Malware Detection for Android Operating Systems Applications

Aiman A. Abu Samra[1], Hasan N. Qunoo[2], Mahmoud Z.Alkurdi[3]

[1]*Islamic University - Gaza, aasamra@iugaza.edu.ps*

[2]*University of Palestine, - Palestine, hassanq@gmail.com*

[3]*Palestine Electricity Company, Palestine, Eng_mazk@hotmail.com*

Abstract: Many researches were done to find creative techniques, for Android platform, that can detect malware in easy and reliable manner. The aim is not only the effectiveness but to have less processing time, and less resources consumption. This research provide a solution for a part of this problem by finding an easy and fast way to analyze static application code and to generate its figure-print or signature to be used later in similarity measurement with available database of malwares signatures. We proposed a new method depends on SimHash algorithm which generate signature for reverse code from .apk android package kit. We compare the proposed algorithm with an existing Androguard tool, which also analyze static code and generate signatures using reverse engineering. We found that the proposed method saves 70% of time with similar results and time distribution behavior in comparison with Androguard.

1. Introduction

Smartphones are the latest technology trend of the 21st century. Today's social expectation of always staying connected and the need for an increase in productivity are the reasons for the increase in Smartphone usage [1] (Heloise Pieterse, 2012).

Popular platforms such as Android made the downloading and uploading process very convenient. So they have enabled the application marketplace to grow dramatically. The black market presence has also grown rapidly, where paid applications are modified for free download [2]Seung-Hyun Seo.

A Smartphone user may exposed to various threats. These threats can disrupt the operation of the Smartphone, and transmit or modify the user data. For these reasons, the applications deployed there must guarantee privacy and integrity of the information they handle.

There are several countermeasures and researches to detect and prevent malware in mobile devices some of these are signature-based antivirus scanners, which efficiently detect known malwares. Others depend on detection and classification method of the source code. These countermeasures and researches are different in its accuracy and in mobile resources consumption.

Android malware detecting using static analysis can provide a comprehensive view, it is still subjected to high cost in environment, So the question is how to detect unknown malware by reliable and useful method with low cost?.

2. Android Malwares and analysis tools

Google's model for accepting applications to be released in the Android Marketplace follows a very open policy. This means that malware can be distributed easily, compared to iOS applications where a rigorous vetting is conducted. Additionally, applications can be released anywhere on the web.

There are two types of code analysis that can be used to detect malwares, Static Code Analysis and Dynamic Code Analysis, The different between these types is that static program analysis is the analysis of software that is performed without actually executing programs while analysis performed on executing programs is known as dynamic analysis

3. Related Work

In [3] Aiman Abu Samra, authors explain how to apply clustering techniques in Malware detection of Android applications. They used machine-learning techniques in auto detection of malware

applications in the Android market. Their evaluation is given by clustering two categories of Android applications: business, and tools. They extract the features of the applications from XML-files which contain permissions requested by applications.

In [4] (Te-En Wei, 2012) , automatic malware detection mechanism for the Android platform based on the results from sandbox tool is proposed.

To identify possible information leakage, LeakMiner [5] (MoutazAlazab, 2012) applies a static taint analysis to apps within Android market. The approach introduces three steps in identifying possible leakages: first, apk files of Android apps are transformed to Java bytecode so that the following analysis can directly work on Java bytecode. Besides, application metadata are extracted from the manifest file of Android app.

A feature-based mechanism to provide a static analyst paradigm for detecting the Android malware proposed in [6] (ZheMin Yang, 2012). The mechanism considers the static information including permissions, deployment of components, Intent messages passing and API calls for characterizing the Android applications behavior. In order to recognize different intentions of Android malware, different kinds of clustering algorithms can be applied to enhance the malware modeling capability.

In [**Error! Reference source not found.**] (Md. Sharif Uddin, 2011) use SimHash to enhance detection of clones codes in large system which lead to unresolved bug or maintenance related problems by increasing the risk of update anomalies, they investigate the effectiveness of SimHash, a state of the art fingerprint based data similarity measurement technique for detecting both exact and near miss clones in large scale software systems they took an existing code cloning system and improved the time performance by an order of magnitude using SimHash, and demonstrated its feasibility for use with large systems such as the Linux Kernel. As well, they adapted SimHash to a code cloning framework and demonstrated its viability for the clone detection in large scale systems.

4. Methodology and proposed algorithm

In this research a new method of Android malware detection will be proposed, this model depends on extracting features from .apk file by generating a Hash code (SimHash) from its reverse code, and Manifest .xml file. This Hash will be used for similarity measurement to detects application behavior by comparing it with a dataset of malwares application. This method were compared with well know application used for the same purpose and called Androguard

Reverse Engineering

Reverse Engineering is a process of analyzing program code or software in order to test it from any vulnerability or any errors. Reverse engineering is the ability to generate the source code from an executable code. This technique is used to examine the functioning of a program or to evade security bugs, etc. Reverse engineering can therefore be stated as a method or process of modifying a program in order to make it behave in a manner that the reverse engineer desires. Example of reverse engineering tools is Androgaurd. Androguard is mainly a python tool done by VirusTotal project, VirusTotal is a subsidiary of Google, is a free online service that analyzes files and URLs enabling the identification of viruses, worms, trojans and other kinds of malicious content detected by antivirus engines and website scanners. [8] (VirusTotal).

Similarity Measurement

To compare two things, the human brain create a list of criteria for each item [10] (Galopin), SimHash will do the same, it will compare two texts using each word as a crtieria that describe the text. In maths, we can sum up it as:

$$sim(A,B) = \frac{A \cap B}{A \cup B}$$

SimHash is the process in which fingerprints will be created to compare the texts. Therefore, $(A \cap B)$ will be replaced by the comparison between their binary fingerprints, which is much more effective.

Malware detection by SimHash algorithm

In this research we propose a new method which measure similarity between two android applications to detect malware application by using output of Androguard as reverse engineering tool and SimHash algorithm.

SimHash algorithm has been used to find the distance between two strings or two files by converting them to hash codes of 32 or 64 bits. The idea of SimHashing two strings is to convert similar strings to similar hashes and then compare between two hashes. That makes the process of comparison has less time and less resources consumption.

In our proposed method we make a static code analysis so the input will be .apk file, which is the setup file of android application. The first step is to use Androguard "androlyze.py" and "get_package()" tool to extract and analyze .apk file. Androlyze program has many functions it could disassemble an Android application, it has "get_dex()" method which return a content of dex code, compiled code by Dalvik machine for Android application. The returned of this function is used as the string input for SimHash algorithm. Also the returned of the "get_package()", which returns the content of manifest.xml file, will be concatenated to the string input.

In step two we use SimHash input to compare between the generated .Dex , .XML from .apk files. The result of hamming distance of those two hashes will be the distance between applications.

Finally in step three we use step one and two but with an existing database of malware application SimHash signatures, if the results are very closed, which means it is higher than the threshold, then the input will be considered as malware. If not the application will be considered as legal so not worm or malware application.

Proposed algorithm

So proposed algorithm can be summarized as follows:

- *Input .apk file*
- *Use Androlyze. get_dex() and get_package() method to get content of .DEX compiled code and .XML manifest file*
- *SimHash .DEX and .XML for inputs*
- *Measure similarity with exists malware database signatures by using hamming distance*
- *If the output > threshold… input is malware.*

5. Experiment 1: Comparison with Androguard.

i. Experiment Environment:

Androguard library has been installed on Ubuntu Linux based operating system which has been setup as virtual machine run by oracle virtual box on Dell OptiPlex 9010 brand name pc (CPU: Intel core i5-3470 @3.2GHz 4 CPUs,RAM:8G, OS: Windows 7 64bit) ,the virtual machine setting was as follow:(Processor: 2CPU execution cap:100% and Ram:4G,OS:Ubunto4.3) , Also our new program has been written on the same virtual machine with python programming language and it has used some of libraries from Androguard tools as shown in appendix.

ii. Experiment Inputs and dataset

In this research training data has been collected from [11] Contagiodump (Mila, 2014) which provides smartphone malware that infects various smartphone platforms such as Android, iPhone, BlackBerry and Windows mobile, Contagio mobile mini-dump is a part of contagiodump.blogspot.com. Contagio mobile mini-dump offers an upload dropbox for you to share your mobile malware samples.

A complete dataset of different malware android applications (130 different malware) has been downloaded from Contagiodump, All experiment has been done on those applications as will be shown in next sections. Those applications are different in functions, types, source and type some are business applications, games, social and….etc., All application has been downloaded from http://www.mediafire.com/?78npy8h7h0g9y link.

iii. Comparison between Androguard and SimHash.

To compare between Androguard and SimHash as a tool for similarity measurement, we have to test two tools on different similarity processes, First; We have make pairing between all application from downloaded dataset, about 130 applications produce a number of 8257 measure similarity process measured by Androguard and SimHash as shown in table 1.

Table 1: Similarity measurement, Androguard and SimHash

Similarity Measurement Method	# of comparison	Total time(hours)	Average Time (sec)
SimHash	8257	11	5
Androguard	8257	39	13

As shown in table 1 there are a big difference in computation time for two methods our proposed algorithm is faster than Androguard which takes about 3 time more than SimHash tool. The reasons is the complexity of Androguard, which make a lot of complicated process to find signature of malware, it uses Elsim tool, this tool analyze the reversed code to find identical method.

After generating data from SimHash we found that all data are centralized between (75%-100%) so it is preferred to normalize data to be between (0%-100%) in order to compare it with Androguard. Figure 1 show the comparisons process of random applications. In general similarity distance are identical for values more the 70%, this indicates that proposed method has a very good result near to Androguard but in small time. As shown in figures 1,2,3 and 4 some processes takes about two minutes by Androguard and takes little than one minutes by SimHash with the same similarity distance in both ways. It is obviously that proposed method are very efficient for time and resources saving.

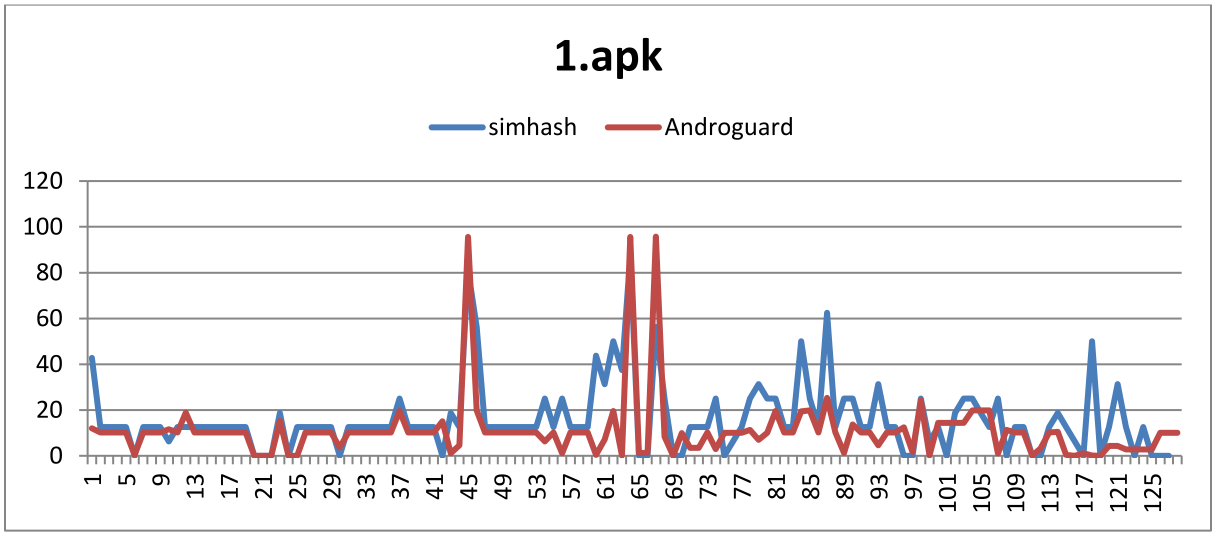

Figure 1: similarity measurements for 1.apk and data set by SimHash and Androguard

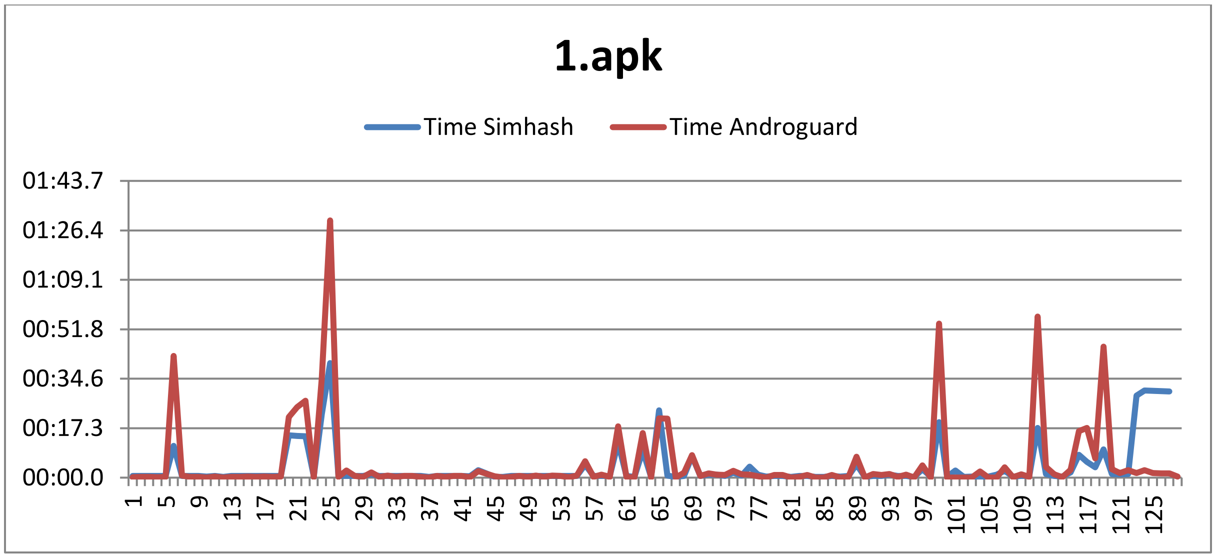

Figure 2: Time for similarity measurements for 1.apk and data set by SimHash and Androguard

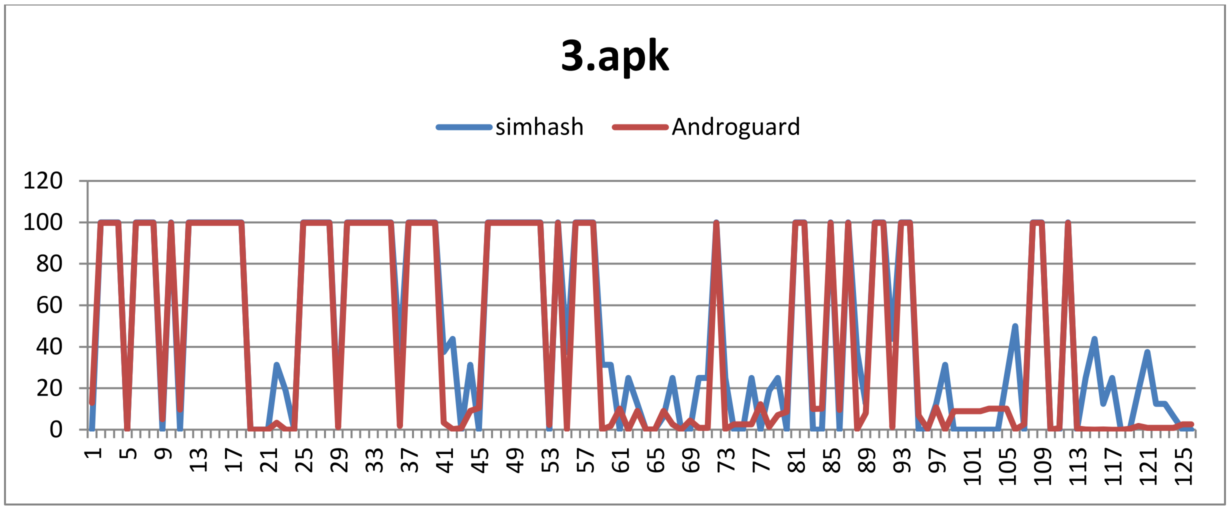

Figure 3: similarity measurements for 1.apk and data set by SimHash and Androguard

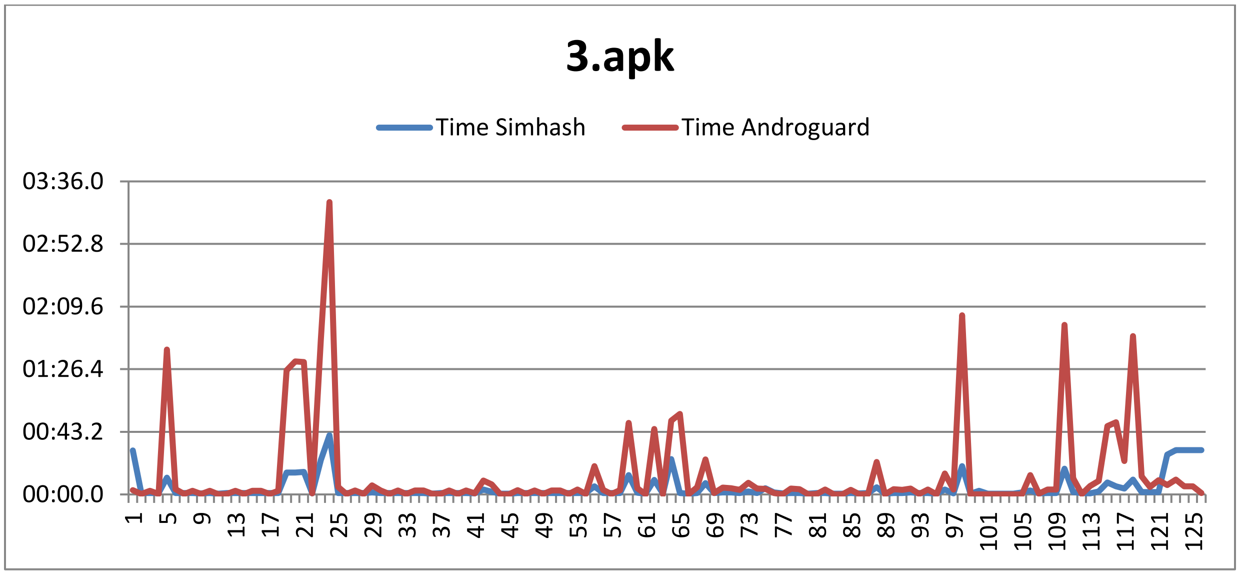

Figure 4: Time for similarity measurements for 1.apk and data set by SimHash and Androguard

To study the time behavior of proposed method and to ensure its feasibility, A time distribution model has been studied for the same experiment results, We have use SPSS PASW statistics Release 18 in order to generate histogram model for collected data, The histogram output as in figures 5 and 6 show that experiment takes similar time distribution behavior for both method Androguard and SimHash, they act as exponential distribution not as normal distribution this indicate that overall measurements process are concentrate at period between 1-10 sec , for both Androguard and SimHash, but it is noticed that SimHash has more distribution for that period between 1 and 30 sec that major process are accrued at this time while Androguard has more distribution at long time periods, this ensure the feasibility of our proposed method for time saving.

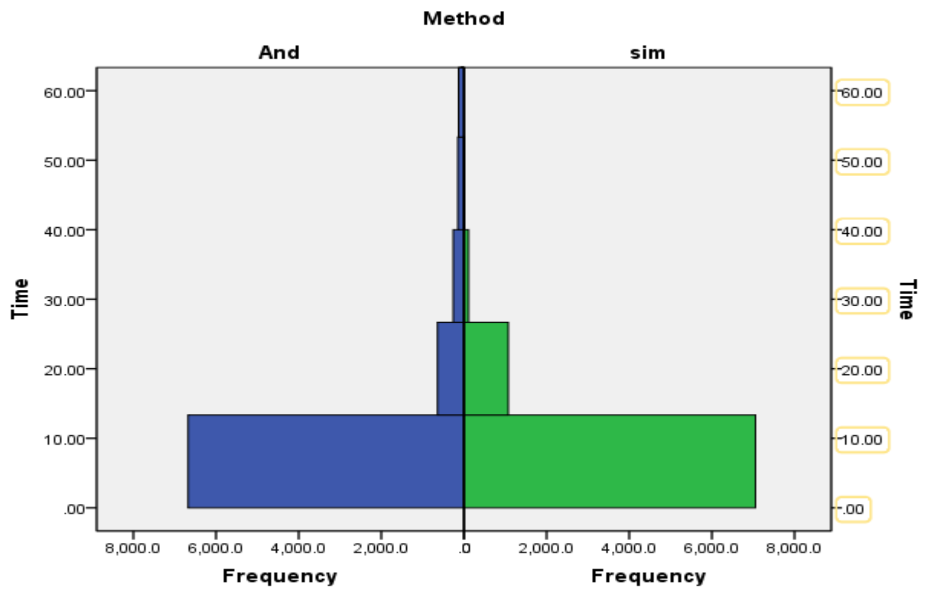

Figure 5: Histogram for SimHash and Androguard (1)

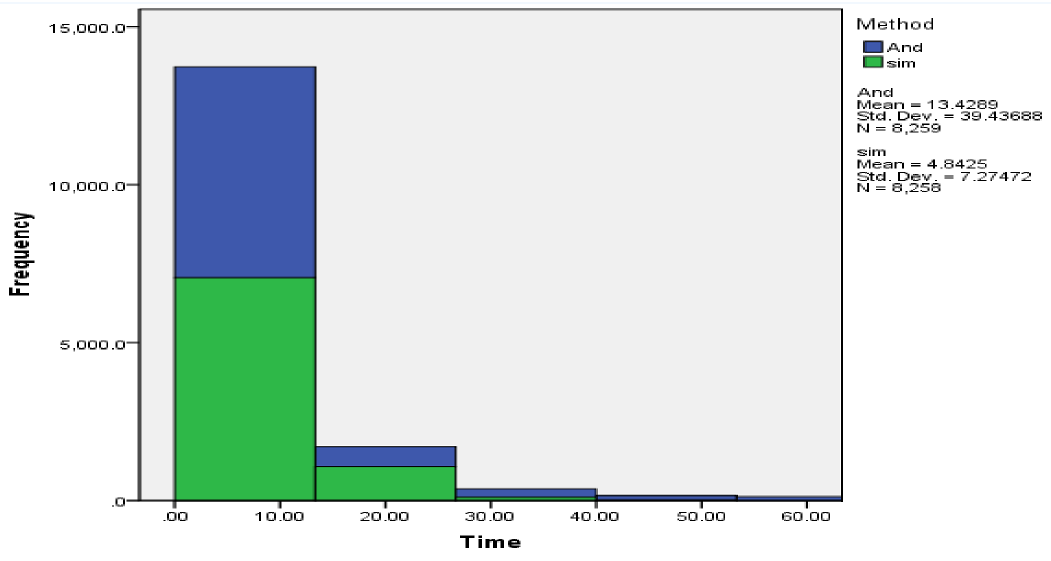

Figure 6: Histogram for SimHash and Androguard (2)

6. Experiment 2: Malware Detection.

The first experiment shows that research proposed algorithm save a lot of memory and time resources during the process of similarity measurement. In this experiment it is important to show the ability of detecting a malware by new method. So the input of this experiment will be a real malware injected application and the same application but without injection, the goal is to detect the injected one by our algorithm. Table 2 describes the dataset information

First of all we use .apk downloaded dataset to generate a database of SimHash signatures, then we insert a test malware in order to find if it will be detected and if it will be similar to another malware.

The same process sequence will be used for Androguard, and the result of two algorithms will be compared.

Table 2: Dataset information

Dataset source link	http://contagiominidump.blogspot.com/
# of applications	130
Types of application	Games, social, education…etc.
Shared by	Mila blog

We used angry birds cheat application as a test application, install this application form Google play insure that it is free of malware and it has been inspected by Google static analysis tool,

In other hand we download the same application but with malware from contagion dump blog, but it is not from dataset we have download before, We enter the tow .apk application in to SimHash and Androguard to measure it similarity with data source elements, first the injected application has been detected by two algorithms since it takes score more than 80% two times as shown in figure 7

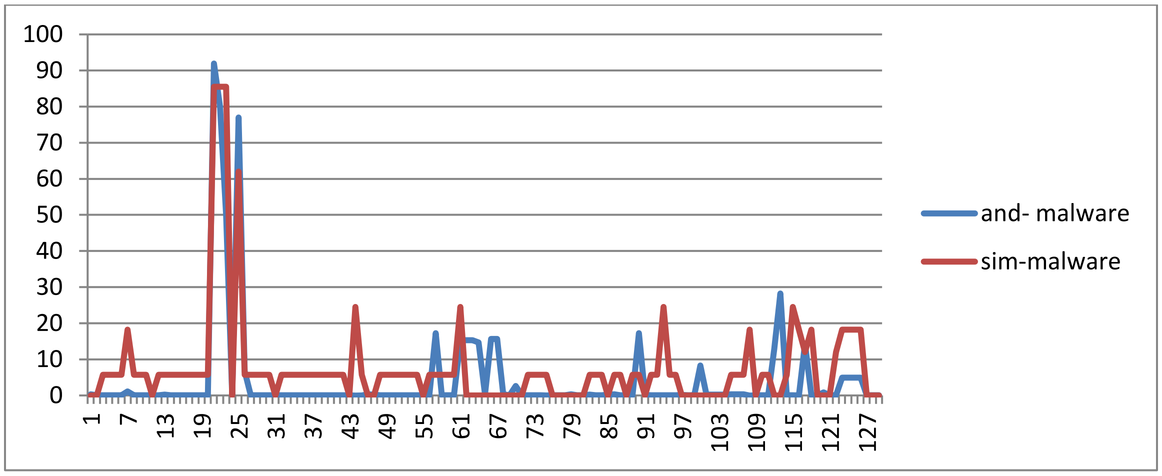

Figure7: Similarity measurement for tested application with malware

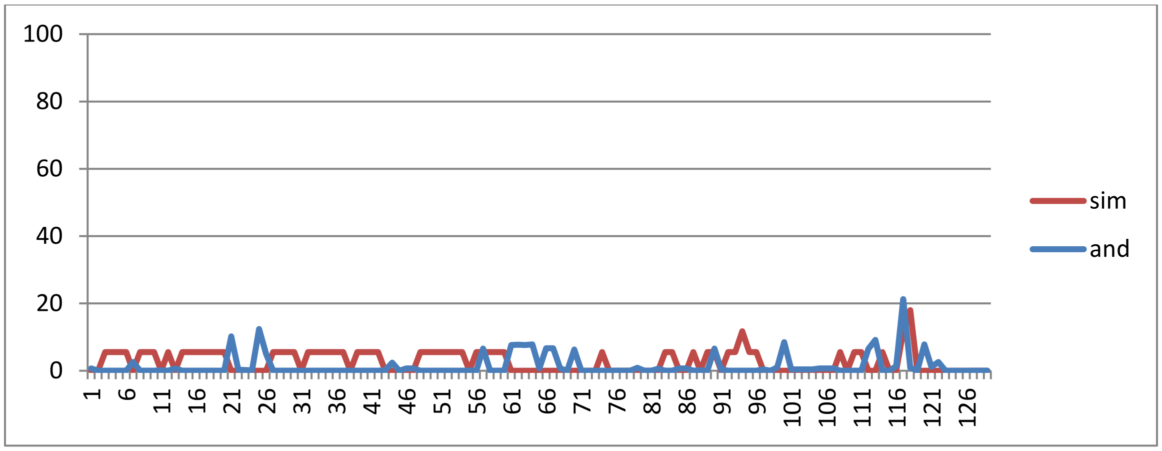

Figure 8: Similarity measurement for tested application with malware

In second graph it is clear that all results are less than 20%, it means that this application is free of malware.

Table 3:Experiment parameters and output

Method	Is injected (yes/no)	Time (mm:ss)
Androguard	Yes	34:59
Androguard	No	32:34
SimHash	Yes	14:24
SimHash	No	10:18

Table 3 shows that our proposed method takes less time than Androguard to generate database of signatures and detect malware this also insure the efficiency of our proposed algorithms

7. Conclusion

In this research, we presented a way to detect android malwares by new method to measure the similarity of android application with dataset of malwares using SimHash algorithm to generate application signatures which performs faster by sacrificing a small accuracy.

Research proposed method has been compared with a well-known tool used by Antiviruses companies for malware detection and similarity measuring, This tool called Androguard, As a result from more than one experiment our method is faster than Androguard with the similar accuracy for overall distance measuring processes, and similar time distribution behavior, this refer to the simplicity of signature generating by SimHash method and similarity measuring of two Hash codes instead of complex method used for Androguard.

8. References

[1] Heloise Pieterse, M. S. (2012). Android Botnets on the Rise: Trends and Characteristics. IEEE.

[2] Seung-Hyun Seo, Analysis on Maliciousness for Mobile Applications. Internet Incidents Response Div., Korea Internet & Security Agency, Seoul, South Korea

[3] Aiman A. Abu Samra, K. Y. (2013). Analysis of Clustering Technique in Android. 2013 Seventh International Conference on Innovative Mobile and Internet Services in

Ubiquitous Computing.

[4] Te-En Wei, C.-H. A.-M.-T.-J. (2012). Android Malware Detection via a Latent Network Behavior Analysis. IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications .

[5] MoutazAlazab, V. L. (2012). Analysis of Malicious and Benign Android Applications. IEEE 32nd International Conference on Distributed Computing Systems Workshops .

[6]ZheMin Yang, M. Y. (2012). “LeakMiner: Detect information leakage on Android with static taint analysis. IEEE Third World Congress on Software Engineering.

[7]Md. Sharif Uddin, C. K. (2011). On the Effectiveness of Simhash for Detecting Near-Miss Clones in Large Scale Software Systems. IEEE 18thWorking Conference on Reverse Engineering

[8] VirusTotal. (n.d.). VirusTotal. Retrieved July 3 2016, from about-VirusTotal : https://www.virustotal.com/en/about

[9] Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. Proc. ACM STOC.

[10] Galopin, T. (n.d.). A web developer blog. Retrieved 3 3, 2014, from http://titouangalopin.com/blog/2013/11/simhash-or-the-way-to-compare-quickly-two-datasets

[11] Mila. (2014, 3 10). Contagi Mobile. Retrieved from (http://contagiominidump.blogspot.com.au/),