National College of Ireland

DETECTION OF MALWARE USING MACHINE LEARNING IN ANDROID DEVICES/APPLICATIONS

MSc Internship
Cyber Security

Hanok Vijay Chukka
X19128622

School of Computing
National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. Hanok Vijay Chukka……

Student ID: ……x19128622……..……

Programme: ……Msc. CyberSecurity…… **Year:** ……2020……..

Module: ……Msc. Intership…….……

Supervisor: ……Imran Khan…….……

Submission Due Date: ……28th September 2020…….……

Project Title: Detection of Malware using Machine Learning in Android Devices/Applications

Word Count: ……4091…… **Page Count**……22……..……..

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: ……Hanok Vijay Chukka……

Date: ……..28th September 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):	

Detection of Malware using Machine Learning in Android Devices/Applications

Hanok Vijay Chukka
X19128622

Abstract:
Spreading malware through Android devices and applications became an important strategy of cyber attackers. Therefore, malware detection in Android applications has become an important area of research. In this context, it is important to answer the question that reads "how can we develop a model based on Machine Learning (ML) to detect malware in Android devices/applications?" When malware is detected in real time from Android mobile applications, it can relive the users of Android phones from the risk of malware. I will also help stakeholders of Android devices to be safe from malicious software. The proposed system extracts featurefrom. APK files and training is given for supervised learning. Different ML models like Multinomial Naïve Bayes, Random Forest and SVM are used as prediction models. With these ML techniques a framework is realized to have provision for protection of malware in Android devices or applications. The proposed solution continues giving support with increased quality. The rationale behind this is that as the applications are protected and malware is detected, the training data gets increased. With increased training data, it will become much more accurate as time goes on. With some changes, it can be made to detect Android applications live when it is associated with a competing device.

Keywords – Malware detection, feature extraction, machine learning, SVM, Random Forest, Multinomial Naïve Bayes

Contents

1 1. INTRODUCTION

Malware detection research has been around for many years. However, in the context of the latest technology known as Internet of Things (IoT), the threat of malware became more apparent. When there are number of IoT applications being used in the real word, each IoT application is made up of many devices and sensor networks. The devices include smart phones as well [11], [12], [14]. As the Android operating system (OS) is used by many smart phones, in this research Android mobiles and applications are considered for malware detection. Machine Learning (ML) based approach is followed for the detection of malware. As mobile devices are increasingly used in the modern applications, it is essential to know the reasons for malware spread and have a methodology for detection of malware.

In this project, **malware detection framework is proposed using ML techniques such as SVM, RF and DT. By using the concept of extraction of features from. APK file, the proposed methodology works based on the features learned**. The ML models are used to predict the class labels. Multinomial Naïve Bayes classifier, Random Forest Classifier and SVM classifier are effectively used as underlying models with feature selection in order to improve classification accuracy. The remainder of this report is structured as follows. Section 2 reviews literature on malware detection methods existing. Section 3 presents the proposed methodology. Section 4 covers the project plan in the form of Gantt chart.

2 2. LITERATURE REVIEW

This section reviews literature prior works on the malware detection in Android applications.

2.1 Mobile Botnet Detection

The use of mobile devices is increasing every year and with that there has been increased number of attacks or malware intrusions on Android mobile applications. Anwar et al. [1] proposed focused on mobile botnets. Mobile botnet is one of the

threats to mobile devices. Malicious applications associated with Android mobiles target mobile botnets in order to spread malware. A static method is proposed in order to detect botnets in Android applications. This method is made up of different techniques such as broadcast receivers, permissions and MD5. The features of Android applications are extracted and subjected to machine learning. Especially supervised learning approach is followed in order to classify the applications as malware affected or not. **They used a dataset known as UNB ISCX**. The dataset has around 14 kinds of malwares. The proposed method extracts feature from the mobile applications and then use the features for training a classifier. Then the trained classifier is further used to detect botnets association with applications.

Malatras et al. [5] investigated on different mobile botnets and the challenges thrown by them. A mobile botnet contains many components such as bot master, server, bots in the form of servants and clients and communication channels. The taxonomy includes mobile botnets, target, attacks and detection measures. Sensors are used as side channel in order to detect botnets and take corrective measures.

2.2 Deep Learning Based Methods

There is many deep learning based solutions existed on the malware detection in Android applications. Watcher and Yu [2] made an extensive review of literature on the deep learning based methods. The methods include supervised learning methods with **regression and classification, unsupervised learning methods like clustering, dimensionality reduction and density estimation and reinforcement learning methods like policy search, value function, TensorFlow, DeepLearning4J**, **Theano, Torch** and so on. There are different applications of deep learning including malware detection.

2.3 Dynamic Methods for Malware Detection

Dynamic methods are the methods that change at runtime based on the situation. It is on the contrary to static methods. Hasan et al. [3] proposed a methodology towards cyber security. It is known for working good against **StuxMob** which is a situational-aware malware that targets Android mobile applications. The StuxMob works differently when compared with the existing applications. It has its own threat model

and it makes use of payloads based on targets unlike traditional methods that make use of command and control based botnet. The StuxMob makes use of physical activities of mobile users and then plans attacks based on the runtime situations. The targeted attacks include **spying, hacktivist, ransomware, advertising, and attacks on health**. The StuxMob makes use of sensor data, performs action identification, maintain an action log, send trigger signal and classification is made finally from action log. Han et al. [4] proposed a big data and cloud based model for malware detection. It is one of the smart approaches that are available.

Eustance et al. [7] focused on security and privacy of mobile devices. They considered self-awareness, human factors and intervention approaches towards cyber security. Kor et al. [8] on the other hand discussed about security measures in presence of Internet of Things (IoT) integration with mobile applications. They proposed a layered architecture that plays crucial role in security. In a smart healthcare scenario, it was suggested to make use of security measures advised by the law known as HIPAA in distributed environments. Meng et al. [9] studied security in terms of "Bring Your Own Device" concept. As the organizations allow devices of users and their smart phones in the work place, they investigated security issues and measures pertaining to detecting malware. They discussed about Android container solutions such as **Samsung Knox, secure boot, trusted boot, trust zone, TIMA, sensitive data protection**, **authentication and access control and security with cryptographic techniques.**

In a distributed environment, device to device (D2D) communication is very important. However, such scenario makes use of mobile devices their security concerns may arise. Pedhadiya et al. [10] explored different aspects of D2D communication is studied. It includes dynamic situations in vehicular networks, mobile networks, and other networks where distributed scenarios are realized. The study also includes resource allocation related things and also security.

2.4 Edge Computing Based Security Methods

In case of mobile computing and Mobile Cloud Computing (MCC) there is provision for edge computing where local resources are made available to improve response time of the applications. At the same time, it needs security measures to protect

applications from cyber-attacks. Xiao et al. [6] made review of edge computing and the security measures being taken care of. They discussed different aspects of security. They include weak computation power, attack awareness, OS and protocol heterogeneity, coarse grained access control issues and other security problems. The edge computing architecture is studied along with security measures. They discussed about different attacks such as Distributed Denial of Service (DDoS) and side channel attacks. Malware injection attacks are also discussed in the paper. They used security measures like authentication, authorization, access control and prevention of cyber-attacks.

Edge computing is widely used in IoT as well. Ghorbani and Ahmadzadegan [11] explored security challenges associated with IoT devices that include mobile applications as well. The case studies they discussed include smart homes and other Mobile to Mobile (M2M) environments. The use cases where security is discussed include smart homes, transportation, retail, agriculture, factories and industries, wearable, smart cities and healthcare. Miloslavskaya and Tolstoy [12] also studied IoT environments for study of security. They focused on mobile devices through which malware is spread in such distributed applications. They discussed IoT architecture and the loop holes. There are specific attacks such as DDoS, replay and forgery. Security intelligence is included as part of the infrastructure in order to detect and prevent attacks. It has risk identification and also risk protection. The environment incudes different devices and networks besides applications that cater from simple modules to complex data streaming applications.

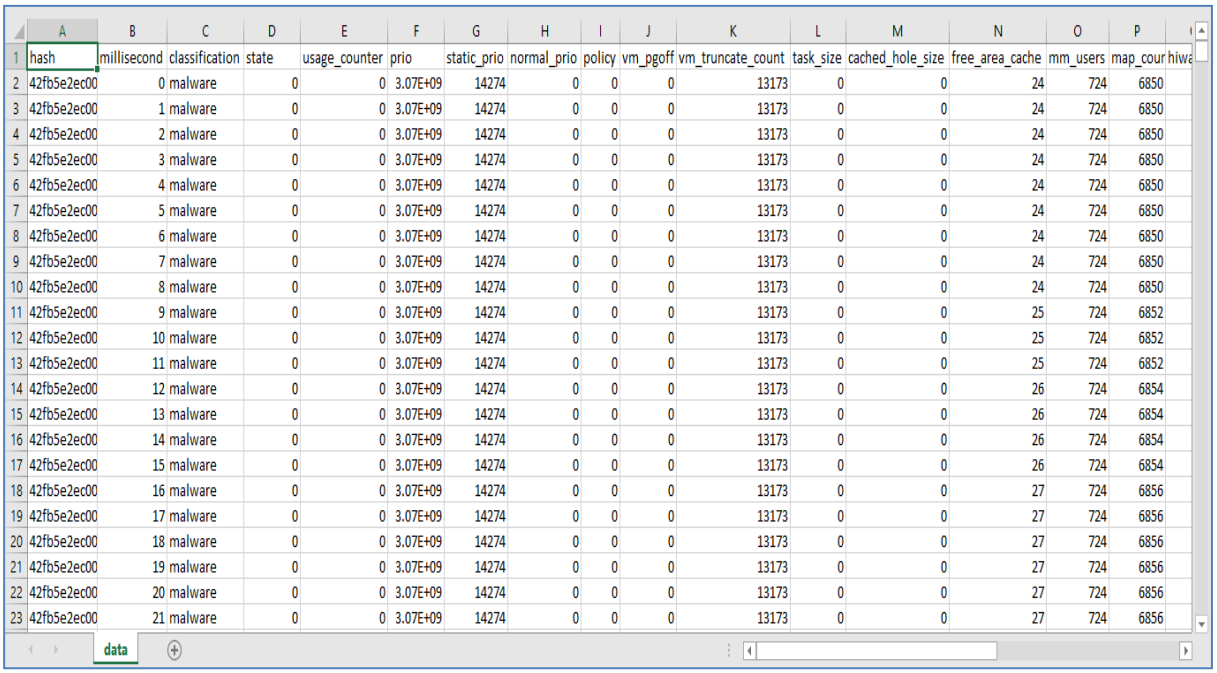
Security threats may arise due to different reasons. Li et al. [13] focused on general hacking approaches that are used by adversaries. It also provides different security issues and countermeasures. Different laws governed by state and international laws are provided. Lee et al. [14] proposed a concept known as Secure and Smart Network (S2Net) that is used to protect systems in IoT integrated use cases. It has rich security functionalities. The security features are installed in smart routers where the data is verified and attacks are detected. Stellios et al. [15] on the other hand focused on various attacks that are possible in IoT applications. In such applications Android mobiles are also used. The security provisions take care of application security. From the literature, it is understood that there is need for a malware detection framework based on machine learning for understanding features of. apk files and protect applications in Android devices.

3. METHODOLOGY

This section presents the proposed framework and methodology used to detect malware in Android applications. The following sub sections provide details related to dataset used, proposed framework and different ML techniques used for empirical study.

3.1 Dataset Used

Dataset used in the project contains thousands of Android applications and the details extracted from **.apk files**. There are plenty of features that are associated with each APK file.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	hash	millisecond	classification	state	usage_counter	prio	static_prio	normal_prio	policy	vm_pgoff	vm_truncate_count	task_size	cached_hole_size	free_area_cache	mm_users	map_cour
2	42fb5e2ec00	0	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850
3	42fb5e2ec00	1	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850
4	42fb5e2ec00	2	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850
5	42fb5e2ec00	3	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850
6	42fb5e2ec00	4	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850
7	42fb5e2ec00	5	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850
8	42fb5e2ec00	6	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850
9	42fb5e2ec00	7	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850
10	42fb5e2ec00	8	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	24	724	6850
11	42fb5e2ec00	9	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	25	724	6852
12	42fb5e2ec00	10	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	25	724	6852
13	42fb5e2ec00	11	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	25	724	6852
14	42fb5e2ec00	12	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	26	724	6854
15	42fb5e2ec00	13	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	26	724	6854
16	42fb5e2ec00	14	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	26	724	6854
17	42fb5e2ec00	15	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	26	724	6854
18	42fb5e2ec00	16	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856
19	42fb5e2ec00	17	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856
20	42fb5e2ec00	18	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856
21	42fb5e2ec00	19	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856
22	42fb5e2ec00	20	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856
23	42fb5e2ec00	21	malware	0	0	3.07E+09	14274	0	0	0	13173	0	0	27	724	6856

Figure 1: An excerpt of dataset used in experiments

As presented in Figure 1, the features extracted from the APK files of Android applications are presented. The details are an excerpt from the dataset collected. However, it provides some important attributes or features that are identified from the APK files.

3.2 Proposed Framework

This sub section provides the proposed framework as shown in Figure 2. It has all the mechanisms used in the proposed methodology in order to have an intelligent malware detection mechanism.

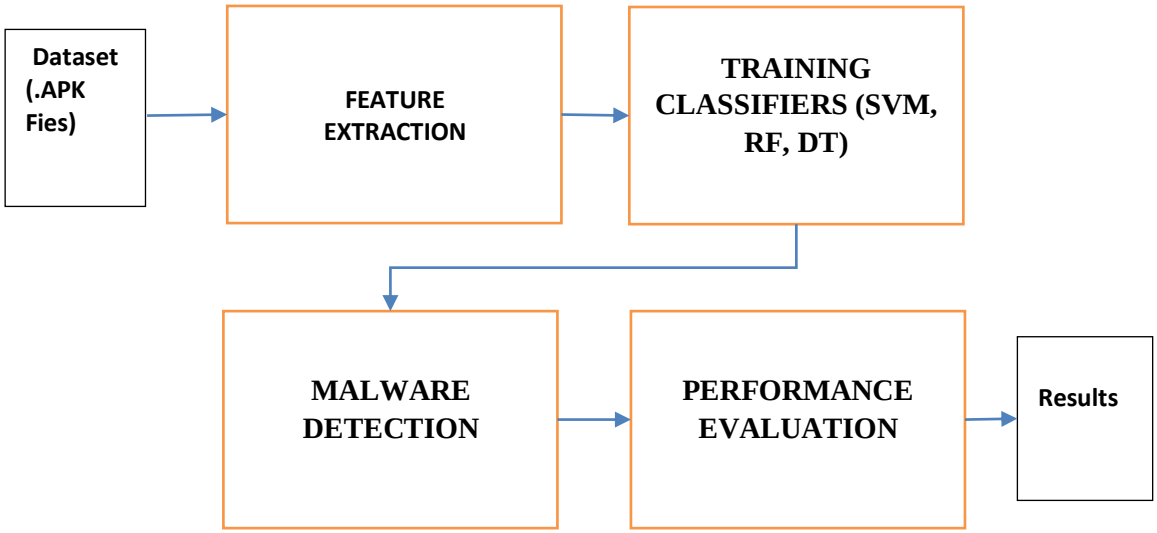


Figure 2: Proposed malware detection framework

As presented in Figure 2, it is evident that the proposed framework takes. APK files as input. In other words, the dataset contains the. APK files from which features are extracted. With the extracted dataset a training set is generated. The training set is used for learning classifiers like SVM, RF and DT. Once the classifiers are learned, it is possible to have detection of malware with new arrival of applications. The following sub sections provides the ML techniques used in the experiments.

3.3 Support Vector Machine

Support vector machine (SVM) is a widely used ML techniques. It analyzes the training set and makes an optimal hyperplane in order to have the capacity to classify the testing samples. The concept of maximum margin is used to clearly distinguish the positive samples from negative ones.

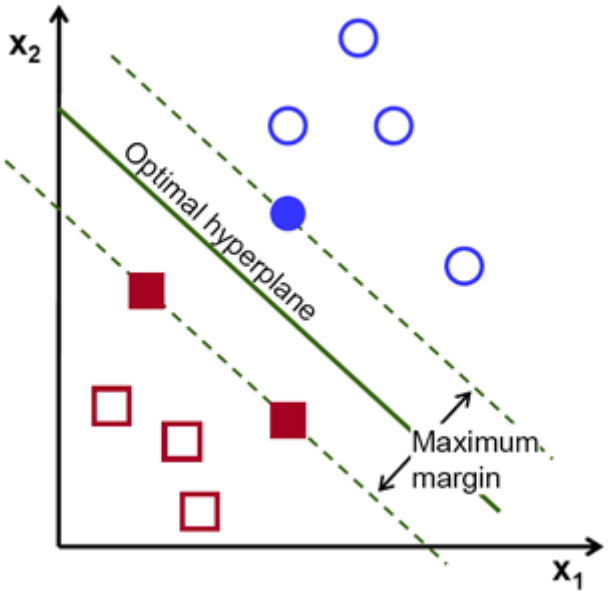


Figure 3: Shows mechanism used by SVM

As presented in Figure 3, the optimal hyperplane has power to distinguish classes. It is meant for binary classification by default. However, it can be used for multiple classes with kernels usage. In this project, it is used for binary classification of malware.

3.4 Multinomial Naïve Bayes

Multinomial Naïve Bayes is the classifier that is based on multinomial event model. It is the model were events are generated using multinomial. This event model is used for documentation purposes where histogram x is observed as in Eq. 1.

$$P(X\,|C_K) = \frac{(\Sigma_i x_i)!}{\Pi_i x_i!} \Pi_i pki^{x^i} \qquad (1)$$

3.5 Random Forest

Random Forest is used to have multiple decision trees and get ensemble of them with majority voting in order to select the final class label.

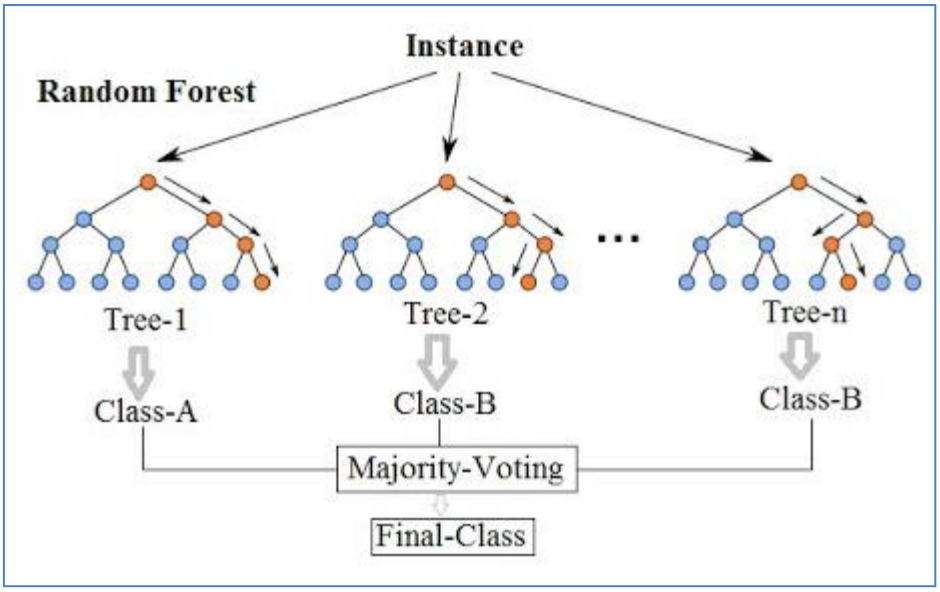


Figure 5: Random forest illustration

As shown by Figure 5, the RF makes use of many trees from the given instance. The results are ensemble and the majority voting is used in order to finalize the result.

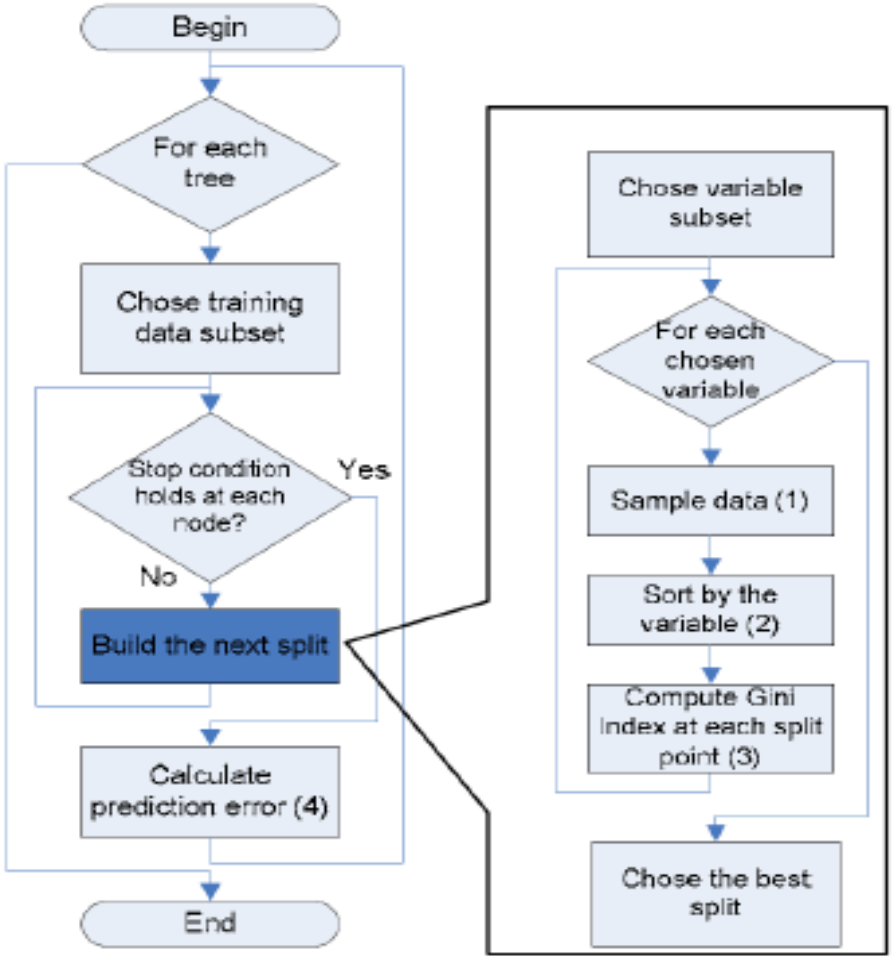


Figure 6: More details of RF classifier

As presented in Figure 6, the RF chooses training subset. Afterwards, the stop condition is verified. Based on the stop condition, decisions are made to continue operations. With ensemble method, it can handle large volumes of data. It can reuse trees and use them for supervised learning. It is also best used for outlier detection.

3.6 Tools and Technologies

This Android malware detection project is implemenned using Python data science platform. It needs the following software environment for execution.

1. **Python 3.6**
2. **Spyder (Python IDE)**

Python is general purpose high level programming language that is used to develop many kinds of real world applications. It have support for machine learning libraries. It has many IDEs such as Spyder for rapid application development.

4. IMPLEMENTATION

Implementation of the project is done using Python programming language. Drebin Android malware dataset is collected from https://www.sec.cs.tu-bs.de/~danarp/drebin/ for experimets. It has around 179 malware families and 5560 applications. The code is implemented with modularity. It resulted in multiple python source codes.

Constants.py

It is used to declare different constants used in this project. The constants avoid errors in coding. Different features of Android applications are provided as constants in this file. For example, S5 is the constant which means “restricted API calls”. Table 1 shows the details.

Constant	**Description**
S1	**Hardware components**
S2	**Permissions that have been requested**
S3	**Application components**
S4	**Filtered intents of Android application (s)**

S5	**API calls that have been restricted**
S6	**Permissions used**
S7	**API calls that are suspicious**
S8	**Usage of network addresses**

Table 1: Android features considered for malware detection research

Data_Cleaner.py

This source file contains the Python code that is used to obtain data and create required data frame for further analysis.

Data_Plotter.py

This source file contains python code to have different kinds of plots that reflect the Android malware detection analysis.

Data_Reader.py

This source file contains code that is used to read data from datasets and obtain the data to a data frame which is used for further analysis.

Extract.py

It extracts different categories of features that are used in data analytics using machine learning techniques.

Model.py

This source file contains code related to different machine learning models and the performance metrics.

Utils.py

This source file contains code for implementation of a timer utility that is used to measure time taken for different machine learning models.

Main.py

This source file is the entry point into the project. It imports all other modules and is the main execution unit of the application. It helps in running different supervied machine learning techniques for Android maklware dection. It has provison to use command line arguments in order to run specific machine llearning model selectively.

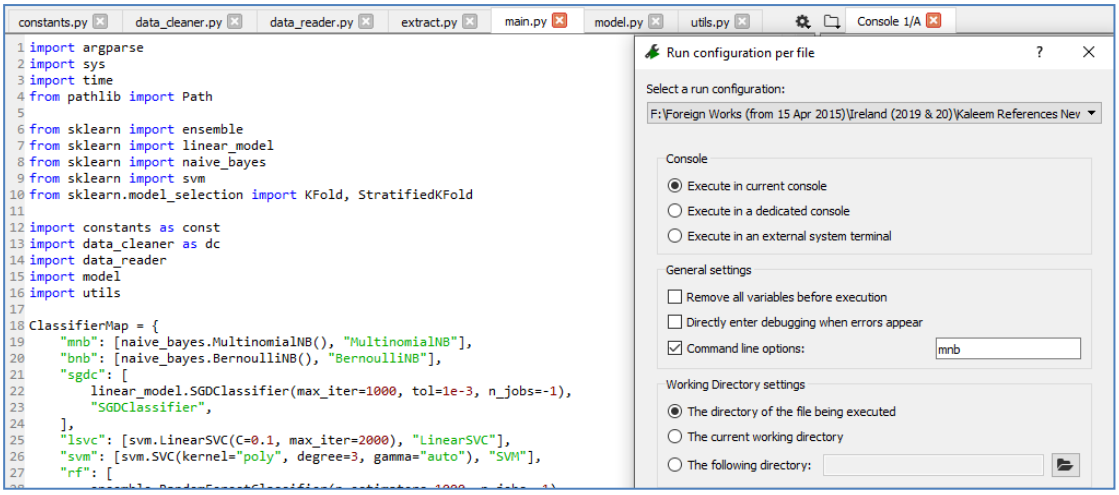


Figure 7: Execution of Android malware detection system with Multinomial Naïve Bayes model

As presented in Figure 7, the execution of the Android malware prediction model is made with command line argument “mnb”. Based on the argument, the ClassifierMap function in invoked and corresponding model is applied to the dataset in order to achieve Android malware detection.

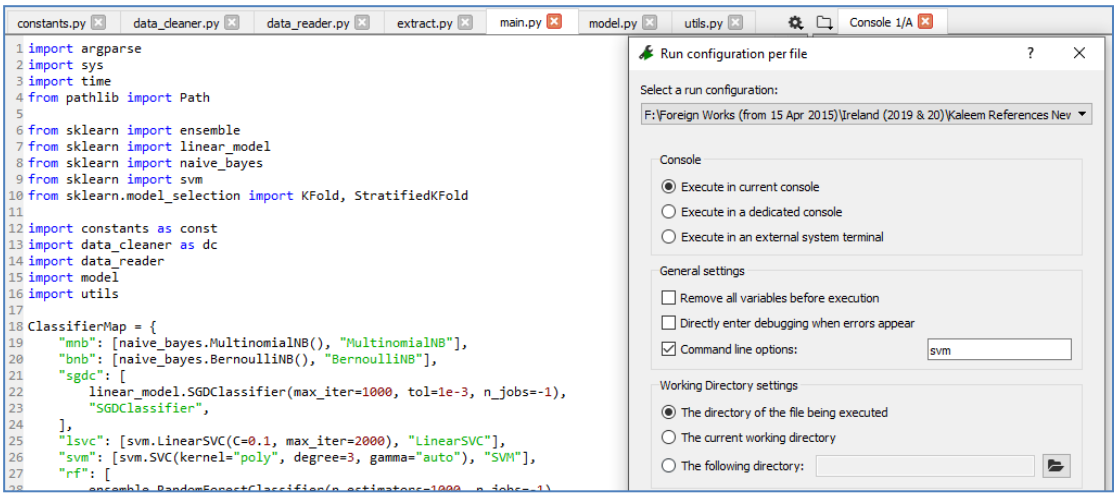


Figure 8: Execution of Android malware detection system with SVM model

As presented in Figure 8, the execution of the Android malware prediction model is made with command line argument “svm”. Based on the argument, the ClassifierMap function in invoked and corresponding model is applied to the dataset in order to achieve Android malware detection.

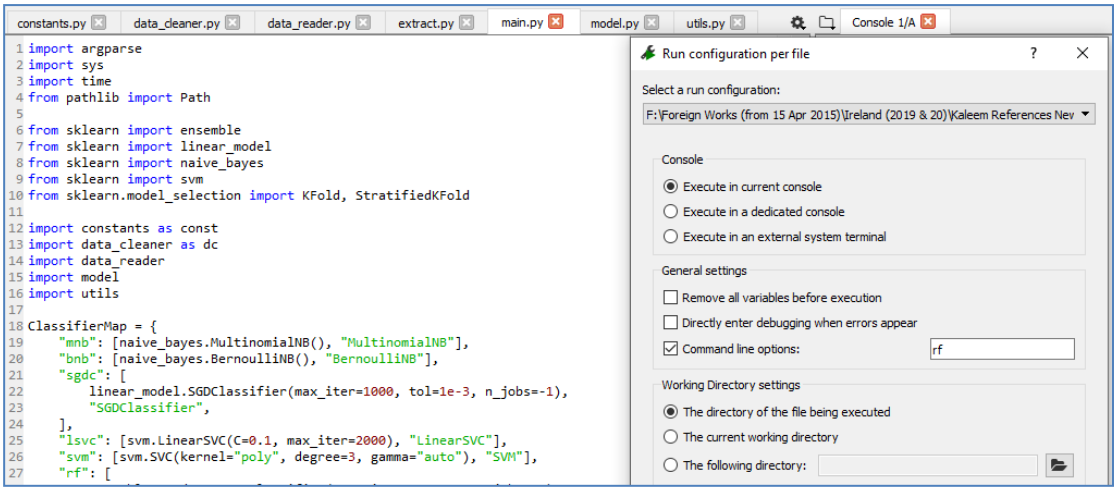


Figure 9: Execution of Android malware detection system with RF model

As presented in Figure 9, the execution of the Android malware prediction model is made with command line argument "rf". Based on the argument, the ClassifierMap function in invoked and corresponding model is applied to the dataset in order to achieve Android malware detection.

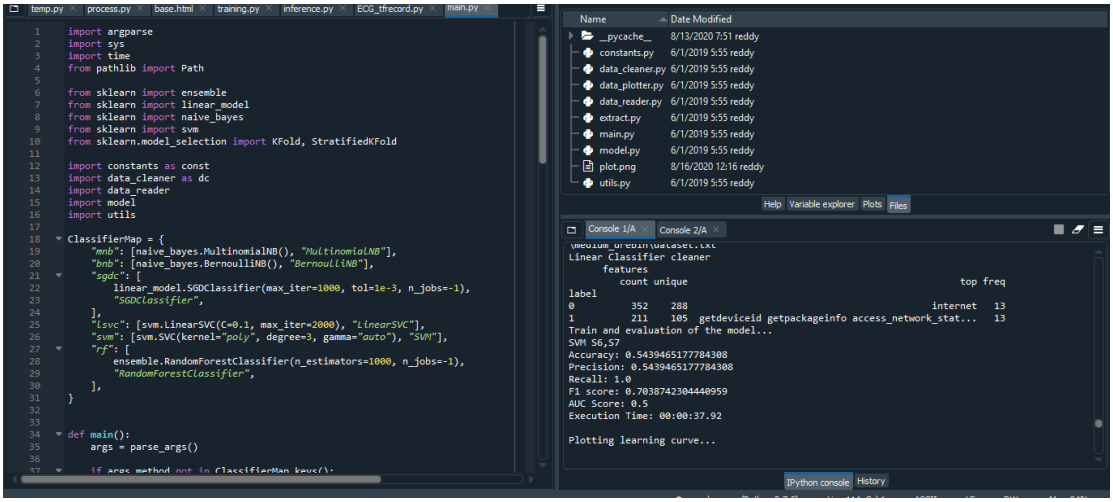


Figure 10: Result of SVM execution

As shown in Figure 10, SVM execution results are shown.

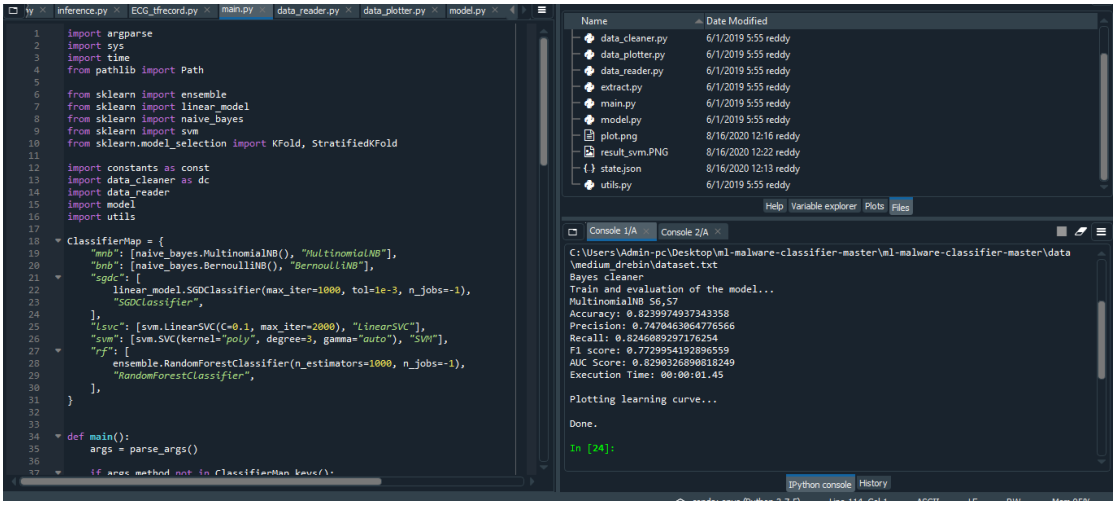


Figure 11: Results of Multinomial NB

As shown in Figure 11, Multinomial NB execution results are shown.

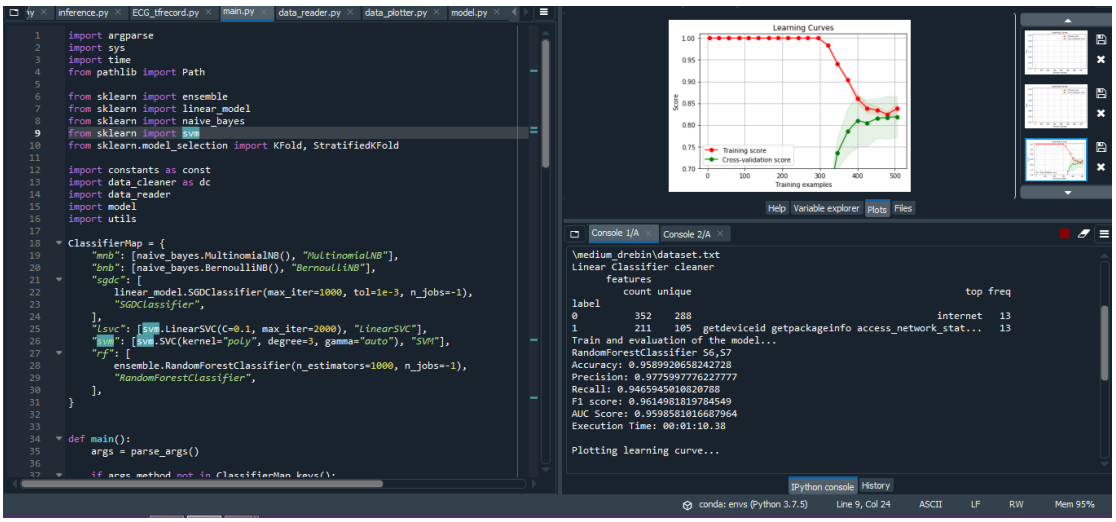


Figure 12: Results of RFAs shown in Figure 12, RF execution results are shown.

4.1 Performance Metrics

The Android malware detection models are evaluated using the metrics derived from confusion matrix provided in Table 2.

	Ground Truth (Yes)	Ground Truth (No)
Prediction Model (Yes)	True Positive (TP)	False Positive (FP)
Prediction Model (No)	False Negative (FN)	True Negative (TN)

Table 2: Confusion matrix used to derive different metrics

As presented in Table 2, the TP, FP, TN and FN are used to derive new metrics for performance evlaution. The metrics are provided s in Eq. 1, Eq. 2 and Eq. 3.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F - Measure = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \quad (3)$$

Precision is the measure related to exactness of the detection models while the recall indicates the completeness of the models. The mean of these two is the F-Measure that reflects overall performance of the prediction models.

5. RESULTS AND DISCUSSION

The results are obtained using the performance metrics like precision, recall and F-Measure. Execution time is also compared for all the prediction models used in the empirical study.

	Performance (%)			
Android Malware Detection Model	**Precision**	**Recall**	**F-Measure**	**Accuracy**
Multinomial NB	**0.74704**	**0.82460**	**0.77299**	**0.82399**
RF	**0.97759**	**0.94659**	**0.961498**	**0.95898**
SVM	**0.54394**	**0.54394**	**0.703874**	**0.543946**

Table 3: Comparison of Android malware prediction models

As presented in Table3, the prediction models are compared with metrics such as precision, recall and F-measure.

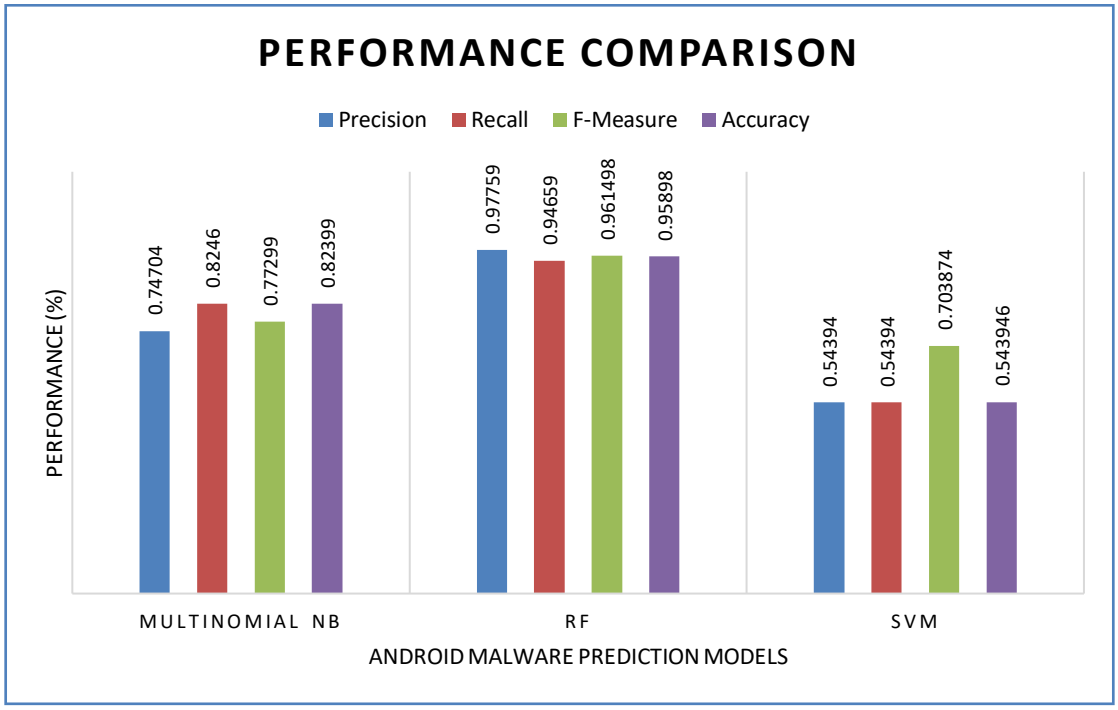


Figure 13: Performance comparison

As presented in Figure 13, the performance of the Android prediction models is compared. The prediction models and the performance (%) are shown in horizontal and vertical axes respectively. The models showed different performance levels. However, RF showed better performance over the other models in terms of accuracy.

Android Malware Prediction Model	**Time Taken (seconds)**
Multinomial NB	1.45
Random Forest	1.10
SVM	0.37

Table 2: Comparison of execution time (seconds)

As presented in Table 2, the execution time of different Android malware prediction model used in the empirical study are compared.

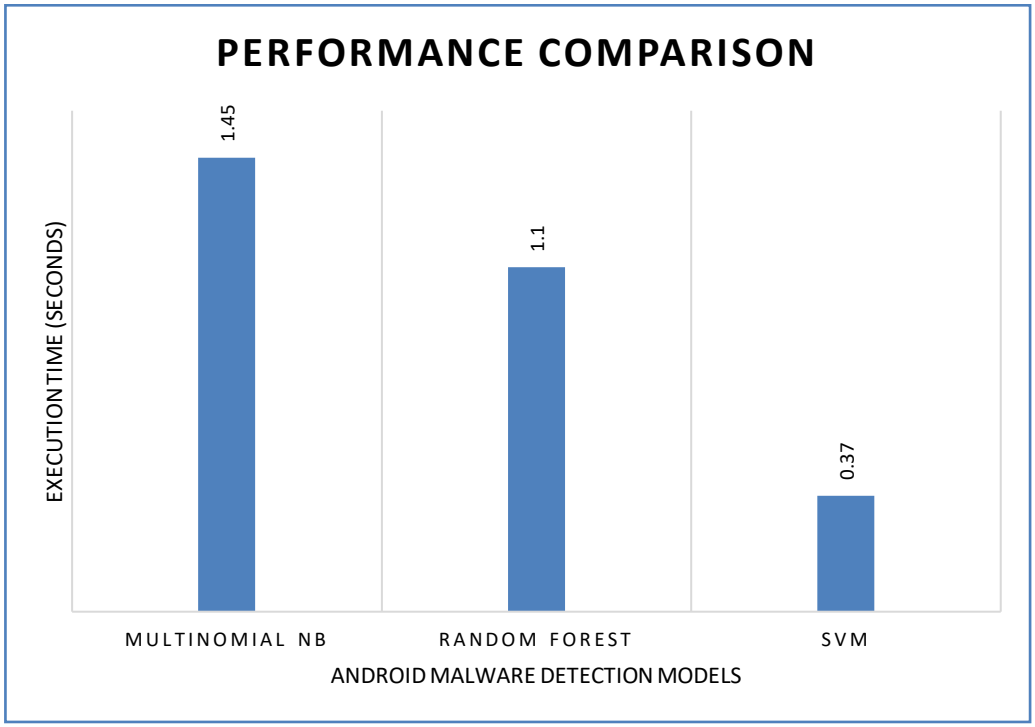


Figure 14: Execution time comparison

As presented in Figure 14, the time taken by each Android malware detection model is shown in vertical axis and the prediction models are provided in horizontal axis. The multinomial NB took 1.45 seconds for completion of Android malware detection. The time taken by the Random Forest model is 1.1 seconds while SVM took 0.37 seconds. Multinomial NB took more time for prediction. SVM showed better performance in terms of execution time.

6. CONCLUSION AND FUTURE WORK

In this project, **investigation is made in detection of Android malware. Different supervised learning methods such as Multinomial NB, Random Forest and SVM. Android malware dataset is obtained from. APK files. Different features of the Android applications such as suspicious calls and restricted API calls. Before application of classification models pre-processing is made. Features are extracted from the. APK files. The data is then used for training and prediction purposes. The implementation is made with modularity and the modules are reused in the main source file. The main() method takes command line arguments in order to evaluate prediction models selectively. The results are**

evaluated using metrics like precision, recall and F-measure besides execution time. The results revealed that RF showed 0.95898 accuracy which is better than multinomial NB and SVM that show 0.82399
and 0.543946respectively. With respect to execution time, the SVM model showed better performance in terms of execution time.

References

[1] Anwar, S., Zain, J. M., Inayat, Z., Haq, R. U., Karim, A., & Jabir, A. N. (2016). A static approach towards mobile botnet detection. 2016 3rd International Conference on Electronic Design (ICED) p1-5

[2] Hatcher, W. G., & Yu, W. (2018). A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. IEEE Access, 6, p24411–24432

[3] Hasan, R., Zawoad, S., & Haque, M. M. (2016). StuxMob: A situational-aware malware for targeted attack on smart mobile devices. MILCOM 2016 - 2016 IEEE Military Communications Conference p1-6

[4] Han, Q., Liang, S., & Zhang, H. (2015). Mobile cloud sensing, big data, and 5G networks make an intelligent and smart world. IEEE Network, 29(2),p 40–45

[5] Malatras, A., Freyssinet, E., & Beslay, L. (2015). Mobile Botnets Taxonomy and Challenges. 2015 European Intelligence and Security Informatics Conference p1-4

[6] Xiao, Y., Jia, Y., Liu, C., Cheng, X., Yu, J., & Lv, W. (2019). Edge Computing Security: State of the Art and Challenges. Proceedings of the IEEE, p1–24.

[7] Eustace, K., Islam, R., Tsang, P., & Fellows, G. (2018). Human Factors, Self-awareness and Intervention Approaches in Cyber Security When Using Mobile Devices and Social Networks. Security and Privacy in Communication Networks, p166–181

[8] Kor, A.-L., Yanovsky, M., Pattinson, C., & Kharchenko, V. (2016). SMART-ITEM: IoT-enabled smart living. 2016 Future Technologies Conference (FTC)p1-11

[9] Meng, T., Shang, Z., & Wolter, K. (2017). An empirical performance and security evaluation of android container solutions. 2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security).p1-8

[10]Pedhadiya, M. K., Jha, R. K., & Bhatt, H. G. (2018). Device to device communication: A survey. Journal of Network and Computer Applications.p1-26

[11] Ghorbani, H. R., & Ahmadzadegan, M. H. (2017). Security challenges in internet of things: survey. 2017 IEEE Conference on Wireless Sensors (ICWiSe).p1-6

[12] Miloslavskaya, N., & Tolstoy, A. (2017). Ensuring Information Security for Internet of Things. 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud).p 1-8

[13] Li, C.-Y., Huang, C.-C., Lai, F., Lee, S.-L., & Wu, J. (2018). A Comprehensive Overview of Government Hacking Worldwide. IEEE Access, 1–21.

[14] Lee, S., Shi, H., Tan, K., Liu, Y., Lee, S., & Cui, Y. (2019). S2Net: Preserving Privacy in Smart Home Routers. IEEE Transactions on Dependable and Secure Computing, p1–13

[15] Stellios, I., Kotzanikolaou, P., Psarakis, M., Alcaraz, C., & Lopez, J. (2018). A Survey of IoT-enabled Cyberattacks: Assessing Attack Paths to Critical Infrastructures and Services. IEEE Communications Surveys & Tutorials, 1–43