Android Anomaly Detection System Using Machine Learning Classification

Harry Kurniawan
School of Informatics and Electrical Engineering, Institut Teknologi Bandung
Jl. Ganeca no. 10, bandung 40132, Indonesia
harry_kurniawan@s.itb.ac.id

Yusep Rosmansyah
School of Informatics and Electrical Engineering, Institut Teknologi Bandung
Jl. Ganeca no. 10, bandung 40132, Indonesia
yusep@stei.itb.ac.id

Budiman Dabarsyah
School of Informatics and Electrical Engineering, Institut Teknologi Bandung
Jl. Ganeca no. 10, bandung 40132, Indonesia
budiman@stei.itb.ac.id

***Abstract*— Android is one of the most popular open-source smartphone operating system and its access control permission mechanisms cannot detect any malware behavior. In this paper, new software behavior-based anomaly detection system is proposed to detect anomaly caused by malware. It works by analyzing anomalies on power consumption, battery temperature and network traffic data using machine learning classification algorithm. The result shows that this method can detect anomaly with 85.6% accuracy.**

Keywords—android, anomaly, malware, power, internet, battery, temperature, machine learning.

I. INTRODUCTION

Android is the fastest growing and the biggest customer number for mobile device operating system. In 2013 it was reported that almost 550,000 Android's malware has 92% of all known mobile device malware and it increased 47% from 2012 to 2013 [19]. In general, there are two methods of malware detection, static and dynamic analysis [26]. Static method focuses on the source code of a program, which is containing malware function or not. On the other hand, dynamic method focuses on the behaviors.

Advance malware has functions which can evade static detection, for example Polymorphic, Metamorphic, and Obfuscation Code. Therefore, current malware detection system tends to use dynamic analysis [6]. On dynamic analysis, malware still can actively but act like they're inactive (dormant) and some malware have obfuscation technology [5]. Most of malwares when they are in active mode, they access hardware resources such as CPU, memory, Bluetooth and wireless devices. It will consume more energy and elevate battery temperature. It can be opportunities for detecting malware behaviors [20].

II. BACKGROUND

A. Malware

According to [22] Malware is software designed to attack, damage, disable, or disrupt computers, computer systems, or networks. In more specific, [23] describes Malware is code designed to changes the behavior(s) of computer operating system, computer kernel or applications, without user permission and cannot to be detected.

B. Malware on Android

Android has the most malware issue than another operating system like IOS and Windows Mobile. According to [18] there are 3.729.082 Android malware detected in 2013. This number is increasing every year, where in 2010 there were only 206 malware detected. 82% of Android applications and 100% of malwares try track user's network and location.

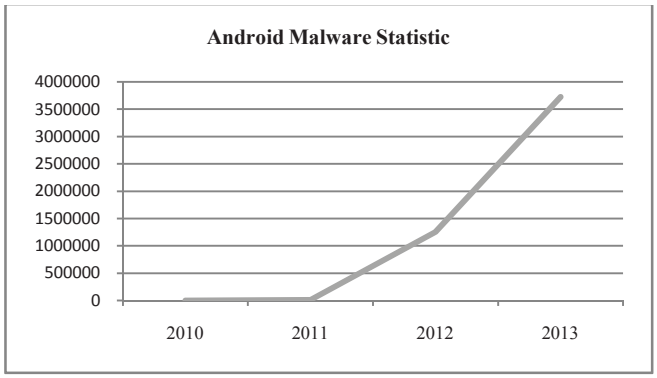


Figure 1. Android malware statistic
Source: McAfee security report on february 2014

Digging into malware trends, malicious software behavior is evolving from spyware and rooting exploits toward more malicious and invasive forms. The increasing behaviors include premium SMS, fraud, and downloader or installers that give the attacker remote access over the device or use exploits to get installed on vulnerable systems [30][33][16].

C. Machine Learning

Machine learning is a subject that studies how to use computers to simulate human learning activities. It is possible to find, describe, learn, obtain new knowledge and new skill, and identify existing knowledge by using experience, time, and mode from a dataset [17][9].

According to [32][3][27][17] the methods of Machine Learning are:

- Supervised Learning – the system works by mapping labeled inputs to desire output. The learner is required to learn function by considering input and output samples.

978-1-4673-7319-7/15/$31.00 ©2015 IEEE

- Unsupervised Learning – Given unlabeled input, the algorithm try to get hidden pattern on the dataset. There is no right or wrong on this learning type.
- Semi-supervised learning – the combination of labeled and unlabeled dataset to generate an appropriate function or classifier.

D. Evaluation Measurement

A classification system is expected to perform the classification of all dataset correctly. However, a system usually cannot work 100% correctly. Because of that, the classification performance should be measured [31].

Generally to measure the performance of a classification is by using confusion matrix. Confusion matrix is a table that records the results of the classification performance. Table 3 is an example of the confusion matrix binary classification for two classes, such as 0 and 1. Each cell in the matrix f_{ij} states amount of data which is classification results from class *i* for class *j*.

Based on the content of confusion matrix, it can be calculated amount of data for each class that are classified correctly ($f_{11} + f_{00}$) and the data are misclassified ($f_{10} + f_{01}$). By knowing the amount of data that classified correctly, it can be calculated the value of the accuracy of a classification. Accuracy has following formula [12].

Table 3. Matrix confusion for 2 classes

f_{ij}		*Class after classification*	
		Class = 1	***Class = 0***
Original class	***Class = 1***	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

$$Accuracy = \frac{\text{amount of data are classified correctly}}{\text{amount of data}}$$

$$Accuracy = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}} \quad (1)$$

Another quantity that can be used as a matrix evaluation of the performance of the classification is the True Positive Rate (TPR) and False Positive Rate (FPR). Both quantities can provide relevant performance value. TPR also called Sensitivity; measures the proportion of positives classes which are correctly identified. FPR refers to the probability of falsely rejecting the null hypothesis for a particular test. A good classifier has high TPR and low FPR [12].

- True Positive - correctly identified
- False Positive - incorrectly identified
- True Negative - correctly rejected
- False Negative - incorrectly rejected

		Class Classification Result	
		Positive	**Negative**
Original class	**Positive**	True Positive (TP)	False Negative (FN) Error Type II
	Negative	False Positive (FP) Error Type I	True Negative (TN)

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

$$FPR = \frac{FP}{FP + TN} \quad (3)$$

III. Related Work

Static Analysis – In [31] Yerima presents a malware detector with Bayesian classification model by building application profile obtained from application reverse engine, command detector and permission detector. Another research [23] provides an offline Android malware detection method by extracting permission, control flow diagram (CFG) and kernel analysis using Support Vector Machine algorithm. Sato [22] proposes malware detection system by analysing manifest file which contains permission, intent filter, install shortcut and process name. DroidAnalytic [34] is signature based analytic that can associate android malware by using three four signatures: application, classes, permission and method.

Dynamic Analysis – Ham and Choi [9] analyse android malware detection performance by analysing several resources: network, SMS, CPU, memory and Virtual Memory. Kato and Matsuura [15] analyse application permission, intent and Content Provider. The result shows it can keep good control for application and personal information. Wei et al. [29] propose android detection system which focuses on latent and spatial network analysis. The result shows 100% accuracy for network behaviour analysis using Naive Bayes algorithm.

Static and Dynamic Analysis – AASandbox is an Android sandbox proposed by Blasing et al. [4]. It has two analyses, static and dynamic. For static, it analyses uncompress APK, disassembles APK, and searches application pattern. In dynamic it installs, executes and gives input to APK. By analysing application, operating system scheduling, network, hardware and power Shabtai [23] proposes a Host-based Intrusion detection System (HIDS) and use Artificial Neural Network (ANNs) as the algorithm. Fedler et al. [8] proposes android antivirus API. It uses hash technology for signature matching and dynamic heuristic approach for malware detection.

IV. METHOD

This research works by collecting benign and malware samples and change to be dataset. The dataset will be the input for machine learning classification. The detail is on below.

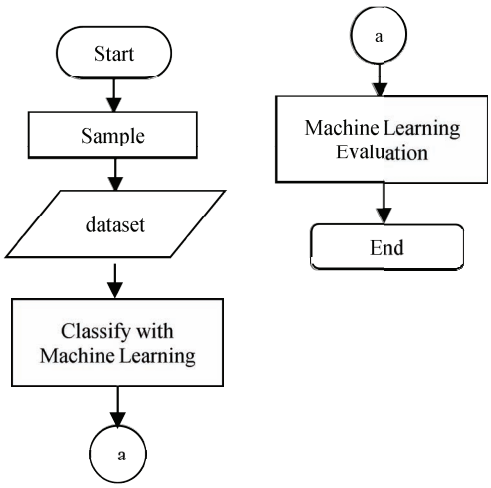


Figure 2. Research steps

A. Samples

Samples that used in this research are android applications. The samples are divided into two categories: malware and benign application. Malware sample applications are from Android Malware Genome Project [1]. The samples consist of 49 malware variants and 200 malware samples in total. Benign applications are downloaded from 200 Top Free Application on Google Play for Indonesia [28].

Table 1. List of malware samples

No.	Sample Name	No of Sample		No.	Sample Name	No of Sample
1	ADRD	6		16	DroidKungFu 3	7
2	AnserverBot	6		17	DroidKungFu 4	6
3	Asroot	6		18	DroidKungFu Sapp	3
4	BaseBridge	8		19	DroidKungFu Update	1
5	BeanBot	6		20	Endofday	1
6	Bgserv	6		21	FakeNetflix	1
7	CoinPirate	1		22	FakePlayer	6
8	CruseWin	2		23	GamblerSMS	1
9	DogWars	1		24	Geinimi	6
10	DroidCoupon	1		25	GGTracker	1
11	DroidDeluxe	1		26	GingerMaster	4
12	DroidDream	8		27	GoldDream	6
13	DroidDreamLight	6		28	Gone60	6
14	DroidKungFu1	6		29	GPSSMSSpy	6
15	DroidKungFu2	8		30	HippoSMS	4
31	Jifake	1		41	SMSReplicator	1
32	jSMSHider	6		42	SndApps	6
33	KMin	6		43	Spitmo	1
34	LoveTrap	1		44	Tapsnake	2
35	NickyBot	1		45	Walkinwat	1
36	NickySpy	3		46	YZHC	6
37	Pjapps	6		47	zHash	6
38	Plankton	6		48	Zitmo	1
39	RogueLemon	3		49	Zsone	6
40	RogueSPPush	6		TOTAL		213

B. Dataset

Dataset that used in this research are information from three features: sum of internet traffic, percentage of battery used, and battery temperature in every minute. To get all information it is needed a Logger Application which automatically records the three features. The process called logging. Logger application only works when the battery is not on charging.

Table 2. Dataset features and measurement

No.	Features	Measurement
1.	Internet traffic	Kilobyte
2.	Battery level	Percentage
3.	Battery temperature	Celsius

Logger application gets two inputs: application name and time duration of logging. The output is information about the three features in plain text format (.txt). This output will be used for machine learning for classification.

The dataset is divided into two categories: malware dataset and benign dataset. Benign dataset is dataset from installed benign application and malware dataset is dataset from malware application.

A factory reset phone is used to make sure there is no malware application inside. Then, it is needed to install sample application (malware or benign application) and also the logger application. Restart the phone and run the sample and logger application.

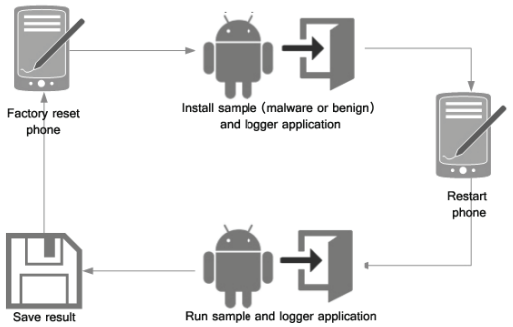


Figure 3. Step by step logging technique

To make sure the sample application works properly, we give input for all sample. The input can be different depends on the application types. Some application asks for login or registration to get service. For example, games application will be given input until the game can be played. Then run the logger application and let the process until finish. After logging is finish, we run factory reset on the phone and continue with the next sample application.

V. Result And Discussion

We use Weka as Machine Learning Library for testing and training with use 10-fold cross validation and 10 times testing for each test. Weka also provides the evaluation measurement and time duration in every testing. To get the best testing result we have 7 testing with different features combination for each. We compare 3 best machine learning algorithms result in whole testing for the comparison. Table 4 shows the feature combination that is used for each scenario.

Table 4. Features combination

Number of Feature combination	Testing Number	List of features		
		Network Data	Battery Consumption	Battery Temperature
1	1	√		
	2		√	
	3			√
2	4	√	√	
	5	√		√
	6		√	√
3	7	√	√	√

A. One Feature

In this test there is only one feature is used for each training and testing process on machine learning. The three features are: network data, battery consumption and battery temperature. The result for this testing is provided on Table 5.

Table 5. Evaluation measurement with one feature

Algorithm	Evaluation Measurement	Test 1	Test 2	Test 3
		Network Data	Battery Consumption	Battery Temperature
Random Forest	TPR	0.593	0.519	0.63
	FPR	0.422	0.587	0.368
	Accuracy	0.599	0.474	0.643
	Time Average	1.026	0.459	0.672
SVM	TPR	0.667	0.519	0.704
	FPR	0.456	0.587	0.346
	Accuracy	0.692	0.474	0.700
	Time Average	10.323	3.272	5.462
LMT	TPR	0.741	0.593	0.667
	FPR	0.320	0.593	0.400
	Accuracy	0.742	0.351	0.660
	Time Average	6.518	82.124	5.357
Highest Accuracy		0.742	0. 474	0. 700

As seen in figure 4, based on these three features we can summarize that the best features that can be used for detecting anomaly on this research is by analyzing network data. By using this feature we can get 74% accuracy with LMT as the algorithm. It also has highest TPR and lowest FPR, but not the lowest process time. The second best feature is by using battery temperature with 70% accuracy using Support Vector Machine (SVM) Algorithm. Using only battery consumption feature is not a good way to detect anomaly and it has only 47,7% accuracy .

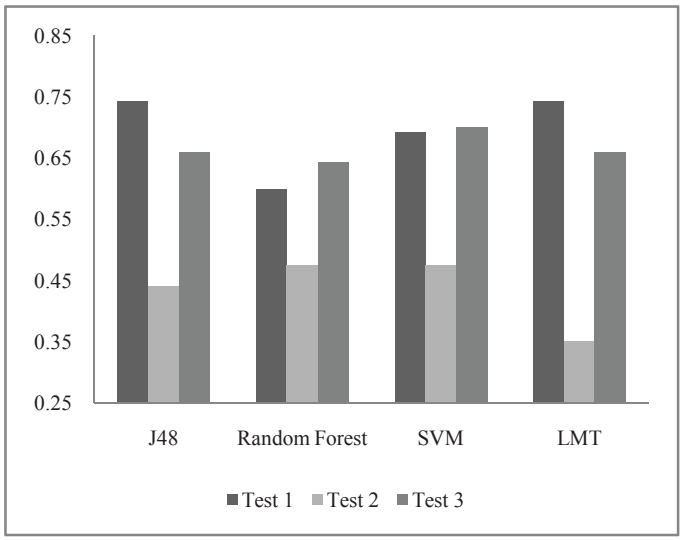


Figure 4. Chart for accuracy

B. Two Features Combination

This test uses combination of two features for each training and testing process on machine learning. The test has three combinations of each three features as written in Table 4. The result for this testing is provided on table 6.

Table 6. Evaluation measurement with two features

Algo-rithm	Evaluation Measurement	Test 4	Test 5	Test 6
		Network Data and Battery Consump-tion	Network Data and Battery Tempera-ture	Battery Consump-tion and Battery Tempera-ture
Random Forest	TPR	0.704	0.741	0.630
	FPR	0.346	0.263	0.397
	Accuracy	0.700	0.746	0.630
	Time Average	2.444	1.183	0.947
SVM	TPR	0.667	0.704	0.741
	FPR	0.428	0.403	0.292
	Accuracy	0.667	0.730	0.738

	Time Average	10.032	9.408	4.762
LMT	TPR	0.593	0.778	0.704
	FPR	0.422	0.238	0.317
	Accuracy	0.599	0.778	0.704
	Time Average	8.66	10.476	9.788
Highest Accuracy		0.700	0.778	0.738

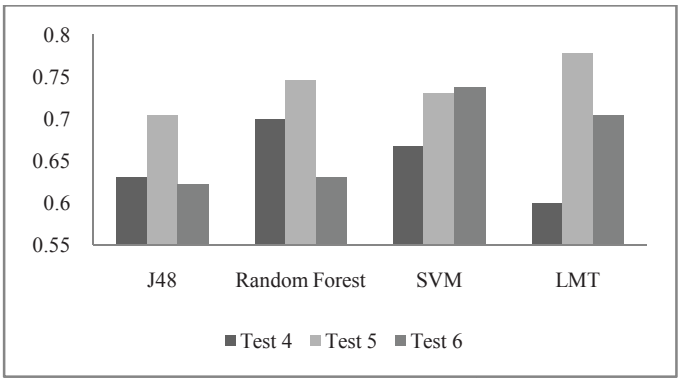


Figure 5. Accuracy chart with two features

Based on chart above we can see that the best two feature combination to detect anomaly is by using network data and battery temperature with Logistic Model Trees (LMT) as the algorithm. This combination has the highest score with 77.8% accuracy. It has the highest TPR, lowest FPR but the longest time average. The second best combination is by using battery consumption and battery temperature features and uses Support Vector Machine (SVM) as the algorithm. It has 73.8% for accuracy. The last is by using combination of network data and battery consumption with Random Forest as algorithm but it has only 70% accuracy.

C. Three Features Combination

Different as the last two tests before, in this test there is only one test can be created: network data, battery consumption and battery temperature. The result for this testing is provided on table 7.

Table 7. Evaluation measurement with three features

Algorithm	Evaluation Measurement	Test 7
		Network Data, Battery Consumption and Battery Temperature
Random Forest	TPR	0.856
	FPR	0.144
	Accuracy	0.856
	Time Average	0.901
SVM	TPR	0.793
	FPR	0.203
	Accuracy	0.842
	Time Average	6.047
LMT	TPR	0.853
	FPR	0.147
	Accuracy	0.853
	Time Average	20.135
Highest Accuracy		0.856

The best algorithm to work with theses three features combination is by using Random Forest algorithm with 85.6% accuracy. Having very small different score with Random Forest, the second best algorithm is LMT which has 85.3% accuracy . Support vector Machine (SVM) is on the third with 84.2% accuracy . The lowest score in this test is by using Logistic Model Tree (LMT) algorithm with 85.3% accuracy.

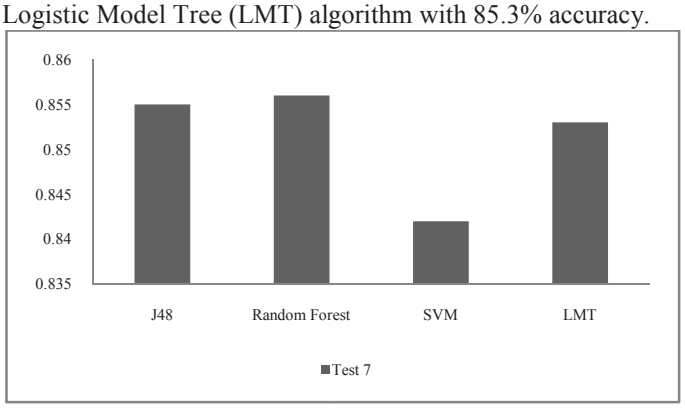


Figure 6. Accuracy chart with three features

VI. IMPLEMENTATION

This method is designed by using cloud computing technology. Installed application will collect information from client and sent to server. The server will analyze and classify whether the client is infected by malware or not and send the result to the client.

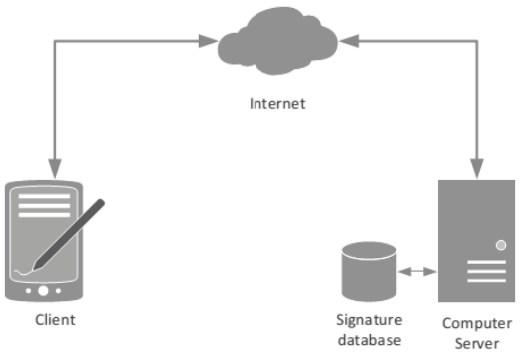
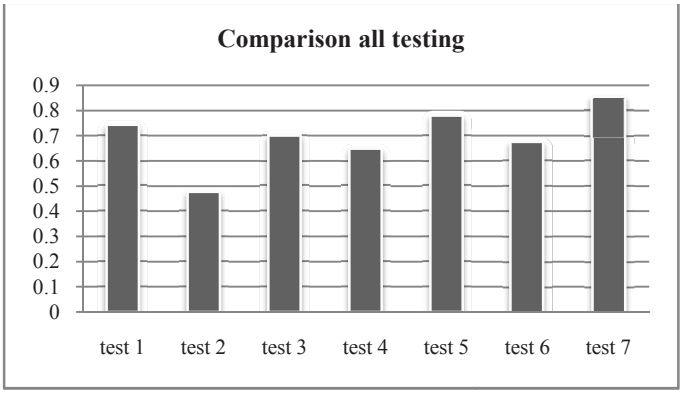


Figure 7. Design proposed implementation

VII. CONCLUSION

In this research we can conclude that power consumption, battery temperature and network traffic we can used as features to detect anomaly on android caused by malware using machine learning classification algorithm.



The best result to detect anomaly is by using combination of three features: network data, battery consumption and battery temperature with Support Vector Machine (SVM) as the algorithm and it has 85.6% accuracy.

ACKNOWLEDGMENT

This research was conducted in Cyber Security Center Institut Teknologi Bandung by funding from Markany Inc. and the malware samples are from Malware Gnome Project.

REFERENCES

[1] *Android Malware Gnome Project.* http://www.malgenomeproject.org

[2] *Android Open Source Project*. https://source.android.com/index.html

[3] Bell, Jason. (2015). *Machine Learning. Hands On For Developers And Technical Professionals*. Willey, US.

[4] Blasing, T., Batyuk, L., Schmidt, A.,D. (2010). *An Android Application Sandbox System for Suspicious Software Detection.* 5th International Conference on Malicious and Unwanted Software. France

[5] Comparetti, P.M., Salvaneschi, G., Kirda, E., Kolbitsch, C., Kruegel, C. Zanero, S. (2010). *Identifying Dormant Functionality in Malware Programs*. 2010 IEEE Symposium on Security and Privacy. California, USA.

[6] Dang, Bruce. At al. (2014) : *Practical Reserve Engineering: x86, x64,ARM, Windows kernel,reversing Tools, and Obfuscation.* Wiley. USA.

[7] Dini, G. (2012). *Madam: A Multi-Level Anomaly Detector For Android Malware.* Pisa: Universita Di Pisa. Italy.

[8] Fedler, R., Kulicke, M., Schutte, J. (2013). *An Antivirus API for Android Malware Recognition*. 8th International Conference on Malicious and Unwanted Software: "The Americas" (MALWARE). USA.

[9] Ham, H., S., Choi, M., I.. (2013). *Analysis of Android Malware Detection Performance using Machine* Learning Classifiers. International Conference on ICT Convergence. South Korea.

[10] Harley, David, et al. (2007) : *AVIEN Malware Defense Guide for the Enterprise*. Syngress Publishing, Burlington, USA.

[11] Hua, W., Lijuan, Z., Cuiqin, MA. (2009). *A Brief Review of Machine Learning and Application*. International Conference of Information Engineering and Computer Science.

[12] Japkowicz, Nathali. (2011). *Evaluating Learning Algorithm*. Cambridge University Press. United State.

[13] Ketari, L. (2012). *A Review of Malicious Code Detection Techniques for Mobile Devices*. International Journal of Computer Theory and Engineering , Vol. 4 No. 2.

[14] Kim, H. (2008). *Detecting Energy-Greedy Anomalies And Mobile Malware Variants*. USA: The University Of Michigan.

[15] Kato, M., Matsuura, S. (2013). *A Dynamic Countermeasure Method to Android Malware by User Approval*. IEEE 37th Annual Computer Software and Applications Conference. Japan.

[16] Maier, D., Muller, T., Protsenko, M. (2014). *Divide-and-Conquer: Why Android Malware cannot be stopped***.** IEEE 9th International Conference on Availability, Reliability and Security

[17] Mostafa, Y., A. (2012). *Learning From Data. A Sort Course*. http:amlbook.com

[18] Mcafee. (2014) *: Who's watching you* . McAfee Mobile Security Report.

[19] Networks, Juniper. (March 2012-March 2013). *Juniper Networks Third Annual Mobile Threats Report.* USA: Juniper Networks.

[20] Reavis, Jim. (2014) : *The Ongoing Malware Threat*. Symantec, White Paper.

[21] Rutkowska, Joanna. (2006) : *Introducing Stealth Malware Taxonomy*. COSEINC Advanced Malware Labs, Report.

[22] Sato, R., Chiba, D., Goto, S. (2013). *Detecting Android Malware by Analyzing Manifest Files*. Proceedings of the Asia-Pacific Advanced Network. South Korea.

[23] Shabtai, A., Elovici, Y. (2010). *Applying Behavioral Detection on Android-Based Devices.* Lecture Notes of the Springer Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Volume 48, 2010, pp 235-249

[24] Shas, J., Khan, Latifur. (2012). *A Machine Learning Approach to Android Malware Detection*. European Intelligence and Security Informatics Conference. Denmark.

[25] Security-Enhanced Linux in Android. https://source.android.com/devicestech/security/selinux/index.html

[26] Sikorski, Michael dan Honig, A. (2012) : *Practical Malware Analysis*. No Strach Press, California, USA.

[27] Sutton, S., R., Barto, A.,G. (2005). *Reinforcement Learning: An Introduction*. MIT Press. London, England.

[28] *Top Free in Android Free Apps.* https://play.google.com/store/apps/collection/topselling_free

[29] Wei, T., E., Mao, C., H., Jeng, A., B., Lee, H., M., Wang, H., T., Wu, D.,J. (2012). *Android Malware Detection via a Latent Network Behavior Analysis*. IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications. Liverpool, UK.

[30] Vokorokos, L., Hurtuk, Jan,. Mados, B. (2014). *Malware categorization and recognition problem*. IEEE 18th International Conference on Intelligent Engineering Systems • July 3-5, 2014, Hungary.

[31] Yerima, S.,Y., Sezer, S., Mutik, I. (2013). *A New Android Malware Detection Approach Using Bayesian Classification*. IEEE 27th International Conference on Advanced Information Networking and Applications.

[32] Zang, Yagang. (2010). *New Advance in Machine Learning*. Intechopen Publication. Croatia.

[33] Zhou, Y., Jiang, X. (2012). *Dissecting Android Malware: Characterization and Evolution*. IEEE Security and Privacy. San Fransisco.

[34] Zheng. M., Sun, M., Lui, J., C., S. (2013). *DroidAnalytics: A Signature Based Analytic System to Collect, Extract, Analyze and Associate Android Malware.* 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications.