Android Malware Detection Framework Based on Static Analysis

Yan Naung Soe, Khine Khine Oo
University of Computer Studies, Yangon, Myanmar
yannaungsoe@ucsy.edu.mm, khinekhineoo@ucsy.edu.mm

Abstract

Mobile devices have gained tremendous popularity over the last few years. The most popular usage is the smart phones. They are accepted and admired by many mainly because they are capable of providing services such as banking, social network, etc all on the go. There are many operating systems used in mobile devices. Among them, IOS and Android systems are the most acceptable technologies. Android platform is the fastest growing market in smart phone operating systems to date. Therefore, the malicious applications targeting the Android system has exploded in recent years. The android malware detection framework is established by the static ways by analyzing the android permission and signature of source codes. For signature based detection, it is used clone detection technique. For permission-based detection, it is detected by using machine learning classifier. By combining with these two approach, this framework improves the performance of the malware detection.

Keywords: *Android, Malware, Mobile Security, Signature, Permission*

1. Introduction

Android operating system is the most targeted smart phone OS. In the growing market of smart phones, Android, and open source platform of Google has become one of the most popular Operating System. Android is mainly used in smart phone and tablets. They equipped with several features such as Wi-Fi, voice, data, GPS, etc. They are capable of providing services such as banking, social networking, etc.

The worldwide smartphone market grew 0.7% year over year in 2016Q2, with 344.7 million shipments, according to data from the International Data Corporation (IDC) [15]. Android dominates the market with an 87.6% share in 2016Q2.

Android has gained popularity mainly because it is an open source operating system and it has some basic features such as middleware in the form of virtual machines, system utilities and a few core applications such as calculator, browser, etc. Mobile malware will continue to be a major security threat. The most attractive feature of smart phone is the ability to extend its functionality with third-party applications. This feature inevitably brings with it the threat of malicious smart phone applications. The increase of smart phone applications causes concern in terms of user security. Smart phones have become a very soft and vulnerable target for malicious application developers.

With a reported over 1 billion smartphones shipped in 2015, this large population of potential victims gives malware writers ample motivation to target smartphones devices, which states that the number of new smartphone malware simples detected has doubled from 1000 per day in 2013 to 2000 per day in 2014 [16]. Based on these facts, it is clear that mitigating the threat of smartphone malware is an important problem for the security community. The android malware increased to the double rate within 2014 and 2015. In the Trend Micro 2016 Security Predictions report, CTO, Raimund Genes predicted the following: China will drive mobile malware growth to 20 million by the end of 2016 [17].

There are many techniques for detecting of android malware. It can be categorized into two separated groups such as static analysis and dynamic analysis. The static analysis uses the reverse engineering techniques. The dynamic technique is the type of real time analysis. All of these techniques have several pros and cons. Another mechanism is as inthe hybrid analysis by combining of static and dynamic techniques. To maintain security for the system and users, Android requires apps to request permission before the apps can use certain system data and features. Depending on how sensitive the area is, the system may grant the permission automatically, or it may ask the user to approve the request.

In this framework, there are two approaches such as signature based detection and permission based detection. The clone code analysis is used for

signature based detection and one of the machine learning classifier is used for permission based detection. In this paper, it is categorized the types of mobile malware and how it is affected in each year. And then, the android malware detection framework is established by the use of android permission and machine learning classifier. Section 2 expresses the related work for android malware detection in static and dynamic analysis. Section 3 consists of the challenges of android malware. Section 4 consists of the propose approach for android malware detection. In section 5 evaluates the performance analysis of this framework and the paper will conclude in section 6 how it is effective this framework.

2. Related Work

There are many detection techniques for android malware that was implemented by various researches. Most of them are based on static analysis. Some are based on dynamic and hybrid analysis. Signature based, Dalvik bytecode based and permission based analysis are also static analysis. The following is the recent malware detection approach in each research work.

Faruki et al. proposes Androsimilar for android malware detection with robust statistical feature signature [9]. It creates variable length signature and compares with signature database and uses fuzzy hashing technique. And then, it generates the result by differentiating between benign and malicious apps on the basis of similarity percentage.J. Chen et al. express the detecting of android malware using clone detection [10]. It shows the clone similarity between android apps by using file selection, reverse engineering and clone detection. There are two parts such as signature detection and signature matching in clone detection. They express that their approach can detect malware with high accuracy.

C.Y. Huang et al. propose their research work with the performance evaluation on permission-based detection for android malware [11]. It analyzes the required and requested the permissions for application and labels the apps as benign or malware using site based, scanner based and mixed labeling. And then, it uses machine learning algorithms on three data sets and evaluates the permission based malware detection performance.

S.M. A. Ghani et al. presented the static analysis technique that extracted the benign and malware application to get their source code [2]. It will be compared and categorized into API and manager classes. The most frequent API and manager class used in malware will be detected. They extracted the feature by using Androguard, a reverse engineering tool and compared the extracted source code by categorized the APIs and manager classes.The result in this paper shows the relationship between the most used API and manager classes in malware.

Karlsen et al. presented the first formalization of Dalvik Bytecode along with java reflective features [12]. Theyexamined 1700 popular Android Apps to determine what Dalvik Bytecode instructions and features are mostly used bythe Android Apps. Such formalization helps to perform controland data flow analysis in order to detect the malicious apps or to identify the sensitive API calls invoked during execution. Itsupports the dynamic dispatch and reflective features. But itrequires extension in analysis of concurrency and reflectionhandling.

3. Challenges of Android Malware

Mobile malware is malicious software that targets mobile phones by causing the collapse of the system and loss of leakage of confidential information. The first known mobile virus, “Timofonica”, originated in Spain and was identified by antivirus labs in Russia and Finland in June 2000 [18].

3.1. Types of Malware

The top android malware families are shown in Fig.1. Trojan is the most spread types in android malware. All types of Trojan malware are totally 60.16% all of malware [19]. The second more attack type is the Advertising Malware (Adware).

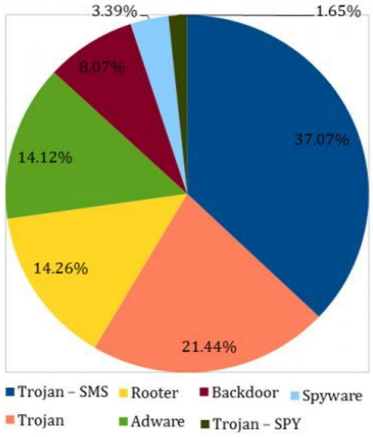

Figure 1. The growing threat of android mobile malware

The behavior of different malware families is provided in subsequent sections.

- *Trojans:*Trojans appear to a user as Benign application. In fact, it is actually steal the user's confidential information without the user's knowledge. Such applications can easily get access to the browsing history, messages, contacts and device IMEI numbers [8]. Mobile banking Trojans can run together with Win-32 Trojans to bypass the two-factor authentication – mTAN theft (the theft of banking verification codes that banks send their customers in SMS messages). However, in 2013, autonomous mobile banking Trojans developed further. Currently, such Trojans attack a limited number of bank customers, but it is expected that cybercriminals will invent new techniques that will allow them to expand the number and the geography of potential victims.
- *Rooter:*Originally, the word "root" is used to refer to the root account on Linux, that is to say, the system administrator, who has all the rights on the device and can modifyall OS elements as it sees fit, including sensitive files. So rooter phone or tablet, it means the system administrator become "all powerful" and have all the rights on the Android phone to do many wonderful things and satisfy the fans of customization and crafts. The root may include a phone blocking risk.
- *Adware:*Adware is software that contains advertisements embedded in the application. Adware is considered a legitimate alternative offered to consumers who do not wish to pay for software. There are many ad-supported programs, games or utilities that are distributed as adware.
- *Backdoor:*A backdoor is a means of access to a program that bypasses security mechanisms [20]. A programmer may sometimes install a back door so that the program can be accessed for troubleshooting or other purposes. Backdoors employ the root exploits to grant root privileges to the malwares and facilitate them to hide from antivirus.
- *Spyware:*Mobile spyware has been reported numerous times in the media as a serious threat for phone users. Spyware threats are also highly pervasive—according to security company Lookout, .24% of Android phones they scanned in the U.S. had surveillance-ware installed intended to target a specific individual.
- *Botnet:*Botnet is a network of compromised Android devices. Botmaster, a remote server, controls the botnet through the Command and Control Server (C&C) network [11]. The botnet tendencies to actually hijack and control infected devices.

3.2. Malware Infections

A new 'RAT' or a Remote Access Tool has been discovered running on the Android platform. While RATs are common on the open source software, this one has serious damage wrecking potential. The HijackRAT discovered by experts combines several malicious tasks into one package [21]. These include executing privacy leakages, stealing banking credentials and having a remote access to your data/device.

A framework was found which is designed for bank hijackings. Starting from South Korea, currently eight banks are on the attacker's list. But the hacker has the potential to expand to new banks with just 30 minutes of work. Both the developer of the malware and its victims are Korean speakers. Even more worryingly, this malware has an extremely low detection rate. Only 5 out of 54 antivirus programs were able to detect the malware. This is primarily due to its ability to change its command and control servers.

The mobile phone Trojan family, known as "Hummer" [22], gained traction in early 2016 when it was infecting nearly 1.4 million devices daily at its peak, according to Cheetah Mobile. Hummer is thought to have originated in China relative to underground industry there, based on an email address linked to the domains used, and it saw 63,000 daily infections in China alone.

It will root the phone to gain admin privileges. This leads to frequent pop-up ads and background installation of unwanted apps, games, pornographic applications and malware. Even if a user uninstalls these apps, the Trojan will reinstall them. In several hours, the Trojan accessed the network over 10,000 times and downloaded over 200 APKs, consuming 2 GB of network traffic.

Android malware can also be infected the apps on Google Play Store. There are some infected apps [23] in 2016 is shown in Table 1.

The following are four different methods which malware can be infected on the devices:

- Repackaging Legitimate Application
- Exploiting Android's Application Bug
- Fake Applications
- Remote Install

Table 1. Some of Infected Apps in 2016

APKs Name	Malware Types	Infected Effect
Xiny	Notorious Trojan	Intercepts IM chat messages and targets online bankers
MoDaCo	Phishing Spyware	To have broken in via a compromised admin account
CallJam	Remote Access Trojan	Keeps ringing premium rate numbers
DressCode	Botnet	Initiates communication with its command and control server.

4. Proposed Methodology

This framework aims for developing the detection system that focuses on feature collection extraction to classify the malicious applications upon the android permission and code analysis by clone detection. Figure 2 shows the overall process of the proposed approach.

It is a framework for detection of android malware using signature based and permission based detections.It is needed the reverse engineering process for generating the AndroidManifest.xml and java source code. It is needed to enter the android app as the input and this framework will analyze the app is benign or malicious.

It sets the android apps from real world android market as the testing data and it uses the malware apps from the two sources from ContagioDump and Kaggle as the training data. For signature based detection, it uses NiCad clone detector. If the clone detection result is higher similarities, it will continue the second phase for permission based analysis. In this process, one of the machine learning algorithms is used for classification. Back propagation algorithm is used for the android malware classification.

4.1 Feature Collection and Extraction

The android apps used for testing datasets are collected from popular apps stores such as Google Play, 1Mobile Market, Mobango and so on. Some of the apps are same and most for them are different categories. A total of 600 apps are collected from these apps stores and malicious apps are based on ContagioDump project and the total number of malware apps included in the sample of 189. Another dataset are collected from Kaggle website. It has 398 records of android apps permissions for tested apps.

The first process is the signature base detection and it needs java sources to check the clone codes. To apply the clone code detection, it is needed to transform the android apps to java source codes. Dex2jar is used to disassemble apk files.The second process is the permission based detection and it needs the android permission of the collected apps. Apktool is used to generate the Androidminifest.xml from apk files.

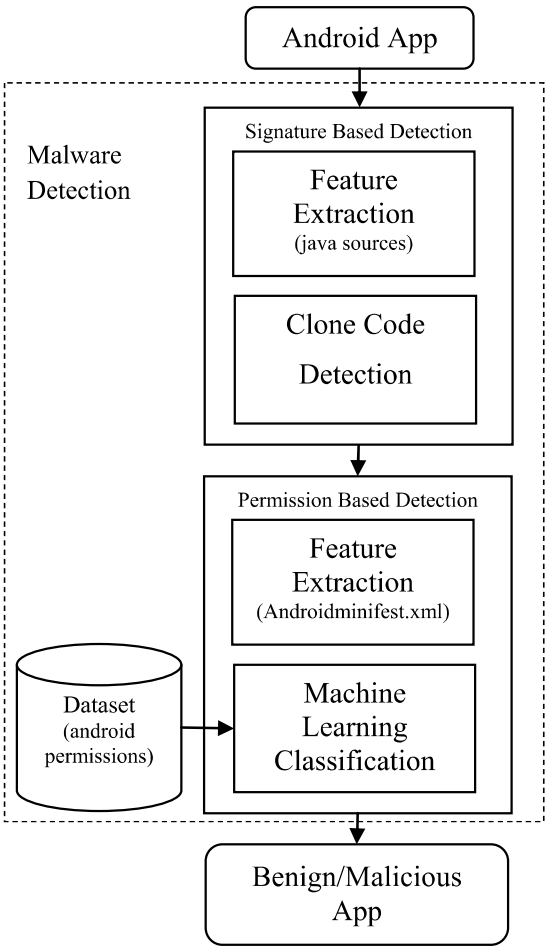

Figure 2. Framework for Android Malware Detection System

The permission database include the various attributes and the values of selected features are stored as abinary number (0 or 1), which is represented as a sequenceof semi-comma separated values. The sample records are shown in Table 2. There are 329 permissions attributes and a result attributes. The first attributes are the use of android permission and the last attribute indicates the result 1 as malware and 0 as benign app.

Table 2. Sample of Android Permission from the Dataset

Benign App	Malicious App
0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;1;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;1;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;…… ………………;0	0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;1;0;0;0; 0;1;0;0;0;0;0;0;0;1;0;0; 0;1;0;0;0;0;0;0;0;0;0;1; 0;0;0;0;0;1;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;…… ………..;1
0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;1;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;1;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;… ……………….;0	0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;1;0;0;0; 0;1;0;0;0;0;0;0;0;0;0;0;0; 1;1;1;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;1;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;0;0; 0;0;0;0;0;0;0;0;0;0;0;… …………;1

4.2 Signature Based Detection

In the signature based detection, it is used the way of clone code analysis. Clones are segments of code that similar according to some definition of similarity. Roy et al. demonstrated that NiCad [13] can yield an outstanding result among the text-based techniques and tools to identify the malicious code clones in malware apps. There are two sets of results generated by NiCad, each reported in both HTML and XML formats. First, the results of the comparison are reported as clone pairs. A clone pair is a pair of code fragments for which one is a clone of the other. Second, the clones in the input source are grouped into clone classes. Each clone class contains all the clones in the input which are similar and differ in the number of lines only up to the specified difference threshold. Both formats contain the source of the clones, specifying the degree of similarity, start and end line numbers of the clones found, and the size of the clones. NiCad provides the ability to find clones at various granularities (classes, functions, blocks, statements, etc.), with varying degrees of near-miss similarity. It is given a set of code files to identify the clones and clusters these clones into classes. And it can find elements in the second set that are similar within the threshold the elements of the first set. It is supported for java source codes. Figure 3 shows the clone detection process of NiCad Clone Detector.

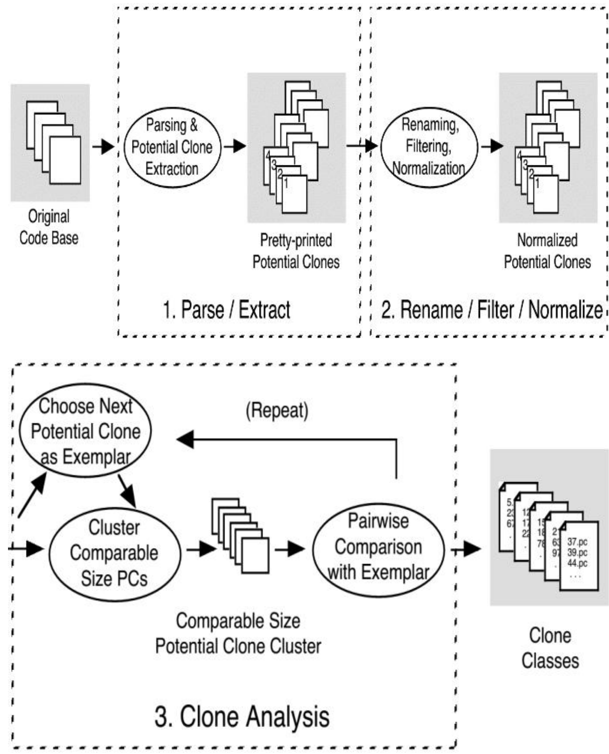

Figure 3. NiCad Clone Detector

4.3 Permission Based Detection

The permissions are required to create android apps and the permission access is stored in the AdroidManifest.xml file. While installing the android apps, the user must allow the application permission.

But, some of apps embedded the unnecessary permission in the apps. It can join the malicious code in the executing java code and it can steal the user data as a backdoor nature. In this analysis, it also needs the reverse engineering tool such as apktool for re-generate the AndroidManifest.xml file. After getting this file, it can detect the malicious apps by using machine learning classifier.

Artificial Neural Networks (ANN) is important machine learning tool used for classification and clustering.It is an attempt to build machine that will mimic brain activities and be able to learn. ANN is able to perform classification and even discover new trends or patterns in data. Basic ANN is composed of three layers, input, output and hidden layer. Each layer can have number of nodes and nodes from input layer are connected to the nodes from hidden layer. Nodes

from hidden layer are connected to the nodes from output layer. Those connections represent weights between nodes.

4.3.1. Backpropagation Algorithm

Backpropagation is well known algorithm for ANN. It could be broken down to four main steps [14]. After choosing the weights of the network randomly, theback propagation algorithm is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps:

- Feed-forward computation
- Back propagation to the output layer
- Back propagation to the hidden layer
- Weight updates

The algorithm is stopped when the value of the error function has become sufficiently small. Figure 4 shows the pseudo code for backpropagation algorithm. The network diagram is shown in Figure 5 where I1 to I_n is for inputs, H1 to H_n is for hidden neurons and B1, B2 is for bias inputs. There are two output nodes (O1, O2) for benign or malicious apps in this framework.

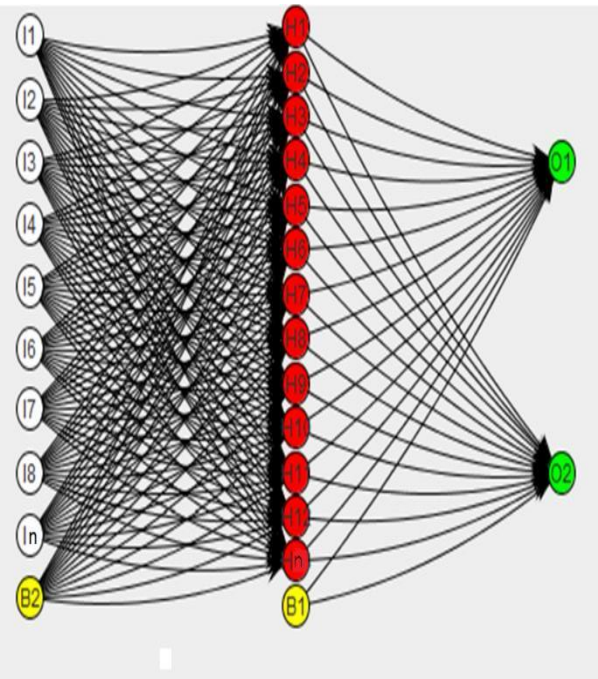

Figure 5. Network Structure for ANN

5. Performance Evaluation

The clone detection is the first process of this framework. To detect the malware, the simples of

```
Assign all network inputs and output
Initialize all weights with small random numbers, typically
between -1 and 1
repeat
   for every pattern in the training set
      Present the pattern to the network
//       Propagated the input forward through the network:
         for each layer in the network
            for every node in the layer
               1. Calculate the weight sum of the inputs to
the node
               2. Add the threshold to the sum
               3. Calculate the activation for the node
            end
         end
//       Propagate the errors backward through the network
         for every node in the output layer
            calculate the error signal
         end
         for all hidden layers
            for every node in the layer
               1. Calculate the node's signal error
               2. Update each node's weight in the network
            end
         end
//       Calculate Global Error
         Calculate the Error Function
   end
while ((maximum  number of iterations < than specified)
AND  (Error Function is > than specified))
```

Figure 4. Pseudo Code for Backpropagation Algorithm

apk are selected from different markets with the same signature.It can be downloaded as android apps (apk). All of them can be used for clone code analysis as signature based detection. If the app is surely being clone of the another, the second process will continue. For the permission-based detection, the total number of malware apps included in the sample of 189 from ContagioDump project. It is split into four categories and these are shown in Table 3.

Table 3. Contagio Malware Dataset

ID	Type	Total Apps
1	Trojan	68
2	Privilege	19
3	Info Steal	88
4	SMS	14

Another dataset is collected from Kaggle website. It includes 198 malicious apps and 200 benign apps. It is categorized the records based on 330 android permissions types. This dataset can be used for permission based detection as a malware dataset. So, the total numbers of 1187 android apps are used for this android malware detection framework.

Most of the malicious apps are commonly used these permissions as shown in Table 4.

Table 4. Some Android Permissions in Malicious Apps

Permissions	Description
android.permission. READ_CONTACTS	Allows an application to read the user's contacts data.
android.permission. WRITE_CONTACTS	Allows an application to write (but not read) the user's contacts data.
android.permission. INTERNET	Allows applications to open network sockets.
android.permission. READ_LOGS	Allows an application to read the low-level system log files.
android.permission. WRITE_SETTINGS	Allows an application to read or write the system settings
android.permission. READ_SYNC_SETTINGS	Allows applications to read the sync settings
android.permission. READ_SMS	Allows an application to read SMS messages.

The evaluation performance classifies as the following criteria.

- **True Positive (TP):** Number of benign apps correctly classify.
- **False Positive (FP):** Number of malicious apps correctly classify.
- **True Negative (TN):** Number of benign apps wrongly classify.
- **False Negative (FN):** Number of malicious apps wrongly classify.
- **True Positive Rate (TPR):** Percentage of correctly identified benign apps.
  TPR = TP/(TP+FN) (1)
- **False Positive Rate (FPR):** Percentage of wrongly identified malware apps.
  FPR = FP/(TN+FP) (2)
- **Accuracy (ACC):** Percentage of correctly identified apps.
  ACC = (TP+TN)/(TP+TN+FP+FN) (3)

The NiCad clone detector detects how many clone pairs in similar apps. The clone detection processing is used for signature-based detection. Some simple results are shown in Table 5. As a result, the simple apps (S1 and S2) have 49 pairs of clone classes and the similarity of these apps are 72%. Among them, one of these apps may be malicious apps and it is needed to check the permissions of these apps. Another comparison of S1 with S6, there aren't any similarity pair.

Table 5. Clone Detection with NiCad

Apps	S4	S5	S6
S1	149/ 72%	23/ 41%	0/ 0%
S2	0/ 0%	3/ 2%	290/ 97%
S3	124/ 65%	155/ 76%	0/ 0%

The classification result is shown in Table 6. The experiment result shows with two categories of dataset. For the first dataset, the result is 0.950 in TRF and 0.238 in FPR with the use of this framework. And, it shows the 0.817 in TRF and 0.670 in FPR with the use of permission only. For the second one, the true positive rate is 0.961 and false positive rate is 0.026 with 97 percent accuracy for detecting of signature and permission. And, there are more accurate with the result of 97 percent than permission only detection.

Table 6. Evaluation of Classification Result

Dataset	Analysis Type	TPR	FPR	ACC
1 (Contagio)	Permission and Signature	0.950	0.238	90
	Permission only	0.817	0.670	62
2 (Kaggle)	Permission and Signature	0.961	0.026	97
	Permission only	0.688	0.272	65

The accuracy result of framework is higher than permission only detection. For permission-based detection, it uses the backpropagation algorithm and the learning rate is calculated by the minimum error with 0.01 and maximum iteration up to 1000. Some of the permission usage of benign apps is similar with malicious apps. This framework can get more accurate result due to perform not only clone detection but also permission based detection.

The comparison of accuracy result is shown in Figure 6. There are two datasets such as Contagio and Kaggle for the analysis. This figure shows the accuracy result of the combining of permission and

signature based detection is more than 90. It indicates that this framework is more suitable for android malware detection.

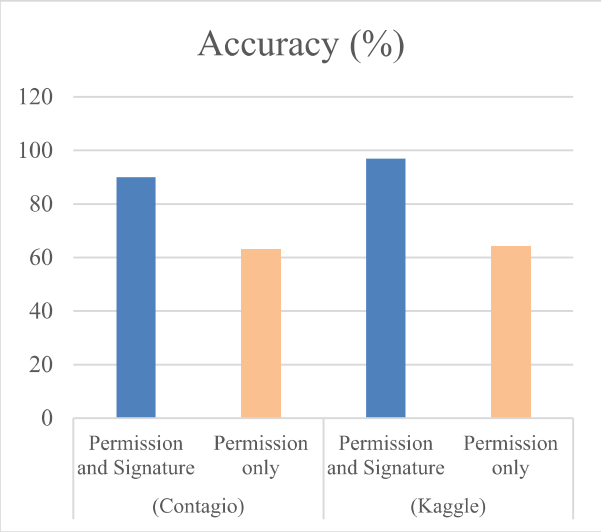

Figure 6. Accuracy of this Framework

6. Conclusion

The implemented framework is used for detection of android malware using signature based and permission based detections. For signature based detection, it uses NiCad clone detector. Due to this process, it only needs to check for similar apps. For permission-based analysis, it uses back propagation algorithm. Only permission based detection, it has some difficulties for deciding the two apps with the same types of permission; one can be benign and another can be malicious. As the experiment results, the usage of this framework is more accurate than the permission only detection. It can generate more suitable result for classification of benign or malicious apps. This framework can only classify upon the known malware type.

References

[1] Y. Zhou and X. Jiang, *"Dissecting Android Malware: Characterization and Evoluation"*, IEEE Symp, 2012.

[2] S .M. A. Ghani, M. F.Abdollah, R.Yusof, M. Z.Mas'ud, *"Recognizing API Features for MalwareDetection Using Static Analysis"*, Journal of Wireless Networking and Communications 2015, 5(2A): 6-12.

[3] K.Shaerpour, A.Dehghantanha, R.Mahmod, *"Trends in Android Malware Detection"*, Journal of Digital Forensics, Security and Law, Vol. 8(3).

[4] H.Gascon, F. Yamaguchi, D. Arp, K.Rieck, *"Structural Detection of Android Malwareusing Embedded Call Graphs"*, ACM, Berlin, Germany, November 4, 2013.

[5] J. Sahs, L. Khan, "A Machine Learning Approach to Android Malware*Detection"*, European Intelligence and Security Informatics Conference", 2012.

[6] D. E. Krutz, M. Mirakhorli, S. A. Malachowsky, A. Ruiz,J. Peterson, A.F ilipski, J. Smith, *"A Dataset of Open-Source Android Applications"*, 12th Working Conference on Mining Software Repositories, 2015.

[7] G.Kapse, A. Gupta, *"Detection of Malware on Android based onApplication Features"*, International Journal of Computer Science and Information Technologies, 2015.

[8] Nancy, D. Sharma, *"Android Malware Detection using Decision Treesand Network Traffic"*, International Journal of Computer Science and Information Technologies, 2016.

[9] P. Faruki, V. Ganmoor, V. Laxmi, M. S. Gaur, and A. Bharmal, *"AndroSimilar: Robust Statistical Feature Signature for Android Malware Detection",* Proc. 6th Int. Conf. Secur. Inf. Networks, pp. 152–159, 2013.

[10] J. Chen, M. H. Alafi, T. R. Dean, Y. Zou, *"Detecting Android Malware Using Clone Detection"*, Journal of Computer Science and Technology, pp. 492-956, September, 2015.

[11] C.Y. Huang, Y.T. Tsai, and C.H. Hsu, *"Performance evaluation on permission-based detection for android malware",* Adv. Intell. Syst. Appl. - Vol. 2, vol. 21, pp. 111–120, 2013.

[12] E. R. Wognsen, H. S. Karlsen, M. C. Olesen, and R. R. Hansen, *"Formalisation and analysis of Dalvik bytecode",* Sci. Comput. Program., vol. 92, no. December 2012, pp. 25–55, 2014.

[13] J. R. Cordy, C. K. Roy, *"The NiCad Clone Detector"* , ResearchGate, June, 2011.

[14] B. Krose, P. V. D. Smagt, *"An Introduction to Neural Network"*, 8th edition, November, 1996.

[15] *"Smartphone OS Market Share, 2016 Q2"* http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[16] *"Cumulative Number of Android Malware in 2015"* https://www.itvoice.in/index.php/it-voice-news/android-malw are-doublyed-in-2015-vs-2014-reports-trend-micro-2015-threat-report

[17] *"Continued Rise in Mobile Threats for 2016"* http://blog.treandmicro.com/continued-rise-in-mobile-threats-for-2016

[18] *"Mobile Malware"* http://en.wikipedia.org/wiki/Mobile_malware

[19] *"The Growing Threat of Mobile Malware"* http://blogarchive.quickheal.com/wp/the-growing-threat-of-mobile-malware-top-android-malware-families-of-2012/

[20] *"Backdoor"*http://searchsecurity.techtarget.com/definition/back-door

[21] *"Structure of Hijack RATMalware"* http://wccftech.com/android-rat-appears-banking-threat-horizon-researchers/

[22] *"Android Malware Hummer could be biggest Trojan ever"* http://www.techrepublic.com/google-amp/article/1-2-million-infected-android-malware-hummer-could-be-biggest-trojan-ever/

[23] *"400 Android apps hiding DressCode malware on Google Play Store"* https://www.grahamcluley.com/mobile/android/