	Precision	Recall	F1-Score	True-Positive	False-Positive
CNN	90.11%	0.900	0.901	0.901	0.901	0.152
ANN	93.24%	0.932	0.932	0.932	0.932	0.107

Based on the detection result of Table 10, the ANN classifier has higher accuracy rate with the value 93.24% while CNN classifier has 90.11% accuracy rate. The precision, recall, f1-score and true-positive rate for ANN classifier also achieves higher rate with the value 0.932, 0.932, 0.932 and 0.932

whereas for CNN classifier are 0.900, 0.901, 0.901 and 0.901. The false-positive rate for ANN is low with the value 0.107 while for CNN is 0.152.

Table 11: Performance Metrics for CNN and ANN using Signature Permission

	Accuracy	Precision	Recall	F1-Score	True-Positive	False-Positive
CNN	86.65%	0.866	0.867	0.859	0.867	0.278
ANN	88.05%	0.888	0.880	0.872	0.880	0.278

Based on the detection result of Table 11, the ANN classifier has higher accuracy rate with the value 88.05% while CNN classifier has 86.65% accuracy rate. The precision, recall, f1-score and true-positive rate for ANN classifier also achieves higher rate with the value 0.888, 0.880, 0.872 and 0.880 whereas for CNN classifier are 0.866, 0.867, 0.859 and 0.867. The false-positive rate for ANN and CNN classifier is same with the value 0.278.

Table 12: Performance Metrics for CNN and ANN using Top 10 Ranked Permission

	Accuracy	Precision	Recall	F1-Score	True-Positive	False-Positive
CNN	88.86%	0.889	0.889	0.884	0.889	0.229
ANN	91.50%	0.914	0.915	0.913	0.915	0.160

Based on the detection result of Table 12, the ANN classifier has higher accuracy rate with the value 91.50% while CNN classifier has 88.86% accuracy rate. The precision, recall, f1-score and true-positive rate for ANN classifier also achieves higher rate with the value 0.914, 0.915, 0.913 and 0.915 whereas for CNN classifier are 0.889, 0.889, 0.884 and 0.889. The false-positive rate for ANN is low with the value 0.160 while for CNN is 0.229.

Table 13: Performance Metrics for CNN and ANN using Top 20 Ranked Permission

	Accuracy	Precision	Recall	F1-Score	True-Positive	False-Positive
CNN	89.21%	0.891	0.892	0.889	0.892	0.204
ANN	95.22%	0.952	0.952	0.952	0.952	0.073

Based on the detection result of Table 13, the ANN classifier has higher accuracy rate with the value 95.22% while CNN classifier has 89.21% accuracy rate. The precision, recall, f1-score and true-positive rate for ANN classifier also achieves higher rate with the value 0.952, 0.952, 0.952 and 0.952 whereas for CNN classifier are 0.891, 0.892, 0.889 and 0.892. The false-positive rate for ANN is low with the value 0.073 while for CNN is 0.204.

In overall, the Artificial Neural Network classifier performs better than Convolutional Neural Network to detect Android botnet attacks by achieving higher accuracy, precision, recall, f1-score, true-positive and low false-positive value on all the above classification models as shown in Table 14. This is due to the structure of Artificial Neural Network which has a hidden layer that contains sigmoid activation function. This increases the detection rate of Android botnet attacks as the sigmoid function exists between the range (0 and 1) [12]. As a result, it is especially beneficial for models that predict probability as an output. The activation function processes a weighted sum of inputs, and the output serves as an input to the next layer. Because the probability of anything exists only between 0 and 1, the sigmoid is the best choice. Due to the inherent complexity of CNN, the large number of layers and the massive amounts of data required. Thus, the detection performance metrics of Convolutional Neural Network is lesser compared to Artificial Neural Network.

Table 14: Comparison of the Detection Result for All Classification Algorithm Models

Types of Features		Accuracy	Precision	Recall	F1-Score	True-Positive	False-Positive
130 Permission	CNN	95.44%	0.954	0.954	0.954	0.954	0.071
	ANN	96.35%	0.964	0.964	0.963	0.964	0.071
Dangerous Permission	CNN	90.94%	0.912	0.909	0.906	0.909	0.200
	ANN	92.94%	0.930	0.929	0.928	0.929	0.148
Normal Permission	CNN	90.11%	0.900	0.901	0.901	0.901	0.152
	ANN	93.24%	0.932	0.932	0.932	0.932	0.107
Signature Permission	CNN	86.65%	0.866	0.867	0.859	0.867	0.278
	ANN	88.05%	0.888	0.880	0.872	0.880	0.278
Top 10 Ranked Permission	CNN	88.86%	0.889	0.889	0.884	0.889	0.229
	ANN	91.50%	0.914	0.915	0.913	0.915	0.160
Top 20 Ranked Permission	CNN	89.21%	0.891	0.892	0.889	0.892	0.204
	ANN	95.22%	0.952	0.952	0.952	0.952	0.073

5. Conclusion

Artificial Neural Network (ANN) algorithm and Convolutional Neural Network (CNN) algorithm managed to detect the Android botnet detection on the features of selected dataset. The 1929 ISCX botnet dataset and 4873 benign applications were used in this study. In this experiment, the total of six classification algorithm models with different categories of features have used to detect the Android botnet attacks which are 130 static permission, dangerous permission, normal permission, signature permission, top 10 ranked permission and top 20 ranked permission. The experiment has constructed with the assist of WEKA tool in order obtain the performance metric for the Android botnet attack detection. In this study, the ANN classifier achieves the higher detection performance for all classification algorithm models compared to CNN in the term of accuracy, precision, recall, f1-score, true-positive and false-positive. The highest Android botnet detection accuracy that achieved by ANN classifier is using 130 static permission features with the value of 96.35% while for CNN is 95.44%. The precision, recall, f1-score and true-positive rate value for ANN are 0.964, 0.964, 0.963 and 0.963 whereas for CNN classifier are 0.954, 0.954, 0.954 and 0.954. The false-positive rate for ANN and CNN classifier is same with the value 0.071. This shows ANN is a more superior algorithm for Android botnet detection attack compared to CNN.

Acknowledgment

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support and encouragement throughout the process of conducting this study.

References

[1] Hojjatinia, S., Hamzenejadi, S., & Mohseni, H. (2020, August). Android botnet detection using convolutional neural networks. In 2020 28th Iranian Conference on Electrical Engineering (ICEE) (pp. 1-6).

[2] Gaonkar, S., Dessai, N. F., Costa, J., Borkar, A., Aswale, S., & Shetgaonkar, P. (2020, February). A survey on botnet detection techniques. In 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE) (pp. 1-6).

[3] Yerima, S. Y., & Alzaylaee, M. K. (2020, June). Mobile botnet detection: A deep learning approach using convolutional neural networks. In 2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA) (pp. 1-8).

[4] Alqatawna, J. F., & Faris, H. (2017, October). Toward a detection framework for android botnet. In 2017 International Conference on New Trends in Computing Sciences (ICTCS) (pp. 197-202).

[5] Anwar, S., Zain, J. M., Inayat, Z., Haq, R. U., Karim, A., & Jabir, A. N. (2016, August). A static approach towards mobile botnet detection. In 2016 3rd International Conference on Electronic Design (ICED) (pp. 563-567).

[6] Joshi, S., & Abdelfattah, E. (2020, October). Efficiency of Different Machine Learning Algorithms on the Multivariate Classification of IoT Botnet Attacks. In 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) (pp. 0517-0521).

[7] Lê, N. C., Nguyen, T. M., Truong, T., Nguyen, N. D., & Ngô, T. (2020, October). A Machine Learning Approach for Real Time Android Malware Detection. In 2020 RIVF International Conference on Computing and Communication Technologies (RIVF) (pp. 1-6).

[8] Yerima, S. Y., Sezer, S., McWilliams, G., & Muttik, I. (2013, March). A new android malware detection approach using bayesian classification. In 2013 IEEE 27th international conference on advanced information networking and applications (AINA) (pp. 121-128).

[9] Xing, Y., Shu, H., Zhao, H., Li, D., & Guo, L. (2021). Survey on botnet detection techniques: classification, methods, and evaluation. Mathematical Problems in Engineering, 2021.

[10] Kirubavathi, G., & Anitha, R. (2018). Structural analysis and detection of android botnets using machine learning techniques. International Journal of Information Security, 17(2), 153-167.

[11] Chen, S. C., Chen, Y. R., & Tzeng, W. G. (2018, August). Effective botnet detection through neural networks on convolutional features. In 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) (pp. 372-378).

[12] Yerima, S. Y., Alzaylaee, M. K., & Shajan, A. (2021). Deep learning techniques for android botnet detection. Electronics, 10(4), 519.

[13] Manifest.Permission. 2022. AndroidDevelopers – Permission/Feature [online] available at: <https://developer.android.com/reference/android/Manifest.permission>

[14] UNB Canadian Institute for. 2014. GitHub – AndroidBotnetDataset [online] Available at: < https://www.unb.ca/cic/datasets/android-botnet.html>

[15] Mahindru, Arvind (2020), "Android permissions dataset, Android Malware and benign Application Data set (consist of permissions and API calls)", Mendeley Data, V3, doi: 10.17632/b4mxg7ydb7.3