National College of Ireland

Android botnet detection using signature data and Ensemble Machine Learning.

MSc Academic Internship
MSc Cyber Security

Viraj Kudtarkar
Student ID: 18178499

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Viraj Kudtarkar
Student ID:	18178499
Programme:	MSc Cyber Security **Year:** 2020
Module:	MSc Internship
Supervisor:	Ross Spelman
Submission Due Date:	17/08/2020
Project Title:	Android botnet detection using signature data and Ensemble Machine Learning.
Word Count:	5120 **Page Count:** 16

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.
I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature: Viraj Kudtarkar

Date: 17 August 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Android botnet detection using signature data and Ensemble Machine Learning.

Viraj Kudtarkar
18178499

Abstract

As the use of smartphones has increased intensely in the past decade for daily activities such as socialising, banking, online shopping and communicating with friends and family. Android operating system is very popular and used universally for smartphones and tablets. Therefore, threats for this android platform is emerging very rapidly. Exploiting smartphones are comparatively easy and more effective than exploiting traditional computer systems and thus attackers started developing applications with hidden botnet capabilities. These applications use to take control of user's device without his permission to steal sensitive data or launch denial-of-service attack with the help of Command and Control (C&C) servers. There are many proposed solutions available to detect botnet application using various approaches. In this paper, I proposed a hybrid model for botnet detection using a combination of signature-based detection at initial layer to perform abrupt detection. At 2nd layer ensemble machine learning method is used to identify botnet components with the help of extracted permissions and intents via static analysis. I compared 5 machine learning classifier algorithms and selected three with highest accuracy to create ensemble model. To extract the features to prepare efficient dataset for training and testing of this machine learning model I analyse 375 applications with botnet capabilities and 1105 benign applications from CICInvesAndMal2019 dataset which is novel and publicly available for researchers by the Canadian Institute for Cybersecurity. To confirm this result, we used Virus Total as a reference point which also showed comparable results of botnet detection. In this experiment, we successfully obtain 95.4% accuracy with the Logistic Regression classifier which was slightly increased to 95.8% after assembling top three algorithms.

Keywords: Android Botnets, Ensemble Machine Learning, Signature-Based detection, Permissions, Intents, and DDoS prevention.

1 Introduction

In past few years, smartphones and tablets plays integral part in our life due to its seamless capabilities and pocket friendly structure. Therefore, these devices outplaced traditional and heavy-handed computers. These devices are used to carry out all the activities from communicating or socializing, gaming, online shopping to sensitive activities like banking or using online payment systems. All the personal and sensitive information gets exchanged between these devices and corresponding application servers; sometimes user store it in the application or device which also provoke attackers to target smartphones. Amongst the many operating system android is most popular OS used in smartphones; it rules the market with 86% share in the available applications for the first quarter of 2020 according to [4] with the 1.6 billion users worldwide [5]. As per statistics, Google Play store was hosting 2.896 million android applications in June 2020 [6]. Android is an opensource platform which is available to users with huge potentiality to develop third party applications encourages botnet writers

with evil purpose to develop more such applications. These applications can be hosted on the official google play store or other third-party stores.

Upon successful infection of device by botnet application, the attacker acts as botmaster and the infected device started acting like a zombie. Attacker gain full control of the infected device remotely without users' consent. Botmaster use command and control (C&C) mechanism to communicate with these Infected zombies [8]. These botnets are capable of launching Denial-of-Service attacks, brute-force attack, stealing information such as contact details, call logs, location information, IMEI information of the device, network operator details, voice mail numbers and even much more sensitive information such as bank transactions and passwords, credit card details from the infected device depending on the motive of the attack [9]. Botmaster also gets ability to intercept and read SMS or Email messages, send fake messages or emails without permission of the device owner which can further lead to SMS frauds or phishing attacks. Furthermore, He can carry out unwanted installations of malwares or banking trojans [14], [15]. Various botnet variants found in the year 2011 such as GoldDream, PjApps, Plakton and DroidKungFu. All of them were communicating via HTTP protocol with their C&C servers. The attack executed in 2012, using the Eurograbber variant of Zeus affected 30, 000 consumers of various banks in Europe and cause loss of nearly $36 million euros [10]. In a report published by Forbes, upstream notified that the malicious applications on play store are getting doubled every year with 55% higher transaction frauds. Last year around 43000 devices were infected by the 98000 malicious applications [7]. As per Kaspersky, 60% of android malware contains some level of botnets components [11]. Also, the mobile botnets are considered as highly reliable compared to traditional botnets as users very hardly shut down their smartphones [12], [13], [14]. These mobile botnets have three basic components which are (C&C) command and control server, infection vectors and the topology. The C&C server is the most essential component which is used for sending commands and receiving information between botmaster and bots. Infection vectors are responsible for transmission of bot binaries into the device. And the topology is for organizing the bot. Usually centralized topology is used while very few botnets use decentralized topology to formulate the zombies [14].

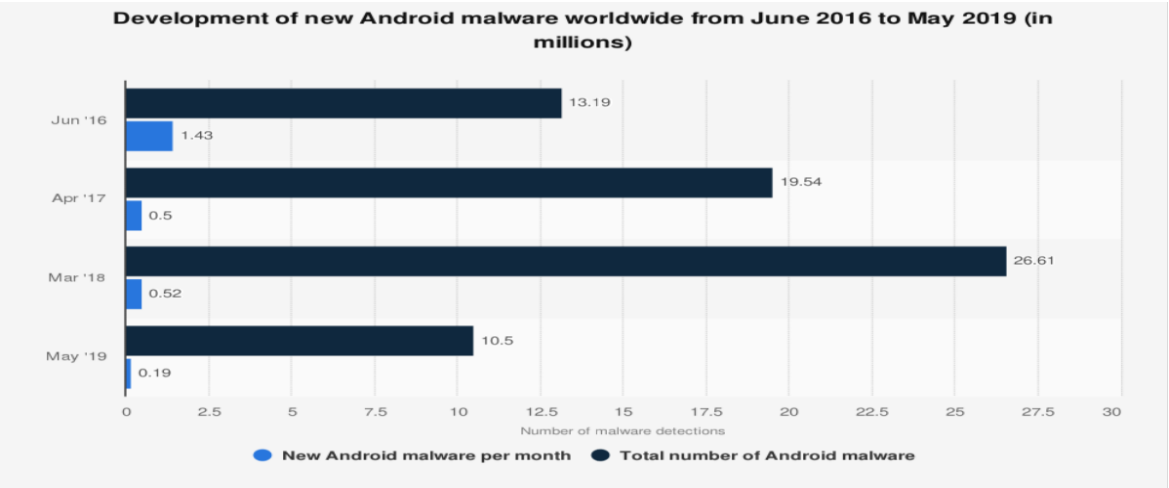


Figure 1: Development of new Android malware worldwide from June 2016 to May 2019 [16].

Considering the rapid growth of the botware applications and the immense potential they hold to exploit android smartphones it is an endless need to find more and more reliable mitigation. I have conducted this study to find a compatible solution which will help to eliminate botnet applications before installation. In this paper I proposed a two layered approach to identify android botnets using both signature-based detection at initial layer and static analysis using ensemble machine learning at 2nd layer.

This report is expected to comply with our research question: Can use of signature data and ensemble machine learning be able to detect android botnet applications.

The rest of the paper is organised in following manner: In Section 2, related works are discussed. In the 3rd section, research methodology is discussed. Section 4 presents the design specification. 5th section presents the Implementation. In section 6, evaluation and results are mentioned and final 7th section is about conclusion and future work.

2 Related Work

Identification of botnet application is very crucial for prevention of cyber-attacks in recent times. Research related to mobile application security is always evolving as this field is new and boosting rapidly. There are various approaches proposed by researchers from all over the world to identify and prevent mobile malware and botnet applications. Before developing this proposed system, an in-depth literature review was conducted to learn about already existing solution. In this section I have highlighted existing research works using many different approaches and methods to determine and restrict mobile botnets attacks. This section is divided into multiple sub-sections based on approaches used for detection. To improve the accuracy of classification this review was very imperative signature-based detection

Signature based approach for detecting applications with malicious components is pretty simple compared to other approaches. It is done by comparing the unique data collected from the application with the data which is previously analysed and stored in the database. It will help in quick determination and also to save processing power. But the minor changes in code allows bypassing signature-based analysis which is the main drawback of this method. Databased has to be updated regularly to defend against newly designed applications. Also, it is not capable to detect zero-day attacks.

Oh, Jadhav and Kim [1], has proposed a multi-layered system using signature-based detection and behavioral analysis. The system is segregated into 5 main layers and at each layer a specified task will be performed. First layer was responsible to extracts the hash value from the provided application to verify if the application was previously analysed or not by comparing it with the data stored in the database. Further it will be forwarded to next layer to collect the data which is required for behavioral analysis by conducting static and dynamic functions. At third layer this data will get parsed and converted into usable format. And finally, the decision will be made by considering on both signature and behavioural analysis. Machine Learning Approaches

In this sub-section I have discussed various approaches based on machine learning for detection of botnet applications. It also includes research related to detection of malware applications, since it has resemblance with detection of botnet detection.

2.2.1 Static Analysis Based Detection

The static analysis is performed by extracting the feature from the application without running the application by doing reverse engineering of the application. These features are used for training testing machine learning model to determine if the application is botware or benign.

Yusof, Saudi and Ridzuan [2], use android permissions and (Application Programming Interface) API calls for the classification of botnet. Researchers used 50 application to extract 5, 560 samples from the Drebin dataset for training and 800 samples are taken from Google Play Store for testing purpose. feature selection was performed to select best 16 permissions and 31 API calls which are most related with android botnet to perform classification. Random Forest, Support Vector Machine, Naïve Bayes and K-NN algorithms are used to evaluate the results. Random Forest algorithm was superior 99.4% accuracy and false positive rate of 16.1%.

In [3], Tansettanakorn, Thongprasit, Thamkongka and Visoottiviseth used similar method to develop a system named as ABIS (Android Botnet Identification System) to identify the detect botnet applications. The ABIS server is capable of connecting to the Google Play server. As per user requirements this ABIS server evaluates APK package from Play Store to analyse the requested application. The dataset containing 14 botnet families with 1,929 applications were used with 150 additional clean applications which are downloaded from the play store.

63 permissions and 1,414 API calls were used to train seven classification algorithms including Random Forest, MLP, Decision Tree, J48, SVM, SMO, Naïve Bayes and Bagging. Random Forest achieved best classification results with 96.9% of recall with 4 Permissions and 41 API calls.

3 Research Methodology

In this paper, we have proposed a unique method to identify hidden botnets from android applications by making use of signature-based detection with ensemble machine learning. In the previous sanctions we have highlighted some of the existing works which are already done by researchers to detect the botnet applications. The linear-sequential model which is also known as waterfall model was used to execute this research. This model is considered best for AI and machine learning where requirement of are nominal. We divide the entire process into small components.

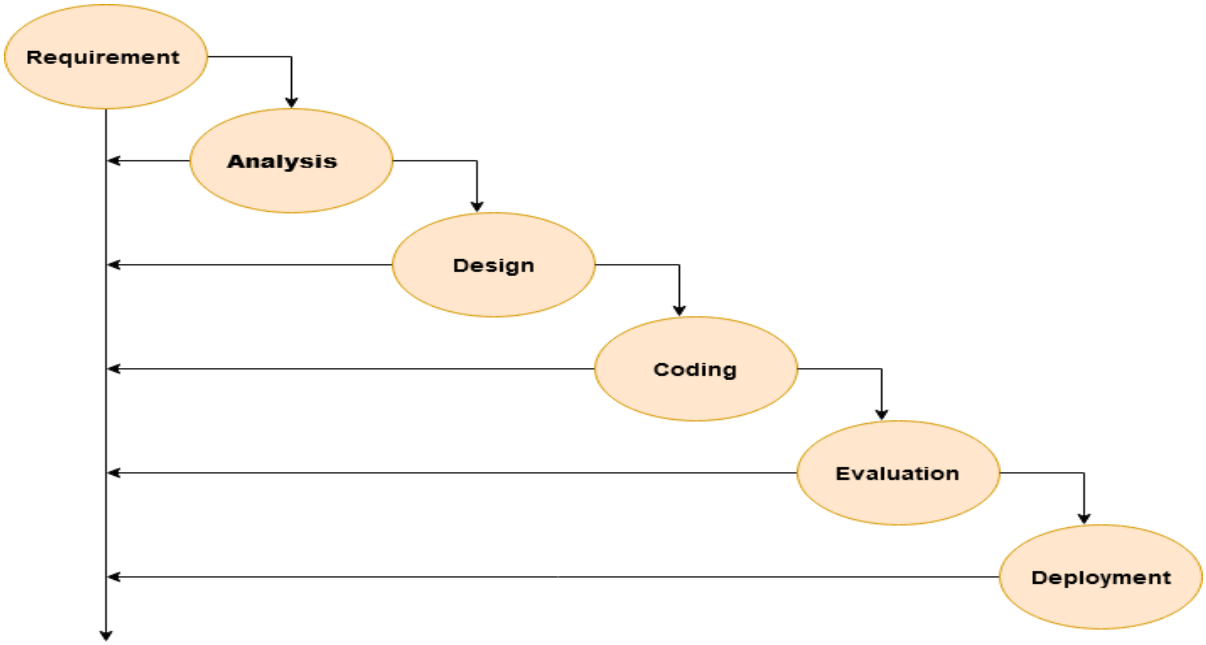


Figure 2: Waterfall Model

1. In the Requirement phase, the objectives and goals of this research are defined. Thorough literature review was done to understand the previously implemented solutions.
2. During the analysis phase, the data required for testing was collected and cleaning of the data was performed to get quality data for testing.
3. Design phase involves finalizing the features, constructing the database to store the signature values, finalising dataset and selection of classification algorithms.
4. At the coding phase actual implementation and building of the model was completed.
5. Actual evaluation of the implemented model was performed during this stage where we validated the results obtain from coding phase.
6. All the outcomes of the previous phases are documented in this phase.

We used static analysis to extract the required features from the android applications such as MD5 hash values, package names, dangerous android permission and intents. From these extracted features MD5 hash values and package names are used for signature-based

detection, where android permissions and intents are used to train and test the machine learning model which is used at second layer for classification. Similar approach is used by previous researchers [14], [17], [15], [18], [19].

3.1 Dataset

For conducting this research, we have used CICInvesAndMal2019 dataset which is openly available for researchers and published by the Canadian Institute for Cybersecurity [20]. This is very comprehensive and novel dataset which is not used in any of the research conducted to identify botnet applications. This dataset contains 426 samples of malware applications and 5,065 samples of benign applications [21]. In this study, we selected only 384 samples of botnet applications and 1105 samples of clean non-malicious applications due to the time limitations. We split the dataset into two parts for training and testing, where the 70% of data is assigned for training and was 30% is assigned for testing purpose. As these applications are malware applications and only malware category and family classification were done, we validate them by scanning via the Virus Total API. Only the applications which are identified as botnet applications are used in this research.

3.2 Data Extraction

To decompress the APK files and extract the required features from the source code of selected android applications, we tried and tested multiple reverse engineering tools like APKTool, Aapt, Simplify, Dex2 jar with JD-GUI which are available online. Finally, we use APKTool for most of the applications which is smoother compared to others. It is possible to clubbed with personalized scripts to automate the process. With the help of this tool we decoded the AndroidManifest.xml file which is present in the root directory also disassembles the DEX files to smali files to make them readable. We collected essential information such as uses−permissions and intents from manifest file where it is stored in an android application. The application gets installed successfully only upon receiving the confirmation from the user for all the requested permissions and intents. We also extracted MD5 hash values from the botnet application and package name from all the benign applications with the help of PackageManager. The data collected from above extraction methods was converted into CSV format file. Feature Selection

3.2.1 Signature Data (MD5 Hash Values and Package Names)

At first layer to perform signature-based detection we used MD5 hash values from the botnet applications and package names which are unique for each application are extracted from the benign applications respectively. We have used these features at initial level to diagnose botnet applications in our proposed method. In the beginning, we built our database of with these extracted MD5 hash values and package names. At the time of evaluation, the

application first tries to compare the hash value and package name with the existing values which are present at the database.

3.2.2 Machine Learning Model (Permissions and Intents)

Various samples of android applications both botnet as well as normal were studied and analysed thoroughly to understand the most essential features. If the higher number of features are used for prediction, then heavy processing power is required to process additional data which also takes longer time. <uses−permission> tag represents the permissions which will be requested by the application at the time of installation. For example, an application which requests combination of READ_CONTACTS and INTERNET or WRITE_SMS, SEND-SMS permission will collect all the contacts and related information stored on the device and send it to the botmaster using the internet or SMS service without users consent [22]. it is observed that botware applications seek higher amount of permissions than benign applications. Similarly, Intents indicates to the system that a specific event has occurred. In an application Intents permit binding among components internally as well as externally. Our analysis affirms that combination of requested permissions and intents have direct relations with malicious activities and are highly capable of predicting botnet applications. A comprehensive study with the frequency analysis was performed for the of both applications to select the final set of permissions and intents which are most prominent in detection of applications with botnet capabilities. Finally, the 18 features with top 15 permission and 3 intents which are prominent in detection of botnet application are selected. If the higher number of features are used for prediction, then heavy processing power is required to process additional data which also takes longer time.

Table 1:Prominent Features Used to Detect Botnets.

Feature Name	Type
READ_SMS	Permission
READ_CONTACTS	Permission
WRITE_SMS	Permission
SEND_SMS	Permission
RECEIVE_BOOT_COMPLETED	Permission
RECEIVE_SMS	Permission
WAKE_LOCK	Permission
ACCESS_NETWORK_STATE	Permission
INTERNET	Permission
ACCESS_WIFI_STATE	Permission
ACCESS_FINE_LOCATION	Permission
ACCESS_COARSE_LOCATION	Permission
CALL_PHONE	Permission
READ_PHONE_STATE	Permission
WRITE_EXTERNAL_STORAGE	Permission
POWER_CONNECTED	Intent
BATTERY_LOW	Intent
BOOT_COMPLETED	Intent

4 Design Specification

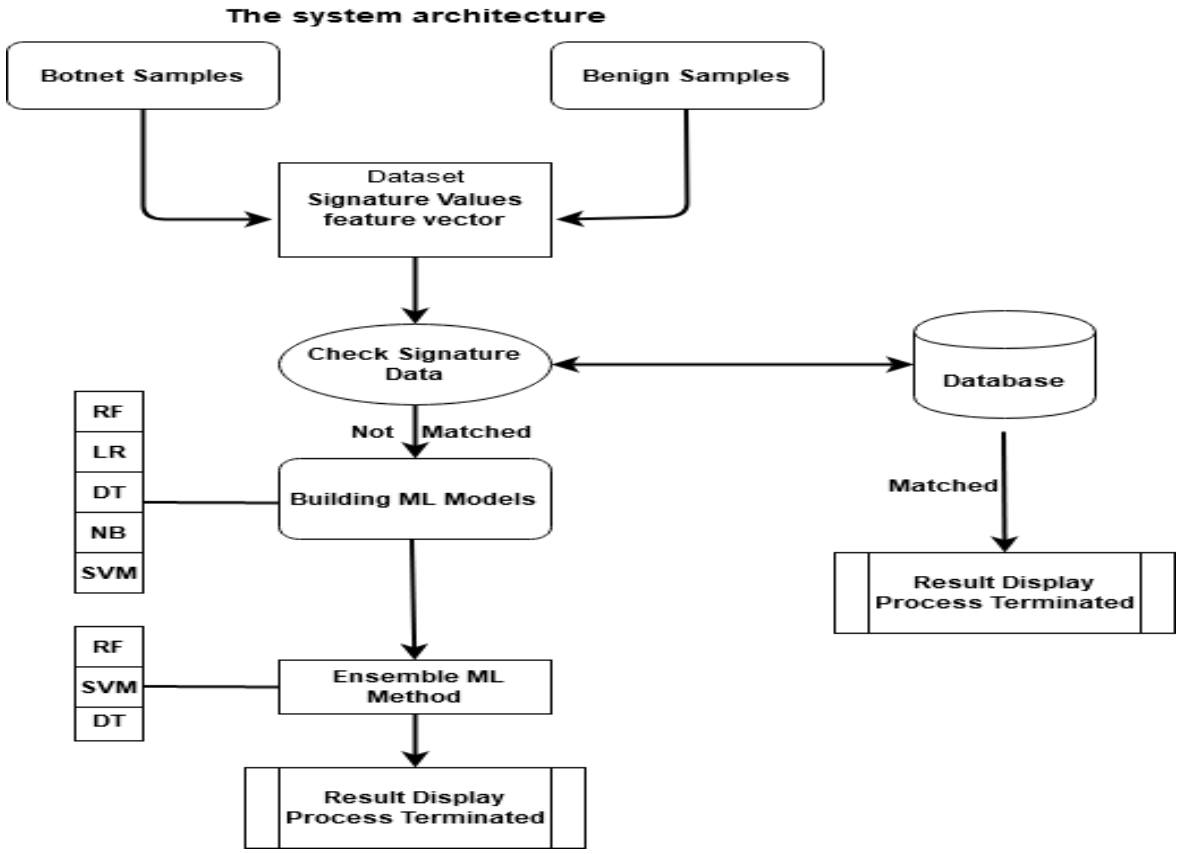


Figure 3: Application Work Flow Diagram

In this section we have presented the work flow and architecture of our proposed two layered android botnet detection model. This tool is capable of detecting applications with botnets capabilities. Figure 3 shows the flow diagram of design of the model.

4.1 Building and Training the Model

The steps include reverse engineering of botnet and normal applications to extract feature vectors. SQLite database was created and data collected for signature-based detection was stored in the database, i.e. MD5 hash values collected from botnet applications and the package names of benign applications. For machine learning based model features selection, cleaning and pre-processing of the dataset and splitting the dataset was done for training and testing purpose. In next phase, 5 different machine learning classifiers (Random Forest, Logistic regression, Decision Tree, Naive Bayes, SVM) are used on final dataset and analysed the results. Top three classifiers are selected with best accuracy to build assemble model based on voting mechanism.

4.2 Detection

Once the model is built, at the time of testing the application will first check the database for the MD5 hash values and package names, if the value is matched it will display the results and terminate the process. The new application is received for which the signature value did not matched with the values available in the database, it will pass the function to the next layer where machine learning based classification will be performed using assemble method and the result will be displayed.

5 Implementation

In this section, we have discussed about final implementation and coding to build our android detection model. The R file contains the step by step process and implementation of all steps and processes performed towards classifying and building our model were implemented with Python 3.8 using Pycharm 2020.1. All the Packages and libraries which are present in python help functions work effectively. We install all the necessary packages and libraries before calling their corresponding functions.

Building Database: Sqlite3

We used an SQLite database to store the android application's unique identifier "Package-name" and MD5 hash key-values. Database helped us in making our system more efficient by returning responses in very less time. Database contains a table named "Apps" which stores all the app's information. We have data of almost 600 application stored in our database for testing purpose. This data needs to be updated periodically to get better detection results.

Pre-processing and Cleaning of Dataset

For training of our ML models, we selected 18 features (permission and intent), which have most significant capability of distinguishing botnet applications. In pre-processing part, we removed the noise from data by deleting rows which are incomplete. Also, we dropped the unnecessary columns which are not necessary for our training process. Once the filtering process is completed, we split the data into 2 parts with 70:30 ratio, 70% for training and 30% for testing purpose.

Machine Learning Model

Before creating final model, we trained five different classifier algorithms which includes: Support Vector Machine, Logistic Regression, Random Forest, Decision Tree and Naive Bayes for our machine learning predictor. Models were trained on 70% of dataset in training process, first the model is created and trained by "model.fit()" method. "Model.predict()" is used for getting test results out of the model and returns a confusion matrix which is used for

finding the accuracy of the trained model. We extracted 3 different performance measures for our models which are Accuracy, Precision and Recall. After completion of training process, all trained models were stored in ".pkl" file format.

Assemble Model

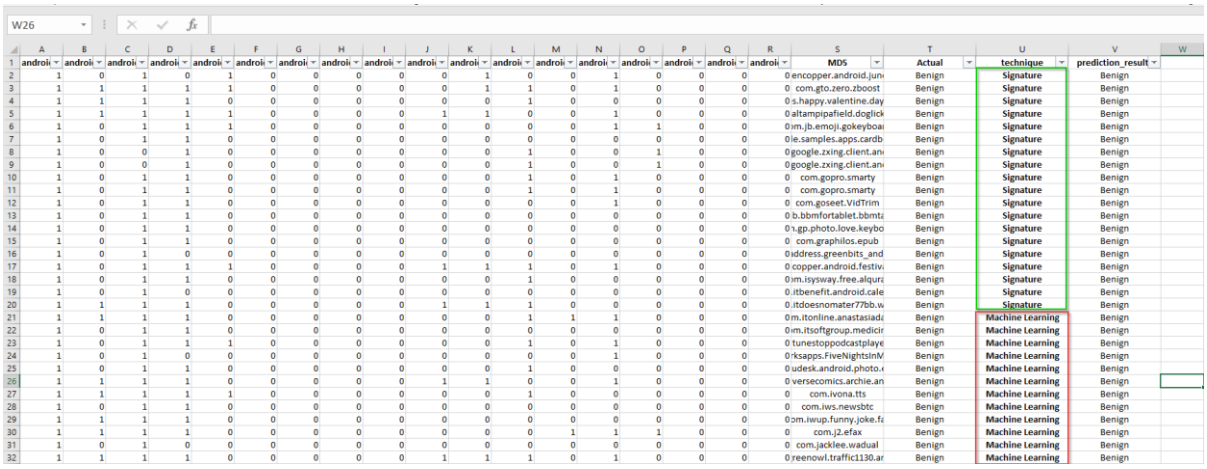
To build our final detection method we used ensemble methods in our predictor. Ensemble method is a technique in which we combine different models to produce improved results. Ensembles can produce more precise and accurate outputs compared to single algorithm. We have created voting-based ensemble model which includes three models which are Random Forest, SVM and Decision Tree. Out of five three best models were chosen on the basis of accuracy, so only top 3 model with best accuracy results were used in creation of ensembles.

6 Evaluation and Results

In this section we have represented the results received from the signature-based detection mechanism, five machine learning algorithms, and also from assemble method which we used in this research. We have also described the metrics used for evaluation.

6.1 Signature-based detection

In figure 4 we have highlighted the results received from signature-based detection with green colour. In this detection process the signature values are getting verified with the values stored in database for the identification, it gives results in less time compared to the machine learning based detection process.



	MDS	Actual	technique	prediction_result
2	encopper.android.jun	Benign	Signature	Benign
3	com.gto.zero.zboost	Benign	Signature	Benign
4	s.happy.valentine.day	Benign	Signature	Benign
5	altampipafield.doglick	Benign	Signature	Benign
6	m.jb.emoji.gokeyboa	Benign	Signature	Benign
7	le.samples.apps.cardb	Benign	Signature	Benign
8	google.zxing.client.an	Benign	Signature	Benign
9	google.zxing.client.an	Benign	Signature	Benign
10	com.gopro.smarty	Benign	Signature	Benign
11	com.gopro.smarty	Benign	Signature	Benign
12	com.goseet.VidTrim	Benign	Signature	Benign
13	b.bbmfortablet.bbmt	Benign	Signature	Benign
14	gp.photo.love.keybo	Benign	Signature	Benign
15	com.graphilos.epub	Benign	Signature	Benign
16	ddress.greenbits_and	Benign	Signature	Benign
17	copper.android.festiv	Benign	Signature	Benign
18	m.isysway.free.alqura	Benign	Signature	Benign
19	itbenefit.android.cale	Benign	Signature	Benign
20	itdoesnomater77bb.w	Benign	Signature	Benign
21	m.itonline.anastasiad	Benign	Machine Learning	Benign
22	m.itsoftgroup.medicir	Benign	Machine Learning	Benign
23	tunestoppodcastplaye	Benign	Machine Learning	Benign
24	rksapps.FiveNightsInM	Benign	Machine Learning	Benign
25	udesk.android.photo.a	Benign	Machine Learning	Benign
26	versecomics.archie.an	Benign	Machine Learning	Benign
27	com.ivona.tts	Benign	Machine Learning	Benign
28	com.iws.newsbtc	Benign	Machine Learning	Benign
29	m.iwup.funny.joke.fa	Benign	Machine Learning	Benign
30	com.j2.efax	Benign	Machine Learning	Benign
31	com.jacklee.wadual	Benign	Machine Learning	Benign
32	reenowl.traffic1130.ar	Benign	Machine Learning	Benign

Figure 4: Results from Signature based detection.

6.2 Machine Learning based detection

We have also described the metrics used for evaluation

- To calculate accuracy, we divide number of correct predictions by total predictions done.

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

- To calculate precision, we divide the number of true positive by the sum of true positive and false positive.

$$\textit{Precision} = \frac{TP}{TP + FP}$$

- To calculate Recall, we divide the number of true positive by the sum of sum of true positive and false negatives.

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

- To compare two models, we use F-Score. It will help in measuring recall and precision at same time.

$$F\text{-}measure = \frac{2*Recall*Precision}{Recall + Precision}$$

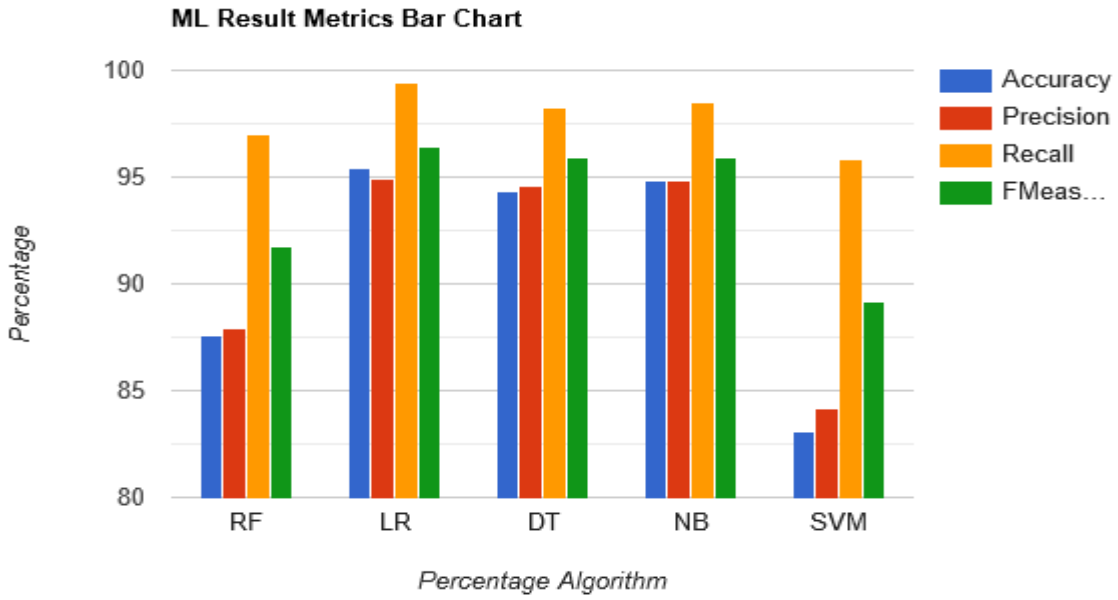


Figure 5: Bar Chart for Metrics

Table 3 shows the metrics scores given by the 5 different algorithms which are used during the testing phase.

Table 2: Metrics Chart of Different Algorithms

Algorithms Used	Accuracy %	Precision %	Recall %	F- Measure
Random Forest	87.6	87.93	97.04	91.72
Logistic Regression	95.4	94.91	99.4	96.43
Decision Tree	94.3	94.58	98.22	95.95
Naive Bayes	94.8	94.87	98.52	95.95
Support Vector Machines	83.1	84 15	95.85	89.16

After comparing metrics scores of all 5 algorithms it was observed that Logistic Regression model has the highest accuracy of 95.4% while Naive Bayes and Decision Tree are giving the 94.8% and 94.3% respectively. We did not rely on only accuracy and checked F1 score and observed that same algorithms are having good F1 score. It indicates that there are very low number of low false positives and low false negatives values and our model has identified the botnets applications correctly without having the false alarms. As we know F1 score will be considered good when it's near to 1, and the model will be considered as unsuccessful when it's near 0.

From above results we got clear idea about the three best classification algorithms out of five algorithms which we have used for training. Therefore, we used Logistic Regression, Decision Tree and Naive Bayes classifiers to build our final assemble model which was based on voting mechanism.

Table 3: Accuracy Chart of Assemble Method

Algorithms Used	Accuracy %	Precision %	Recall %	F- Measure
Assemble Method	**95.8**	**98.69**	**86.59**	**92.24**

Table 4 shows the metrics scores received from the assemble model. Where we are able to achieve the 95.8% accuracy which is very similar with the scores, we achieved from Logistic Regression classifier.

6.3 Discussion

After analysing the results, it was concluded that there is no significant difference in outcomes of Logistic Regression classifier and the assemble machine learning approach which we have proposed. However, we choose top 3 algorithms as our final model because of high accuracy and high F-score. Although, there are several studies [2], [14], [18] where researchers claim of achieving more than 98%-99% of accuracy our hybrid model also capable of detecting botnets by signature data which will help to save processing power and time for the already known botnets as well as known benign applications as we have stored package names of verified benign applications into our database. Model and methods are good as we are still able to achieve great amount of accuracy. Therefore, to improve the results to get optimal results we can suggest the improvement in dataset by adding more feature vectors with dynamic analysis. Other advance classification models or using Neural networks can help to achieve maximum results.

7 Conclusion and Future Work

To conclude this, we can say that we are able to achieve the objective of our research which is developing an android botnet detection model on signature-based detection along with assemble machine learning method by identifying the optimal classifiers. This model will successfully identify the android botnet applications at initial level to prevent the damaged caused by various cyber-attacks. The analysis outcome of results states that the selected feature vectors android permissions and intents are highly capable of detecting botnets correctly. This model has potential to provide a good accuracy and precision score with minimum false positive percentage. We tried five different machine learning classifiers from which top three classifiers are used to build the assemble model. Out of five Logistic Regression classifier achieved the highest accuracy rate of 95.4%. Our hands are tied with the time limitation for the proposed project, the algorithms are effective in detection but the feature vectors are not sufficient to achieve best results. As extracting features and analysing them in order to enhanced the outcomes was so much time consuming.

For the future work of this research we would like to suggest a cloud-based approach by signature and Machine learning methods. It should also include the android model to automatically extract only the selected and vital features from the application to achieve the maximum accuracy.

References

[1] T. Oh, S. Jadhav and Y. H. Kim, "Android botnet categorization and family detection based on behavioural and signature data," 2015 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2015, pp. 647-652, doi: 10.1109/ICTC.2015.7354630.

[2] M. Yusof, M. M. Saudi and F. Ridzuan, "A new mobile botnet classification based on permission and API calls," 2017 Seventh International Conference on Emerging Security Technologies (EST), Canterbury, 2017, pp. 122-127, doi: 10.1109/EST.2017.8090410.

[3] C. Tansettanakorn, S. Thongprasit, S. Thamkongka and V. Visoottiviseth, "ABIS: A prototype of Android Botnet Identification System," 2016 Fifth ICT International Student Project Conference (ICT-ISPC), Nakhon Pathom, 2016, pp. 1-5, doi: 10.1109/ICT-ISPC.2016.7519221.

[4] "IDC - Smartphone Market Share - OS", IDC: The premier global market intelligence company, 2020. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os/. [Accessed: 14- Aug- 2020].

[5] "Topic: Android", Statista, 2020. [Online]. Available: https://www.statista.com/topics/876/android/. [Accessed: 14- Aug- 2020].

[6] "Google Play Store: number of apps | Statista", Statista, 2020. [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. [Accessed: 14- Aug- 2020].

[7] Z. Doffman, "Beware—Millions Of Android Users Must Delete This 'Malicious' Video App Now", Forbes, 2020. [Online]. Available: https://www.forbes.com/sites/zakdoffman/2020/06/03/beware-millions-of-android-users-must-delete-this-malicious-video-app-now/#2cf42678386f. [Accessed: 14- Aug- 2020].

[8] M. La Polla, F. Martinelli and D. Sgandurra, "A Survey on Security for Mobile Devices," in IEEE Communications Surveys & Tutorials, vol. 15, no. 1, pp. 446-471, First Quarter 2013, doi: 10.1109/SURV.2012.013012.00028.

[9] R. R. Nigam, A Timeline Of Mobile Botnets, pp. 1-8, 2015.

[10] E. Kalige and D. Burkey, "A Case Study of Eurograbber: How 36 Million Euros was Stolen via Malware", *Versafe*, December 2012.

[11] Funk, C., Garnaeva, M.: Kaspersky security bulletin 2013. Overallstatistics for 2013. Securelist (2013)

[12] Anagnostopoulos, M., Kambourakis, G., Gritzalis, S.: New facetsof mobile botnet: architecture and evaluation. Int. J. Inf. Secur.1–19 (2015)

[13] Flo, A., Josang, A.: Consequences of botnets spreading to mobiledevices. In: Short-Paper Proceedings of the 14th Nordic Confer-ence on Secure IT Systems (NordSec 2009), pp. 37–43 (2009).

[14] Venkatesh, G.Kirubavathi & Anitha, R.. (2017). Structural analysis and detection of android botnets using machine learning techniques. International Journal of Information Security. 17. 10.1007/s10207-017-0363-3.

[15] S. Anwar, J. M. Zain, Z. Inayat, R. U. Haq, A. Karim and A. N. Jabir, "A static approach towards mobile botnet detection," 2016 3rd International Conference on

Electronic Design (ICED), Phuket, 2016, pp. 563-567, doi: 10.1109/ICED.2016.7804708.

[16] "Global Android malware volume 2018 | Statista", Statista, 2020. [Online]. Available: https://www.statista.com/statistics/680705/global-android-malware-volume/. [Accessed: 14- Aug- 2020].

[17] S. Esmaeili and H. R. Shahriari, "PodBot: A New Botnet Detection Method by Host and Network-Based Analysis," 2019 27th Iranian Conference on Electrical Engineering (ICEE), Yazd, Iran, 2019, pp. 1900-1904, doi: 10.1109/IranianCEE.2019.8786432.

[18] W. Hijawi, J. Alqatawna and H. Faris, "Toward a Detection Framework for Android Botnet," 2017 International Conference on New Trends in Computing Sciences (ICTCS), Amman, 2017, pp. 197-202, doi: 10.1109/ICTCS.2017.48.

[19] A. Karim, R. Salleh and S. A. A. Shah, "DeDroid: A Mobile Botnet Detection Approach Based on Static Analysis," 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), Beijing, 2015, pp. 1327-1332, doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.240.

[20] L. Taheri, A. F. A. Kadir and A. H. Lashkari, "Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls," 2019 International Carnahan Conference on Security Technology (ICCST), CHENNAI, India, 2019, pp. 1-8, doi: 10.1109/CCST.2019.8888430.

[21] "Investigation on Android Malware 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB", Unb.ca, 2020. [Online]. Available: https://www.unb.ca/cic/datasets/invesandmal2019.html. [Accessed: 14- Aug- 2020]

[22] B. Choi, S. Choi and K. Cho, "Detection of Mobile Botnet Using VPN," 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Taichung, 2013, pp. 142-148, doi: 10.1109/IMIS.2013.32.