This is the author accepted version of: Seraj, S., Pimenidis, E., Pavlidis, M., Kapetanakis, S., Trovati, M., Polatidis, N. (2023). BotDroid: Permission-Based Android Botnet Detection Using Neural Networks. In: Iliadis, L., Maglogiannis, I., Alonso, S., Jayne, C., Pimenidis, E. (eds) Engineering Applications of Neural Networks. EANN 2023. Communications in Computer and Information Science, vol 1826. Springer, Cham.
The final published version can be found at: https://doi.org/10.1007/978-3-031-34204-2_7

BotDroid: Permission-based Android Botnet Detection Using Neural Networks

Saeed Seraj[1], Elias Pimenidis[2], Michalis Pavlidis[1], Stelios Kapetanakis[3], Marcello Trovati[4], Nikolaos Polatidis[1]

[1]School of Architecture, Technology and Engineering, University of Brighton, BN2 4GJ, Brighton, U.K
[2]Department of Computer Science and Creative Technologies, University of the West of England, BS16 1QY, Bristol, U.K
[3]Distributed Analytics Solutions, 17 Fawe Street, London, E14 6FD, U.K
[4]Department of Computer Science, Edge Hill University, Ormskirk, L39 4QP, U.K

{S.Seraj@Brighton.ac.uk, Elias.Pimenidis@uwe.ac.uk, M.Pavlidis@Brighton.ac.uk, N.Polatidis@Brighton.ac.uk, Marcello.Trovati@edgehill.ac.uk, Stelios@distributedanalytics.co.uk}

Abstract. Android devices can now offer a wide range of services. They support a variety of applications, including those for banking, business, health, and entertainment. The popularity and functionality of Android devices, along with the open-source nature of the Android operating system, have made them a prime target for attackers. One of the most dangerous malwares is an Android botnet, which an attacker known as a botmaster can remotely control to launch destructive attacks. This paper investigates Android botnets by using static analysis to extract features from reverse-engineered applications. Furthermore, this article delivers a new dataset of Android apps, including botnet or benign, and an optimized multilayer perceptron neural network (MLP) for detecting botnets infected by malware based on the permissions of the apps. Experimental results show that the proposed methodology is both practical and effective while outperforming other standard classifiers in various evaluation metrics.

Keywords: Android Malware detection, Botnets, Neural Networks, New dataset

1 Introduction

Today, Android is one of the most well-known operating systems. It has millions of applications that are distributed through accredited or unofficial distributors. As a result, it is one of the most common targets for malicious cyber-attacks. The Play Store on Android is not very restrictive, making it simple to install malicious apps. Botnet applications are classified as malware because they can be distributed through these

stores and downloaded by unlucky users onto their smartphones. Botnets are among the most dangerous hacking techniques used on the internet today. Botnet developers frequently target smartphone users to install malicious tools and target a larger number of devices. This is frequently done to gain access to sensitive data such as credit card numbers or to cause damage to individual hosts or organisational resources through denial of service (DDoS) attacks. [1] [2].

Botnet attacks have become a threat and risk to network and internet security in recent years. They include several malicious activities in network traffic. A botnet is made up of separate robot and network components. The botmaster programmes and builds the bot for specific purposes using computers known as zombies. In the network, these computers are clearly breaking the law. Botnets are extremely widespread and can affect millions of computers. Botnets are networks made up of personal computers and smart devices known as bots. One or more attackers, known as botmasters, oversee these bots, and their goal is to carry out malicious activities. In other words, bots contain harmful software that runs on host computers and enables the botmaster to remotely command and control the system [3]. Moreover, the popularity and adoption of Android smartphones have attracted malware authors to spread the malware to smartphone users. Malware on smartphones can take the form of Trojans, viruses, worms, or mobile botnets. Mobile botnets, also known as Android botnets, are more dangerous because they pose serious threats by stealing user credentials, sending spam, and launching distributed denial of service (DDoS) attacks. A mobile botnet is defined as a collection of compromised mobile smartphones that are controlled by a botmaster via a command and control (C&C) channel and used to carry out malicious activities [4].

Although numerous studies have been conducted to detect Android botnet attacks, classification accuracy can still be improved. Insufficient or smaller data in the experiments results in lower accuracy. Machine learning is incapable of handling large amounts of unstructured data because it typically requires structured data and uses traditional algorithms. The small size of the dataset is also to blame for Android botnet detection's poor performance. Because the size of the sample data collection is limited, the confidence in the estimate decreases and the uncertainty increases, resulting in lower precision. More data is always a good idea when it comes to achieving the high efficacy of Android botnet detection. Furthermore, the use of untrained data affects an effect on the detection of Android botnets. Trained data is the most important and primary data that machines use to learn and predict. Increased training data provides more information and assist in better user fit [5].

As a result, according to the explanation provided, there is an urgent need to develop new methods for defeating mobile botnets. Because of the popularity of Android mobile devices, the goal of this paper is to propose an innovative method for detecting botnets in Android-based devices. This paper aims to produce a new mobile botnet classification/detection based on permissions. For this purpose, we have created and introduced a new dataset based on permissions that are described in detail in section 3. Moreover, the botnet dataset is a classification dataset that only includes legitimate Android apps and botnets. We have created an optimised multilayer perceptron neural network (MLP) that is highly accurate at detecting botnets.

The contributions of the paper are as follows:

- We deliver a novel dataset with 454 permissions as features to discover Botnets through the Android operating system.

- We propose an optimized multilayer perceptron neural network (MLP) to detect Android Botnets which can detect botnets with very high accuracy.

The rest of the paper is organized as follows: Section 2 is the research background, Section 3 is the related work, section 4 describes the dataset, section 5 explains the proposed method, section 6 delivers the experimental evaluation and section 7 contains the conclusions.

2 Background

2.1 Overview of Android Botnet

Botnets are a type of malware that enables an attacker to gain control of a victim's computer. The botmaster, C&C server, and bot-infected machines are common botnet components. The botnet is designed to infect mobile phones or computers and make them under the control of botnet owners or the "Botmaster". Botmasters are those who operate the command and control of botnets to attack the target via a communication channel, such as HTTP, Internet Relay Chat (IRC), or peer-to-peer (P2P). The botmaster will use a botnet to attack the victim in a variety of ways, including denial of service (DDoS) attacks, spamming, malware and advertisement distribution, espionage, hosting malicious applications, and other activities. The overview of a Botnet is demonstrated in figure 1.

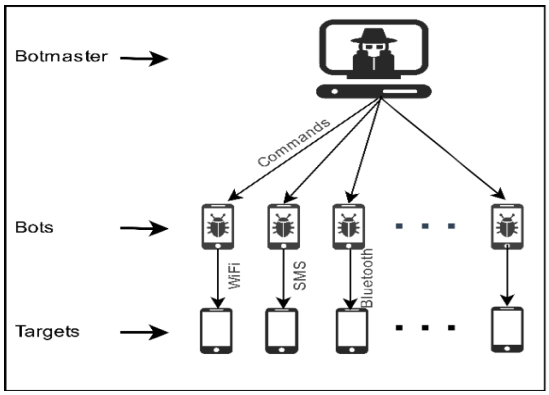

Fig. 1 Overview of a Botnet structure

2.2 Types of Botnets

A botnet includes three types of programmes:

A. Server programmes: These programmes are located on the command-and-control server and are used to control infected computers or bots.
B. Client programme: These are programmes installed on infected computers while they wait for control instructions.

C. Malicious programme: These are the software or programmes, also known as malware, which is used over the Internet to infect or compromise vulnerable computers.

Communication is the most important aspect of a botnet. The command-and-control server continues to communicate with bots, instructing them to engage in malicious behaviour. The bots, in turn, continue to wait for instructions, perform the tasks assigned to them, and send the collected data to the command-and-control server [25].

2.3 Botnet lifecycle

In general, botnets have four main phases in their lifecycle:

A. Phase Of Spread and Infection: Botmasters will employ various methods and techniques to infect new targets and transform them into new bots. After infecting the target, it will run a script or shell code and install itself on the victim machine.
B. Phase Of Command & Control: The command and control (C&C) mechanism create a communication interface between bot-bot, C&C servers-bots, and C&C servers-bot master. Command and control mechanisms are classified into three types: centralised, decentralised, and unstructured.
C. Phase Of Attack: The botnet is a collection of malicious activities that spread throughout computer networks. DDoS attacks, spamming, spreading malware and advertisements, espionage, and hosting malicious applications and activities are just a few examples of attacks.
D. Phase Of Destruction: After performing malicious activities, botmasters may destruct a portion of the botnet [15].

2.4 Botnet Attacks

Botnet attacks are typically carried out by a group of hackers, and the owner has no idea that he or she is on the victim list. Botnets are currently classified into five types based on the Command and Control (C&C) channel. Because the programme is developed by the methods and techniques employed, the botnets are divided into these categories. They are as follows:

A. IRC Botnet (Internet Relay Chat): An IRC botnet is created by using a centralised system to monitor the victim to perform malicious activities, and the targeted bots are controlled by the main C&C channel.
B. P2P Botnet (Peer to Peer): It is accomplished using P2P protocols and a decentralised system with a network of nodes that keeps it alive, containing the attacked bots as well as all relevant data transmission.
C. HTTP Botnet: An HTTP Botnet is a centralised system-based structure that conducts attacks via the HTTP protocol. The bots use a specific URL and IP address specified by the main botmaster as the C&C server. These hacking attempts are carried out for financial theft.

D. Mobile Botnet: This attack makes use of mobile phone sharing, Bluetooth technology, and text messaging. The botmaster can easily access the data using this method via the C&C Channel.
E. Botnet Cloud: This is a very difficult task, so the botmaster creates and manages the bots using the cloud service, putting the bots at significant risk of being discovered.

3 Related works

This section describes various machine learning-based Android malware analysis techniques that have been proposed in the literature. To identify android malware, three approaches have been proposed: static, dynamic, and hybrid.

3.1 Static techniques

Static techniques use few resources and are quick and secure. However, they are unable to decipher malware that has been encrypted or obfuscated. Most static methods frequently produce false positives and are unable to deal with unidentified malware. Therefore, static analysis may need to be combined with other security models for effective malware detection.

In this proposed research work, a security application is developed which scans all applications installed and identifies probable harmful applications on the user's smartphone. It organises all permissions on each application into predefined categories. The risk factor of the respective application is calculated based on the permission category. If the risk factor/score exceeds the predetermined threshold, the user is notified of the application's risk. This will inform users about applications that can exploit personal information stored on their devices in real-time [6]. Another permission-based research presents the Android botnet attack detection using deep learning algorithms, Convolutional Neural Networks (CNN) and Artificial Neural Networks (ANN) using different categories of permission features [26]. In another work, Android Botnets are investigated using static analysis to extract potential features from the source code of the applications after they have been reverse-engineered. To identify such malicious applications, efficient machine-learning models are then developed using the features. The study also suggests a new set of features for using the target mobile to access resources [1]. Another similar work to the previous paper introduces an approach to detect botnet Android mobile apps by using source code mining. Several examples of malicious and non-malicious apps analyse the source code using reverse engineering and data mining techniques. To build datasets, they employ two methods. In the first, they build several datasets by text mining the source code, and in the second, they create one dataset by extracting source code metrics using an open-source tool [2]. This study uses similar features to the previous one. They propose a system for detecting Android botnets using automated text mining of manifest files obtained from apps in this paper. The proposed method extracts the features from manifest files using NLP techniques, and a deep learning-based classification model is used to detect botnet applications [27]. This paper introduces a new classification for mobile botnets based on smartphone

permissions and Application Programming Interface (API) calls. The Drebin dataset [24] is utilised as the training dataset for this classification, which is created using static analysis in a controlled lab setting [7]. In this study, they suggest a static method for detecting mobile botnets. Using a machine learning algorithm and a combination of MD5, permissions, broadcast receivers, and background services, this technique can identify applications that can be used to create mobile botnets. In this method, android application features are extracted and used to create a machine-learning classifier for identifying mobile botnet attacks [8]. In another one, they present a novel approach for identifying Android botnet applications that rely on Android permissions and convolutional neural networks (CNNs). They also proposed a novel way to represent each application as an image that is constructed based on the co-occurrence of permissions given to that application, being the first developed method that applies CNNs for this purpose. A binary classifier that is trained using these images is the proposed CNN [9]. This paper suggests a new method for identifying mobile botnets based on features taken from images and a manifest file. The method uses a Histogram of Oriented Gradients and byte histograms obtained from images representing the app executable and combines these with features obtained from the manifest files. Then feature selection is used to choose the best features for classification using machine learning algorithms [10]. In a research work with similar features, they present Bot- IMG, a framework for machine learning-based image-based visualisation and Android botnet detection. Additionally, they used the ISCX botnet dataset [23] to assess the Bot-IMG framework's effectiveness. They specifically use Autoencoders in with traditional machine learning classifiers to implement an image-based detection method using a Histogram of Oriented Gradients (HOG) as feature descriptors within the framework [11]. The new risk assessment method that focuses on GPS exploitation for Android botnet detection is proposed to assess the level of risk connected to each app in terms of privacy, financial, and smartphone system risk. Static analysis using feature set permission and API calls served as the foundation for the evaluation. Using a quantitative calculation model, it was possible to distinguish between benign and botnet apps [12]. And finally, A comparison of deep learning techniques for Android botnet detection using 6802 Android applications made up of 1929 botnet applications from the ISCX botnet dataset is presented in this paper. Using 342 static features derived from the applications, they assess the performance of several deep learning techniques, including CNN, DNN, LSTM, GRU, CNN-LSTM, and CNN-GRU models [13].

3.2 Dynamic techniques

The application is executed on an Android platform in dynamic approaches, and all related system calls and network traffics are monitored. Malware is detected based on its runtime behaviour and interactions with the system. Dynamic methods can deal with malware that has been obfuscated or encrypted. They outperform static analysis in detecting both known and unknown malware. However, they are slow, resource-intensive, and vulnerable due to the limitation of code reachability. As a result, they may be dangerous at times.

To discover specific trends and characteristics relating to botnet behaviour, this paper analyses Android malware. A thorough literature review of well-known Android malware apps helps identify the trends and characteristics of botnets. The Android

Botnet Discovery Process and the Android Botnet Development Model are then used to further examine the identified characteristics. The frequently recognised trends and characteristics help in both the understanding of Android botnet operations and the potential identification of an Android bot [14]. In this research work, to identify potential malware in Android applications, they created a system called ABIS (Android Botnet Identification System). Their system learns the characteristics of each Android botnet family from the dataset offered by the University of New Brunswick to identify the Android botnets [15]. In another one, to accurately identify Android botnets, they suggest using an approach called Smart Self-Adaptive Learning Based Particle Swarm Optimization Support Vector Machine (SSLPSO-SVM). The SSLPSO algorithm, which is based on the PSO algorithm, simultaneously employs five different search-space scanning techniques [16]. They present an anomaly-based and host-based method for identifying mobile botnets. In the suggested method, statistical features extracted from system calls are used to identify anomalous behaviours. They were able to test the effectiveness of their method in a situation that was like reality using a self-generated dataset made up of 13 families of mobile botnets and legitimate applications [17]. "Logdog," suggests an improved log-based botnet detection method for mobile devices. Their method relies on looking through mobile device logs to find signs of botnet activity [29]. This paper introduces a novel method for botnet detection in networks. The IRC, HTTP, DNS, and P2P attacks that botnets use is compared using the proposed detection model. This model also rates the precision of botnet detection. To identify botnets, they employ network nerves, correlation, and NSA (negative selection algorithm), which is based on an artificial immune system [18]. However, their model differs from one that was previously proposed, which concentrated on 81 attributes gathered from features of network traffic. They used Weka machine learning to test ten families of Android botnets. They have 32762 instances that fall under the attack and non-attack categories [19]. At last, the research focuses on creating a cloud-based malware detection system for Android botnets. The proposed system's prototype, which offers an Android malware analysis in real-time, has been deployed. Using a botnet detection learning dataset and a multi-layered algorithm used to predict the botnet family of a specific application, the paper explains the architectural implementation of the developed system [20].

3.3 Hybrid techniques

For greater accuracy, hybrid techniques combine static and dynamic approaches. They use static analysis to analyse an application first and then use dynamic analysis to overcome both static and dynamic limitations [21]. In general, the hybrid technique usually produces the best results. Nonetheless, due to their complexity, they require a lot of resources and time. This paper aims to use machine learning methods to categorise Android applications (apps) as benign or botnet. System calls, permission requests, and API calls were analysed and classified using machine learning techniques and hybrid analysis, which combines static and dynamic analyses [4]. Also, another research focuses on developing a functional prototype of a system that uses artificial intelligence to analyse various Android application behavioural parameters. Signature-based detection techniques were also incorporated into the prototype during implementation. [22].

4 Dataset

We deliver a new dataset for Botnet detection in Android platforms. As a result, we created an Android Botnet dataset with 2713 entries. The dataset contains 454 columns, including 453 specific features and the label, which is the last column. The first row of the dataset describes column titles, and the remaining rows contain features from 2712 Android Botnets and benign applications. To do this, we downloaded 1483 benign applications from Google Play and different categories and 1229 Android Botnets. All values are in binary format, which means they are either 0 or 1. Figure 2 presents a small portion of the dataset. The entire dataset is available on Kaggle [30].

1	INTERNET	CLEAR APP CACHE	GET TASKS	CHANGE WIFI STATE	READ PHONE STATE	SYSTEM ALERT WINDOW	WRITE EXTERNAL STORAGE	CALL_ PHONE	CAMERA	READ CALL LOG	USES POLICY FORCE LOCK	WAKE UP STOKER	Risk score
2	1	1	0	1	0	1	1	0	0	0	0	0	0
3	1	1	1	1	1	1	1	0	1	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0	0	1
5	1	1	1	1	1	1	1	0	0	0	0	0	0
6	1	1	1	1	1	1	1	0	1	0	0	0	0

Fig.2 A representation of a small portion of the proposed dataset

4.1 Feature selection

Feature selection is critical in detecting mobile malware and botnets. Feature selection can help machine learning algorithms produce more accurate results by removing noise and irrelevant data from datasets. It can also reduce the runtime of machine learning algorithms during training. In this research, permissions are our features. Permissions are used to validate the system's requirements. The developer must declare permissions for use in their applications. Declared permissions are useful and effective in revealing the potential risks of installing Apps. According to [26], the protection level of the permission feature consists of three categories: Dangerous permission feature, Normal permission feature and Signature permission feature as shown in Table 1.

Protection Level	Description
Dangerous	A higher-risk permission provides access to specific application-level features to the requesting applications while posing little risk to other applications, the system, or the user.
Normal	A lower-risk permission would grant the requesting app access to sensitive user information or device control, both of 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.

Table 1. The protection level of a permission feature

4.2 Feature Extraction

VirusTotal [28] was used to decompress our botnet dataset and benign applications'.apk files. By uploading the apk file to VirusTotal, it decompiles the files to source code folders that provide detailed information about each dataset file, allowing the features to be extracted. Basic properties, permissions, activities, receivers, intent filters by action, intent filters by category, interesting strings, warnings, contents metadata, contained files by type, and contained files by extension are among the useful information. In addition, VirusTotal declares the files of the benign application to be virus-free and identifies the malware percentage of the botnet dataset files. We classified the apk files using over 70 trusted anti-malware detection engines. The android Botnet dataset includes several families, including Anserverbot, Botmaster, DroidDream, Sandroid, Wroba and Zitmo. We put all the information in a file to make the dataset usable. CSV file format, which is simple to open and process. When an app requires permission, the value in the corresponding dataset entry is 1, and when an app does not require permission, the value is 0. Based on VirusTotal's report, an Android app recognised as malware by most antivirus companies is considered risky, and the value in the label column is set to 1, indicating a Botnet. The list of Android mobile botnet families and the number of samples are listed in Table 2.

Botnet Family	Year of Discovery	Number of Samples	Type of C&C	Motivation
Anserverbot	2011	244	HTPP	Propagation of possible Malware
Bmaster	2012	6	HTPP	Financial, SMS Stealing
DroidDream	2011	362	HTPP	Data Stealing
Gemini	2010	262	HTTP	Data Stealing
Sandroid	2014	61	HTTP	Financial, Mobile Banking Attack
Wroba	2014	152	HTTP	Financial, Mobile Banking Attack
Zitmo	2012	142	SMS	Financial, SMS mobile Transaction, Authentication Number, (mTAN) stealing

Table 2. Botnet Families

5 Proposed method

We have used an MLP neural network to detect malware Botnets in our dataset. A multilayer perceptron is a good estimator in our case due to the immense flexibility of the math performed in the overall function. It is a purely mathematical system that gradually approximates complex input-output relationships with large amounts of data.

The number of input nodes must match the number of permissions in the dataset which is exactly 453. Besides, only one output node is needed even for so many input nodes since the classification here is a yes/no decision maker. For extremely powerful classification, one hidden layer is enough. The number of nodes within the hidden layer can be variable and we have found 454 as the optimum number through trial and error with extensive attempts. According to the neural network structure, data entries are multiplied by weights and subjected to an activation function. As shown in Equation 1. We used a differentiable activation function called standard logistic sigmoid for both hidden and output nodes in the MLP structure (K=1, L=1) because a gradient tells us how to modify weights. This activation function promotes successful system training while also contributing to the neural network's learning process stability. Each computational node's input is calculated using Equation 2, where N denotes the output of the preceding layer's nodes, w is the weight vector, and n denotes the number of nodes in the preceding layer. In Equation 3, the weights of a node are modified in proportion to the slope of the error function, where the target is the expected output, the learning rate, and f' is the logistic activation function's derivative. It would be unnecessary to use the logistic function's derivative expression for a given input value if we had already calculated the function's output, as shown in Equation 4.

$$f(x) = \frac{L}{1 + e^{-kx}} \quad \begin{matrix} L = 1, K = 1 \\ \rightarrow \end{matrix} \quad f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

$$preN_i = w.N = w_1N_1 + w_2N_2 + \ ... + \ w_nN_n \tag{2}$$

$$weight_{new} = \ weight_{old} + a \ x \ (target - output) \ x \ f'(input) \tag{3}$$

$$f'(x) = \frac{e^x}{(1 + e^x)^2} = f(x)(1 - f(x)) \tag{4}$$

5.1 Data Cleaning

In this section, initially, we performed a 'missing value check'. For this purpose, we performed a lost data operation and managed empty data. In most cases, the data is missing or contains null values known as NANs. There are numerous solutions to this problem: Cleaning the dataset with the NumPy library of the Python language to find

all the NAN data and specifying that there are multiple NAN data in each column, and then deciding whether the data is to be deleted or replaced from the dataset. The average data column is replaced by NAN data. NAN values are also removed from rows with labelled columns. The data should not contain any missing values. If it does, either the missing data should be removed, or some type of missing value imputation needs to be performed. Then we performed 'data type conversion'. For doing this, when using data, we must ensure that we use the correct data and ignore items that will result in errors and unrealistic results. We used the Pandas library for this, which returns the dataset's correct data types.

6 Experimental evaluation

We have developed an optimised Multilayer Perceptron (MLP) using Python and the Scikit-learn library, as described in section 5. Moreover, 5-fold cross-validation has been used throughout all experiments. Using the proposed Botnet dataset, we trained and validated our MLP neural network classifier using the Python programming language. The Numpy and Pandas libraries are required for array operations and reading data from files. The simulation is divided into four stages: defining the network's parameters, such as node numbers and learning rate, reading the Botnet dataset, training the MLP neural network with a portion of the dataset, and finally verifying the neural network with the rest of the dataset.

6.1 Evaluation metrics

We used the Python programming language with the sci-kit-learn library for the experimental analysis. We have used Accuracy, Precision, Recall, and F1 as evaluation metrics; these metrics are described in equations 5, 6, 7, and 8 respectively. True positive, true negative, false positive, and false negative are all abbreviated as TP, TN, FP, and FN, respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{8}$$

6.2 Results

This section describes the experiments and compares the proposed method to other well-known classifiers as well as the most relevant previous research in this field. For evaluating the proposed method, we used our hand-crafted dataset and selected 1229 Android botnet samples from 7 different families. All the benign samples were scanned with the VirusTotal to make sure that the benign class does not include any malware samples. The dataset consists of 2713 samples, and 5-fold cross-validation was used to evaluate the proposed method using this dataset. All experiments were performed on 64-bit Microsoft Windows 11 pro–operating system and using hardware with intel(R) Core (TM) i5-8365U @ 1.60GHz 1.90 GHz CPU, 16.00GB RAM, and an Intel UHD Graphics 620 GPU.

6.3 Comparisons with other classifiers

The following algorithms were used in the comparisons, with the default settings from the sci-kit learn library: Decision Tree, Random Forest, KNN, SVM, and Naive Bayes. The results are shown in Table 3, which compares the proposed method to other well-known classifiers based on Accuracy, Precision, Recall, F-1, and AUC using 5-fold cross-validation. To highlight the significance of this research result, a comparison is made with previous similar research. Table 4 indicates the comparison between [9], [10], [26], and [27], respectively. These comparative results show that the research method in this paper surpasses previous similar efforts.

Algorithm	Accu-racy%	Preci-sion%	Recall%	F-1%	AUC%
Decision Tree	96.50	98.36	94.14	96.20	96.37
Random Forest	98.15	97.63	98.41	98.02	98.17
K-NN	98.34	99.52	96.31	97.89	98.00
SVM	97.60	98.39	96.45	97.41	97.53
Naïve Bayes	79.18	68.53	99.59	81.19	81.00
BotDroid	**98.88**	**99.99**	**98.46**	**98.88**	**98.80**

Table 3. Comparisons with other classifiers

6.4 Comparisons with the most recent botnet detection studies

Table 4 shows the obtained results from the proposed method compared to other best researchers that have used traditional Machine Learning approaches (in terms of the used dataset, number of samples, and performance). The table indicates that the proposed method is completely successful in classifying benign and botnet applications.

Moreover, we distinguished botnet and benign applications just by utilizing given permissions, while some of the mentioned works employed more features alongside given permissions such as API calls or permissions protection level. Our promising results with high accuracy indicate that our method can effectively detect Android botnets based on only given permissions as features.

Reference	Type	Method	Accuracy%	Precision%	Recall%	F-1%
[9], 2020	Permissions	CNN	97.2	95.5	96	95.7
[10], 2022	Images and a manifest file	HOG	97.5	98.0	98.0	98.0
[26], 2022	Permissions	CNN-SVM	96.9	-	-	96.9
[27], 2022	Manifest file texts	CNN	95.44	95.4	95.4	95.4
		ANN	96.35	96.4	96.4	96.3
BotDroid	**Permissions**	**MLP**	**98.88**	**99.99**	**98.46**	**98.80**

Table 4. Comparison with other related botnet detection studies

7 Conclusions

In this paper, we employed Android permissions and an optimised multilayer perceptron (MLP) to propose a novel method to detect Android botnets. To the best of our knowledge, this is the first Android botnet detection method that applies a dataset with 454 permissions as a feature. Initially, we downloaded 2713 apk files from various categories from the Google play store and other third-party websites to create our intended dataset based on permissions. Then, reverse engineering was applied on 1483 benign and 1229 botnet applications from our hand-crafted dataset to extract the AndroidManifest.xml files that provided access to the permissions given to each application. Finally, we trained and tested a proposed MLP model using the employed dataset in a 5-fold cross-validation experiment. Based on our experiments, the proposed method outperforms several conventional ML methods in this field by achieving 98.88% accuracy and 99.99% precision. These promising results indicate that the proposed method can effectively detect Android botnets by employing the given permissions.

In the future we plan to investigate how to use permissions to detect other types of malwares such as Adware. Moreover, we aim to use Android API calls alongside the permissions to detect sophisticated various Android malwares.

References

1. Alqatawna, J. F., Ala'M, A. Z., Hassonah, M. A., & Faris, H. (2021). Android botnet detection using machine learning models based on a comprehensive static analysis approach. *Journal of Information Security and Applications*, *58*, 102735.
2. Alothman, B., & Rattadilok, P. (2017, December). Android botnet detection: An integrated source code mining approach. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)* (pp. 111-115). IEEE.

3. Hosseini, S., Nezhad, A. E., & Seilani, H. (2022). Botnet detection using negative selection algorithm, convolution neural network and classification methods. *Evolving Systems*, *13*(1), 101-115.
4. Yusof, M., Saudi, M. M., & Ridzuan, F. (2018). Mobile botnet classification by using hybrid analysis. *International Journal of Engineering and Technology (UAE)*.
5. Balasunthar, S., & Abdullah, Z. (2022). Comparison of Convolutional Neural Network and Artificial Neural Network for Android Botnet Attack Detection. *Applied Information Technology And Computer Science*, *3*(2), 32-49.
6. Kothari, S., & Joshi, S. (2020, October). Analysis of Android Applications to Detect Botnet Attacks. In *2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC)* (pp. 144-150). IEEE.
7. Yusof, M., Saudi, M. M., & Ridzuan, F. (2017, September). A new mobile botnet classification based on permission and API calls. In *2017 Seventh International Conference on Emerging Security Technologies (EST)* (pp. 122-127). IEEE.
8. 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). IEEE.
9. 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). IEEE.
10. Yerima, S. Y., & Bashar, A. (2022). A novel Android botnet detection system using image-based and manifest file features. *Electronics*, *11*(3), 486.
11. Yerima, S. Y., & Bashar, A. (2021, November). Bot-IMG: A framework for image-based detection of Android botnets using machine learning. In *2021 IEEE/ACS 18th International Conference on Computer Systems and Applications (AICCSA)* (pp. 1-7). IEEE.
12. Yusof, M., Saudi, M. M., & Ridzuan, F. Android Botnet Detection Using Risk Assessment.
13. Yerima, S. Y., Alzaylaee, M. K., & Shajan, A. (2021). Deep learning techniques for android botnet detection. *Electronics*, *10*(4), 519.
14. Pieterse, H., & Olivier, M. S. (2012, August). Android botnets on the rise: Trends and characteristics. In *2012 information security for South Africa* (pp. 1-5). IEEE.
15. Tansettanakorn, C., Thongprasit, S., Thamkongka, S., & Visoottiviseth, V. (2016, May). ABIS: a prototype of android botnet identification system. In *2016 Fifth ICT International Student Project Conference (ICT-ISPC)* (pp. 1-5). IEEE.
16. Moodi, M., Ghazvini, M., & Moodi, H. (2021). A hybrid intelligent approach to detect android botnet using smart self-adaptive learning-based PSO-SVM. *Knowledge-Based Systems*, *222*, 106988.
17. da Costa, V. G., Barbon, S., Miani, R. S., Rodrigues, J. J., & Zarpelão, B. B. (2017, May). Detecting mobile botnets through machine learning and system calls analysis. In *2017 IEEE International Conference on Communications (ICC)* (pp. 1-6). IEEE.

18. Girei, D. A., Shah, M. A., & Shahid, M. B. (2016, September). An enhanced botnet detection technique for mobile devices using log analysis. In *2016 22nd International Conference on Automation and Computing (ICAC)* (pp. 450-455). IEEE.
19. Rasheed, M. M., Faieq, A. K., & Hashim, A. A. (2020). Android Botnet Detection Using Machine Learning. *Ingénierie des Systèmes d Inf.*, *25*(1), 127-130.
20. Jadhav, S., Dutia, S., Calangutkar, K., Oh, T., Kim, Y. H., & Kim, J. N. (2015, July). Cloud-based android botnet malware detection system. In *2015 17th International Conference on Advanced Communication Technology (ICACT)* (pp. 347-352). IEEE.
21. Seraj, S., Khodambashi, S., Pavlidis, M., & Polatidis, N. (2022). HamDroid: permission-based harmful android anti-malware detection using neural networks. *Neural Computing and Applications*, 1-10.
22. Oh, T., Jadhav, S., & Kim, Y. H. (2015, October). Android botnet categorization and family detection based on behavioural and signature data. In *2015 International Conference on Information and Communication Technology Convergence (ICTC)* (pp. 647-652). IEEE.
23. Abdul Kadir, A. F., Stakhanova, N., & Ghorbani, A. A. (2015, November). Android botnets: What urls are telling us. In *International Conference on Network and System Security* (pp. 78-91). Springer, Cham.
24. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., & Siemens, C. E. R. T. (2014, February). Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss* (Vol. 14, pp. 23-26).
25. Baruah, S. (2019). Botnet detection: analysis of various techniques. *International Journal of Computational Intelligence & IoT*, *2*(2).
26. Balasunthar, S., & Abdullah, Z. (2022). Comparison of Convolutional Neural Network and Artificial Neural Network for Android Botnet Attack Detection. Applied Information Technology And Computer Science, 3(2), 32-49.
27. Yerima, S. Y., & To, Y. A deep learning-enhanced botnet detection system based on Android manifest text mining.
28. VirusTotal. Free online virus, malware and URL scanner, https://www.virustotal.com/.
29. Girei, D. A., Shah, M. A., & Shahid, M. B. (2016, September). An enhanced botnet detection technique for mobile devices using log analysis. In *2016 22nd International Conference on Automation and Computing (ICAC)* (pp. 450-455). IEEE.
30. https://www.kaggle.com/datasets/saeedseraj/botdroid-android-botnet-detection/