Permission based Android Malicious Application Detection using Machine Learning

Aditya Kapoor
Deptt. of CSE & IT
Jaypee University of Information Technology, Waknaghat, India
aditya.kapoor33@gmail.com

Himanshu Kushwaha
Deptt. of CSE & IT
Jaypee University of Information Technology, Waknaghat, India
him.kushwaha@gmail.com

Ekta Gandotra
Deptt. of CSE & IT
Jaypee University of Information Technology, Waknaghat, India
ekta.gandotra@gmail.com

***Abstract*—Since the launch of the smartphones, their usage is increasing exponentially. They have become an important part of our lives. We are very much dependent on smartphones for our daily routine and use numerous applications both from the play store or the third party applications. Most of the times, the applications downloaded from unofficial sources pose a threat as there doesn't exist the necessary checks or mechanisms to validate the authenticity of these applications and maybe infected with malware. The malware infected applications can lead to leakage of user's personal data. Anti-virus tools use signature based methods for detecting malwares, but their databases need to be updated regularly. In this paper, we present a system for classifying Android applications on the basis of permissions used by those applications. We used six machine learning algorithms for classifying these applications into malicious or benign applications. On comparing the results, we find that Logistic Regression Algorithm suits best to our dataset and provide 99.34% accuracy.**

Keywords—*Machine Learning; Android; Mobile Security; Malware Detection; Static Analysis; Permissions;*

I. INTRODUCTION

In today's era, smartphones have become a ubiquitous device for storage, computing and for carrying out the transactions among these devices. It is handier, portable and easy to use device. To make the usage experience of the mobile phones better and for determining the features and functions available on the device, a software platform i.e. an operating system (OS) is preinstalled in the mobile phones which is decided by the manufacturer. This OS is specially designed for mobile phones and largely vary from that of computers and therefore able to run advanced functions on smartphones that were previously unable to be done on desktop computers. There are various kinds of OS available in the market namely Android OS, IOS, Windows OS etc. The first fully functional and popular smartphone OS was the Symbian OS which was introduced in 2000. Another OS that revolutionized the market was the IOS by Apple and came along with their first iphone model in 2007. But since the launch of Android OS in 2007, it has become the most popular mobile OS and has grown strongly through the years. According to the **statista** report, the statistics given shows global market share held by the leading smartphone OS, in the second quarter of 2018 is 88 percent of all smartphones sold to end users were phones with the Android OS [1]. Thus, we targeted Android OS for the research. The availability of smartphones with Android OS at relatively cheaper rates have also led to this accelerated migration of regular phone users to smartphone users and thus there is an exponential growth in Android market.

This accelerated growth of the Android OS has largely attracted the malware developers. A large number of applications containing malware are being developed every day. Smartphones are becoming a major target of malware attacks with Android OS being the top on the hit list as it is open source OS and it is relatively easy to penetrate malware in these applications [2]. Thus, the detection and removal of these malicious applications is becoming a major concern for both developers and to the end users. It is becoming the need of the hour to keep the platform safe for the community by providing detection and defensive methods against malware.

Rest of the paper is organized as follows: Section II comprises of the related work and the background followed by the methodology used in Section III. Next, in section IV, the experimental results are analysed on the basis of various performance parameters. In section V, the paper is concluded and the future scope is outlined.

II. BACKGROUND AND RELATED WORK

There have been various methods for malware detection. The most popular method is signature based method. The signature of malicious samples is stored in a database and this database is then used to detect malware. This method is effective only to detect known malwares. ML algorithms have been introduced in order to detect zero-day malware [4],[2]. Due to the rise in the malware activities in the Android community, there have been many research efforts being put by researches toward the detection of malicious Android samples. Researchers from around the world follow different approaches to mitigate this issue. There have been many static as well as dynamic advancements in this field. Static malware analysis involves examining the code of the malware sample without executing it. On the other hand, dynamic malware analysis is to monitor the behaviour of the malware while it is being executed in the sandboxed environment [3]. Many static analysis approaches given in [5], [6], [7] follows basis on already known malware and compute applications through reverse engineering which help to decompile the packaged applications thereby making it easier to look for signatures or other heuristics written in the program code. Some follows different approaches like [8], [9], [10], which checks the usage of power by each application and reports the user or developers about any anomalous consumption.

Many dynamic analysis approaches used in [11], [12], [13] keeps a check on the pattern of system calls. Others like [15], [16] have implemented the approach of universal signature based methods that are able to compare the application in question with many known malware or other heuristics.

978-1-5386-9436-7/19/$31.00 ©2019 IEEE

Recently, the researchers have started to make use of data mining and machine learning (ML) for classifying and detecting malware [17], [18], [19], [20]. These methods are able to detect zero-day malware.

In [12], a framework based on ML known as Crowdroid is proposed that is able to recognize malware on Android smartphones on the basis of system calls and their frequency. Similarly, in [21], an intrusion detection system based on machine learning is proposed. It keeps a check on the user's as well as smartphone's behaviour by observing certain parameters, spanning from sensor activities to CPU usage. This model uses 88 features which together describes the system behaviours along with rooting the system and the external use of Linux server. Further, these are pre-processed by feature selection mechanisms. A machine learning based clustering model is used in [22], this model analyses the static function calls from binaries to detect anomalies. The Symbian OS used this kind of technique. This framework uses a client, Remote Anomaly Detection System, which monitors and visualizes the component. In [23] Dini et al. presented a multi-level system called MADAM (Multi-Level Anomaly Detector for Android Malware) which is capable of extracting features at both the kernel and application levels using 13 features to describe system's behaviour and system calls. But this model was only targeted for rooted device. In [24] Portokalidis et al. presented a different approach based on VMM approach to detect malware in their design of Paranoid Android system where a full malware analysis can be done in the clouds using many replicas of mobile phone. A secure virtual environment is created for the mobile replicas to run and which makes it possible for approximately 105 replicas. Mirela et al [25] presented an approach based on neural network which were very efficient in detecting the fraud calls and imposter. The main drawback of this method is that it is a slow process and it classifies the samples into groups having same behaviour and thus will lead to lot of false positives. Furthermore, Jerry et al. [26] worked on the transmission of viruses through SMS messages or other interfaces like Bluetooth and infrared.

This paper presents a system for classifying Android applications on the basis of permissions used by them.

III. METHODOLOGY USED

This section discuss the methodology used to classify the Android applications into malicious and benign on the basis of permission analysis. Figure 1 shows the Workflow diagram of our methodology.

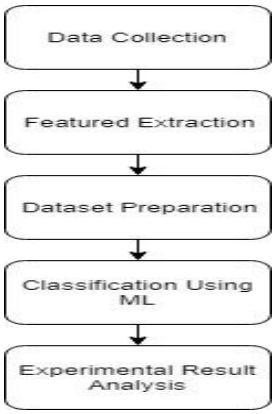

Fig. 1 Workflow Diagram for classifying Android Applications using Permission Analysis

A) Data Collection

We downloaded multiple Android application samples both malicious and benign. We have collected 2500 malicious Android applications from various websites like VirusShare, zeltser etc and 1500 benign application from Google's Android play store and other trusted website. These collected samples are validated and labelled using VirusTotal [27] (which consist of 60+ Anti-virus scanners).

B) Feature Extraction

Firstly, we studied about the malware and the permissions and the different categories of permissions that an application seeks during its functioning. These permission categories were available on google developer website [28] and give clear understanding of how the working of the application is affected by each permission the system requires both for software and hardware access.

We studied about the difference between the benign and malicious applications and the permission categories which were listed unsafe to be accessed by the user.

Android permissions are divided in to several protection levels:

1). **Normal Permissions** are those permissions which have very little risk of user's privacy. These permissions do not require user's involvement; these are granted by the Android system directly. eg. BLUETOOTH, INTERNET, SET_ALARM, WAKE_LOCK etc.

2). **Signature Permissions** are permission granted by the system at the install time. These are only granted to an app when it is signed by the same certificate as the app that defines the permission. eg. BIND_DEVICE_ADMIN, BIND_NFC_SERVICE, READ_VOICEMAIL etc.

3). **Special Permissions** -The permissions that doesn't comes under the normal and dangerous are the special permissions SYSTEM_ALERT_WINDOW and WRITE_SETTINGS are particularly sensitive, if an application wants to access these it must declare it in the manifest and access those with the help of intents.

4). **Dangerous permissions** -These are the permissions which require access to user's private data. For granting these permissions a message is prompt on the screen asking about the user's permissions. eg. READ_CALANDER, CAMERA, READ_CONTACTS, WRITE_CONTACTS, CALL_PHONE etc.

The permissions used in any Android application are found in the manifest.xml file of the zipped. apk package. We developed a Python script to extract these permissions and list them in a CSV file.

C) Dataset preparation

In this paper, we have used an approach which is aimed at uncovering the already known malware families and also the unknown malware to reduce chances of malware in the android community from escaping detection from scanners. For this we created a dataset using multiple Android .apk samples downloaded from both google play and VirusShare and other trusted sites providing malware samples. We got a collection of 4000 samples of both malicious and benign android application samples. As the permissions required by a particular application is inside the android manifest file of the android sample, we have created a script in python which reads and processes multiple samples at the same time and accesses the manifest.xml file and extract permissions and compile the permissions into a CSV file format which could be further used as an input file to the machine learning algorithms. This python script was run multiple times on the same samples to ensure the correct data and to lower the chances of randomization. The final result is used as a dataset in our paper and made free from error and stored.

With the help of this script, we parsed 4000 Android application samples both benign and malicious and extracted permissions which was stored in a CSV file format and used as a dataset. In the developed CSV, 202 permissions are used as the headers and 4000 applications are parsed in the rows and marked '1' if that permission is used in that application or '0' if that permission is not used in that application.

The dataset was further split in the ratio of 80:20 randomly. The 80% of the dataset was used to train the model and to identify the benign and malicious permissions and to create Labelled analysis. The remaining 20% of the dataset was utilized for the testing phase of the algorithm

D) Classification Using Machine Learning

After the creation of the dataset, a script is run to analyse the permissions and classify the samples into two different classes i.e. – malicious and benign. In this paper we focus on creating a labelled dataset for the machine learning algorithms and specifically for the supervised learning algorithms. The classifier is then trained using the previously created dataset and then tested to predict the result according to the feature vectors. No matter how complex or how advanced the machine algorithm is, any approach of the machine learning used can never be fully efficient to prevent the transmission of viruses.

Further, many predefined machine learning algorithms are applied to the input data which either classify the data into groups or identifies patterns among the dataset to [17] predict the output and give appropriate results. Machine learning algorithms are loosely classified into supervised and unsupervised learning algorithms. [29]

Now we will discuss about different ML algorithms:

- **Logistic Regression**- Logistic regression (LR) is an algorithm which uses the statistical concepts and models a relationship between the input and output numerical values. The model is represented by a logistic equation which combines the input values of a specific set and predicts the output for a set of that input values.

- **Linear Discriminant Analysis-**
  Linear Discriminant Analysis (LDA) is a very common technique for dimensionality reduction problems as a pre-processing step for machine learning and pattern classification applications. At the same time, it is usually used as a black box, but (sometimes) not well understood. The LDA technique is developed to transform the features into a lower dimensional space, which maximizes the ratio of the between-class variance to the within-class variance, thereby guaranteeing maximum class separability [27].

- **K-Nearest Neighbour (KNN)-** K-Nearest Neighbours (KNN) is a type of algorithm which can be used both for regression and classification problems but is mostly used in classification problems. This algorithm is easy in interpretation and requires very low calculation time and thus is a widely used ML algorithm. The K in this algorithm is the number of neighbours which are defined by the user. In this algorithm we use the Euclidean distance to measure the K nearest neighbours of the data point and predict the output according to its neighbours.

  Euclidean distance function: In Cartesian coordinates, if p = (p_1, p_2,..., p_n) and q = (q_1, q_2,..., q_n) are two points in Euclidean n-space, then the distance (d) from p to q, or from q to p.

$$D(p,q) = d(q,p) = \sqrt{\sum_{n=1}^{n} (q_i - p_i)^2}$$

- **Decision Tree**- Decision tree (DT) algorithm is a type of supervised learning algorithm in which a data structure is used to solve a problem. In this case the leaf node is referred to as the class label and the internal nodes of the tree represent the attributes. They are able to solve the problems of both classification and regression. Initially, we consider the whole dataset as the root and categorical feature values are preferred and the continuous values are first made discrete values before using them to build the model. Then statistical methods are used for ordering the attributes as internal node or root.

- **Gaussian Naïve Bayes**- Gaussian Naive Bayes (GNB) theorem is a type of classification algorithm which can be used for both binary and multi class classification problems. This theorem is called so because it has its roots of Bayes theorem. Naïve Bayes is often represented by probabilities. In this model the data is stored as probabilities for a learned model. Naïve bayes can be extended to real-valued attributes by assuming Gaussian distribution and by using mean and standard deviation from the training data.

$$P\left(\frac{h}{d}\right) = \frac{\left(P\left(\frac{d}{h}\right) * P(h)\right)}{P(d)}$$

Where

$P\left(\frac{h}{d}\right)$ is the probability of hypothesis h given the data d

$P\left(\frac{d}{h}\right)$ is the probability of data d given the hypothesis h was true

$P\left(\frac{d}{h}\right)$ *is the probability of the data d given the hypothesis h was true.*

$P(h)$ *is the probability of the hypothesis h being true.*

$P(d)$ *is the probability of the data*

- **Support Vector Classifier**- SVC is a supervised machine learning algorithm which is commonly used for both regression and classification problems. It is widely used in classification problems where each data item is plotted in n-dimensional space and n defines the features present and the value of each feature is the value of each coordinate. Further, a separate hyper plane is made to differentiate the two classes.

E) *Experimental Result Analysis*
Experimental results are discussed in the following section i.e section IV.

IV. EXPERIMENTAL RESULTS

This section discusses the experimental results obtained after applying the following ML algorithms on the created dataset as explained in previous section.

Table 1 Confusion Matrix

	Predicted Class		
Actual Class		Yes	No
	Yes	True Positive	False Negative
	No	False Positive	True Negative

To compare the classification models, following performance parameters are used. These parameters are obtained through the confusion matrix as shown in table 1.

- **True Positives (TP)** - These are the values which are correctly predicted and are positive values which can be described as the positive value of actual class and positive value of predicted class. It is denoted by TP.
- **True Negatives (TN)** - These are the values which are correctly predicted but negative values which refers to the negation of actual class and negation of predicted class. It is denoted by TN.
- **False Positives (FP)** – These are the values which are wrongly predicted but are true in real i.e. - when we have positive values of actual class but negation in predicted class.
- **False Negative (FN)** – These are the values which are wrongly predicted and negative in actual class.

Further, we look into more parameters of performance which are accuracy, precision, recall and F1 score.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad \dots (1)$$

$$Precision = \frac{TP}{TP + FP} \quad \dots (2)$$

$$Recall = \frac{TP}{TP + FN} \quad \dots (3)$$

$$F1\ Score = \frac{2 * (Recall * Precision)}{Recall + Precision} \quad \dots (4)$$

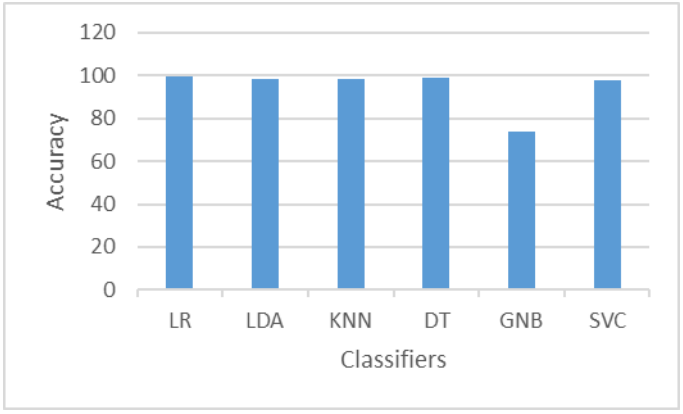

Fig. 2 Comparison of Accuracy for ML Algorithms used

Figure 2 shows the comparative analysis of the Accuracy which is calculated by using equation (1). In our experiment, Logistic Regression gives the best accuracy with the value 99.34%.

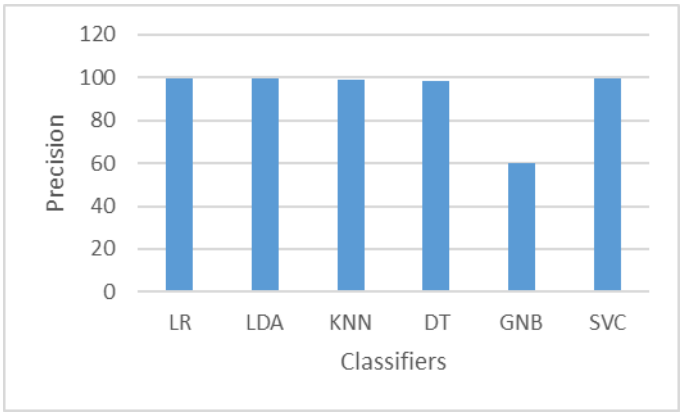

Fig. 3 Comparison of Precision for ML Algorithms used

Figure 3 shows the comparative analysis of Precision which is calculated by using equation (2). In our experiment, SVC shows the best precision with the value 99.6%.

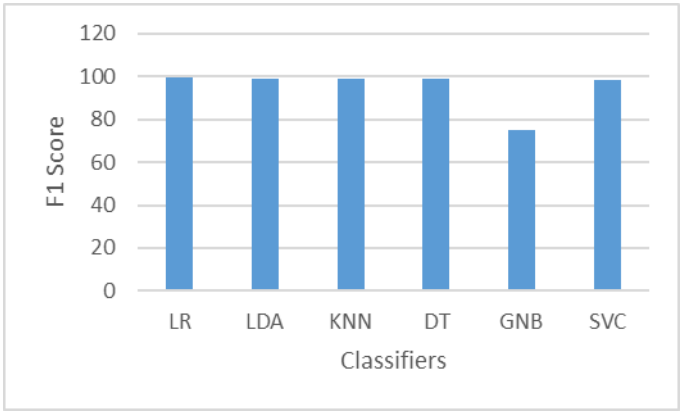

Fig. 4 Comparison of F1 Score for ML Algorithms used

Figure 4 shows the comparative analysis of the F1 score which is calculated by using equation (4). In our experiment Logistic Regression shows the best F1-Score with the value 99.5%.

According to the results obtained, we concluded that out of all the algorithms that were tested with our dataset, the algorithm Logistic Regression gives out the most accurate results with an accuracy measure of 99.34% and the precision and F1 score of 99.40% & 99.50% respectively.

Table 2 Experimental Results

Classifier	FPR	FNR	Accuracy (%)	Precision (%)	F1 Score (%)
LR	0.004	0.003	99.34	99.40	99.50
LDA	0.004	0.012	98.42	99.40	98.80
KNN	0.008	0.007	98.55	98.79	98.89
DT	0.012	0.005	98.82	98.59	99.09
GNB	0.261	0.003	73.68	60.08	74.87
SVC	0.003	0.018	97.89	99.60	98.41

V. CONCLUSIONS AND FUTURE SCOPE

In this paper, we have implemented various supervised machine learning algorithms for detection of malware in the Android application samples and classified them into two groups namely benign and malicious. For this classification, we use a labelled dataset which was created using permissions extracted from multiple Android applications from manifest file which was accessed programmatically using a python script. The dataset is split into training and testing data in the ratio 80:20. On this dataset six supervised ML Algorithms are used to classify the Android application in malicious and benign. From the results, it is concluded that Logistic Regression gives the best accuracy i.e. 99.34% with Precision value of 99.40%. After studying the concepts of various machine learning algorithms and after application of such machine learning algorithms on the dataset created by extracting the permissions from the Android manifest files and comparing the results of these algorithms on the basis of various parameters such as accuracy, precision etc. we concluded that there is a great potential in the machine learning algorithms in detection of malware. The dataset created by the python script is a self-created dataset and can be used in future research and used to implement other complex algorithms to get better outcomes or same algorithms can be implemented with a different approach to improve the time and space complexity of the models. Furthermore, more features and parameters can be included to enhance the analysis of performance measure. In our paper, we only considered the permissions for the classification of Android application samples. This work can be extended by considering other features of Android application like API (Application Procedure Interface) Calls, Network activities etc. with advanced ML Algorithms like deep learning algorithms.

REFERENCES

[1] Statistia, the statistics portal, [online] Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems\

[2] G. J. Tesauro, J. O. Kephart, G. B. Sorkin, "Neural networks for computer virus recognition". IEEE Expert, 11(4), pp.5-6. (1996)

[3] E. Gandotra , D. Bansal and S. Sofat, "Malware analysis and classification: 0A survey". Journal of Information Security, vol 5, pp.56-65 (2016).

[4] E. Gandotra, D. Bansal and S. Sofat, "Zero-day malware detection". In Embedded Computing and System Design (ISED), 2016 Sixth International Symposium on (pp. 171-175) IEEE December 2016.

[5] E. Gandotra, D. Bansal and S. Sofat, "Tools & techniques for malware analysis and classification". International Journal of Next-Generation Computing, vol 7 2016.

[6] D.Stevens,2007, [online] Available: http://blog.didierstevens.com/2007/10/23/a000n0000-0000o000l00d00-0i000e000-00t0r0000i0000c000k/.

[7] S. A. Derrick, "Detection of smart phone malware". Unpublished PhD. Thesis Electronic and Information Technology University Berlin. pp. 1-211.(2011)

[8] C. Mihai, S. Jha, "Static analysis of executables to detect malicious patterns". In Proceedings of the 12th conference on USENIX Security Symposium - Volume 12, SSYM'03, pp. 12, Berkeley, CA, USA, 2003.

[9] W. L. Raymond, K. N. Levitt, R. A. Olsson, "MCF: A Malicious Code Filter". Computers and Security, vol. 14, pp. 541 – 566, 1995.

[10] B. Dixon, Y. Jiang, A. Jaiantilal, S. Mishra, "Location based power analysis to detect malicious code in smartphones". In Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM '11, pp. 27–32., 2011.

[11] H. Kim, J. Smith, K. G. Shin, "Detecting energy greedy anomalies and mobile malware variants". In Proceedings of the 6th international conference on Mobile Systems, Applications, and Services, MobiSys '08, pp. 239–252, 2008.

[12] L. Liu, G. Yan, X. Zhang, S. Chen, "Virusmeter: preventing your cell phone from spies". In Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, RAID '09, pp. 244–264, 2009.

[13] F. Tchakounté & P. Dayang, "System calls analysis of malwares on android". International Journal of Science and Technology vol 2, pp. 669-674, 2013.

[14] I. Burquera, U. Zurutuza, S. N. Tehrani, "Crowdroid: behavior-based malware detection system for android". In Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 15-26, 2011.

[15] L. Xie, X. Zhang, J. P. Seifert, S. Zhu, "PBMDS: A behavior based malware detection system for cel phone devices". In Proceedings of the third ACM conference on Wireless network security, WiSec '10, pp. 37–48. International Journal of Network Security & Its Applications (IJNSA) Vol.7, November 2015.

[16] A. Bose, X. Hu, K. G. Shin, T. Park, "Behavioral detection of malware on mobile Handsets". In Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, MobiSys 2008, pp. 225–238, 2008.

[17] S. Y. Yerima, S. Sezer, G. McWilliams, "Analysis of bayesian classification based approaches for android malware detection". IET Information Security, Vol. 8, January 2014, pp. 25 – 36, DOI: 10.1049/iet ifs.2013.0095.

[18] Z. J. Kolter, A. Marcus, "Maloof: Learning to detect and classify malicious executables in the wild". J. Mach. Learn. Res., 7, pp. 2721–2744, December, 2006.

[19] M. G. Schultz, E. Eskin, E. Zadok, S. J. Stolfo, "Data mining methods for detection of new malicious executables". In Proceedings of the 2001 IEEE Symposium on Security and Privacy, SP'01, pp. 38, Washington, DC, USA. IEEE Computer Society, 2001.

[20] S. Asaf, K. Uri, E. Yuval, G. Chanan, W. Yael, "Andromaly: A behavioural malware detection framework for android devices". Journal of Intelligent Information Systems, pp. 1-30. doi: 10.1007/s10844-010-0148-x , 2011.

[21] S. A. Derrick, J. H. Clausen, S. A. Camtepe, S. Albayrak, "Detectiong symbian OS malware through static function calls analysis". In Proceedings of the 4th IEEE International Conference

on Malicious and unwanted Software (Malware 2009), pp. 1522, IEEE, 2009.

[22] S. Y. Yerima, S. Sezer, G. McWilliams, I. Muttik. “New android malware detection approach using bayesian classification”. 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), 2013.

[23] G. Dini, F. Martinelli, A. Saracino, A. Sgandurra, “MADAM: A multi-level anomaly detector for android malware”. computer network security, 7531,pp. 240-253. Retrieved from www.links.springer.com/book , 2013.

[24] G. Portokalidis, P. Homburg, K. Anagnostakis, H. Bos, “Paranoid Android: versatile protection for smartphones”. In Proceedings of the ACM 26th Annual Computer Security Applications Conference, ACSAC'10, ACSAC '10, New York, NY, USA, pp. 347-356, 2010.

[25] S. M. Mirela, B. Azzedine, N. Annoni, “Behaviour based intrusion detection in mobile phone systems”. Journal of Parallel and Distributed Computing, 62, pp. 1476-1490, 2002.

[26] C. Jerry, H. Y. W. Starsky, Y. Hao, L. Songwu, “SmartSiren: virus detection and alert for smartphones”. In Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys’ 07), ACM New York, NY, USA, pp. 258-271. doi: 10.1145/1247660.1247690, 2007.

[27] VirusTotal site [online]. Avalible: https://www.virustotal.com/#/home/upload

[28] Android developer permissions overview , [online] Available: https://developer.android.com/guide/topics/permissions/overview

[29] J. H. Friedman, R. Tibshirani, T. Hastie, “The Elements of Statistical Learning”. Book published-Februry 2009.