Malware Detection Techniques in Android

Pallavi Kaushik
PG Student CSE Department,
Panchkula Engineering College,
Mouli, Haryana, India

Amit Jain
Astt. Prof. & HOD: Dept. of CSE,
Panchkula Engineering College,
Mouli, Haryana, India

ABSTRACT

Mobile Phones have become an important need of today. The term mobile phone and smart phone are almost identical now -a-days. Smartphone market is booming with very high speed. Smartphones have gained such a huge popularity due to wide range of capabilities they offer. Currently android platform is leading the smartphone market. Android has gained an overnight popularity and became the top OS among its competitor OS. This eminence attracted malware authors as well. As android is an open source platform, it seems quite easy for malware authors to fulfill their illicit intentions. In this paper a new technique will be introduced to detect malware. This technique detects malware in android applications through machine learning classifier by using both static and dynamic analysis. This technique does not rely on malware signatures for static analysis but instead android permission model is used. Under dynamic analysis, system call tracing is performed. Using both static and dynamic techniques along with machine learning provides all in one solution for malware detection. The technique used by us is tested on various benign samples collected from official android market (Google Play Store) and on various malicious applications.

Keywords

Android, Dynamic Analysis, Machine learning, Malware, Malware detection, Static Analysis.

1. INTRODUCTION

Smartphones are used worldwide. Smartphones provide so many features now a days that they are almost equivalent to mini computers. Smartphones facilitate us in many areas like short message service, multimedia messaging service, email, video calling, GPS navigation, voice dictation, voice activated personal assistant and now they take care of our health as well, by containing various applications and sensors that helps us in keeping track of our health related problems. There are many operating systems present in the smartphone market but android leads them. Android platform was launched by Google and Open Handset Alliance in September 23, 2008 and since then it got an overnight popularity because of its user friendliness and ease of developing and publishing applications in android market. Android has total share of 78% in smartphone market in first quarter of 2015 as per reports of International Data Corporation(IDC)[1](Figure 1). Open source availability and popularity of android pleased malware authors as well. As android has largest share in smartphone market, number of attacks on android platform are also very large as compare to other platforms. One reason for the increase in number of android malware is that any developer can develop his application and publish it in android market. Though official android market Google PlayStore is still very much secure as compare to other third party markets.

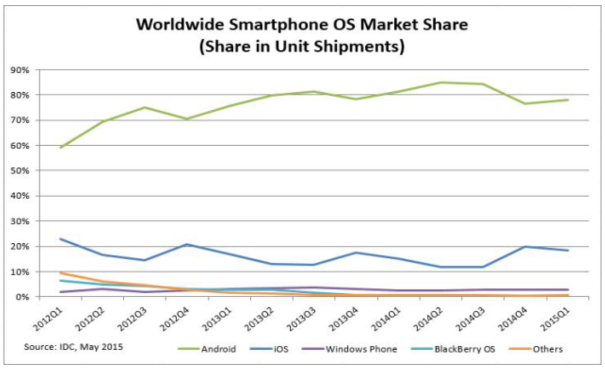

Figure 1: Android market share in first quarter of 2015

Most of the malware come when user unknowingly install applications from third party application stores and many applications in these unofficial stores are the repackaged version of original applications that are present in Google PlayStore. First mobile malware Cabir[2] was detected in 2004 for Symbian mobile operating system. Since then there was gradual increase in number of malwares in mobile market. Various techniques have been used so far to detect malware on mobile platform. Techniques that are used for detecting android malware can be classified into static techniques and dynamic techniques. Static techniques does not execute an application and detects if an application is malicious or not by checking various parameters like signature verification, permission analysis etc. Static techniques alone can be easily obfuscated. Dynamic techniques include analyzing the behavior of an application by executing it in a confined environment like in sandbox. In dynamic analysis some features of an application are extracted to find out if that application is benign or malicious. These features can be api tracing and network monitoring but they are also not all in one solution. Machine learning techniques have been used widely for malware detection.

With all these details, our method for detecting malware on android platform use static analysis by analyzing permissions of an application and then that application is forwarded for dynamic analysis where its system calls are monitored. When both these features are extracted from 219 applications, then machine learning classifiers will be trained with the mentioned list of samples. Once those classifiers are trained, accuracy of our results will be validated . Benign applications are taken from Google's official android market that is Google PlayStore[3] and malicious applications are taken from publically available collection of android malware website, Contagio Mobile Malware Mini Dump[4].

Rest of the work is organized as follows. Section 2 contains related work done in the field of malware detection. Section 3 tells about the dataset that is collected from various sources. Section 4 includes framework used in this paper for malware detection. Section 5 concludes whole paper and section 6 discusses future work that can be done to improve techniques of malware detection.

2. RELATED WORK

Initial techniques that were used for detection of malware in android based phones were those of power consumption and battery usage. Application that uses maximum battery power and CPU resources was seen as malicious application[5]. Mentioned in [6] a power aware malware detection framework that monitors, detects and analyses previously unknown energy depletion threats. But these techniques are not suitable for current scenario as today's mobile devices perform functions almost equivalent to computers so they consume lot of power. So, techniques that were used to detect malware by the amount of power a particular application consumes are obfuscated now.

After that there were some researches that used static techniques for malware detection. Technique used in [7] extracts function calls from Android environment using command readelf. This function call list is then compared with malwares and classified with three algorithms and then collaborative malware detection approach was used to extend the results. Leonid Batyuk et al.[8] proposes a service that access android market applications via static analysis and provide detail reports to the user and then they describe a means to mitigate security and privacy threats by applying automated reverse engineering techniques and refactoring binary application packages. Sanz et al. [9] extracted permissions from applications and used machine learning techniques to detect malware. They used Waikato Environment for Knowledge Analysis(WEKA) and k-fold cross validation to evaluate the performance of machine learning classifiers.

There are other researches that used dynamic techniques to detect malicious behaviour in an application. Enck et al.[10] proposed a system called TaintDroid, that provided system wide taint tracking. In this, applications are passed to Dalvik VM where they are further processed and tracking is done at four levels: variable-level, method-level, message-level, file-level. These tracking techniques record any data that originates from taint sources like from camera, location and from other identifiers. Untrusted applications cannot execute native methods directly as all native libraries are called from virtual machine. At taint sink, all events that are marked as taint are logged. False positives and false negative may occur in taintdroid. Burguera et al.[11] used dynamic analysis for the detection of malicious applications in their framework. Framework consists of many components which complement each other. Their main approach was to make a framework that can easily distinguish between applications with same name and versions but those behave differently(Trojan Horse). Main component of this framework is a client application named as "Crowdroid". This application monitors all the system calls at kernel level. This framework uses crowdsourcing system to obtain behaviour of applications. Logs are created by using a tool Strace after behaviour of application is analyzed. Strace collects all the system calls and then these system calls are forwarded to remote server which will then parse data and record information regarding these system calls, as number of times each system call is used. By using this information, a dataset of benign applications will be created. The detection component uses K- means clustering algorithm for all system calls dataset and will detect any anomalous behaviour. Grace et al. (2012) [12] analyzed applications both from official marketplace and unofficial marketplace. Their main focus was to process maximum number of applications in minimum number of time and to distinguish them as malicious or benign. They performed Two-Order risk analysis. In first order risk analysis, they directly identify applications in high and medium risk categories. During first order analysis, an application is of high risk if it exploit platform level vulnerability in kernel and an application is of medium risk if it charges money from the user or upload any private information and other credentials of the user on the remote server. In second order risk analysis, further investigation is performed to uncover any suspicious behaviour of an application. In this phase, analysis will be performed to detect an application that remains hidden in first phase or any application that might be encrypted. Portokalidis et al.[13] introduced a framework in which all analysis and computations related to security are moved to cloud that is on the remote server. That server has multiple copies of mobile phones that run on emulators. A tracer is located in the smartphone of user that records all the important information which is required to replay that particular application on the server. The recorded information is sent to the replayer, which is located on cloud. Replayer replays whole application in the emulator. During replay of an application various security checks can be deployed such as dynamic analysis can be done there, antivirus scanning, system call anomaly detection or memory scanners. Su et al. [14] collected two sets of features by dynamically analyzing new applications. These two sets of features are system calls and network traffic traces. For system call monitoring Strace was used and it restricted itself to 15 activities that were associated with process, input and output activities and memory. To capture network traffic they used tcpdump tool. The accuracy rate of their method was 94.2% for J48 and 99.2% for RandomForest.

3. DATASET

The authenticity of any malware detection system depends upon the dataset quality which is being used to test the system. We have collected total 219 applications from various sources. Benign applications are taken from official android market and malicious are taken from a public source.

3.1.1 Benign dataset

Benign applications are taken from Google PlayStore[3]. They are mixture of applications from various categories of games, music, news, entertainment, Image Viewers, Maps and Recorders. There are total 107 benign applications. Efforts are being made to maintain the diversity among the applications collected.

3.1.2 Malicious dataset

Malicious applications are taken from publically available repository of all malwares for android phones - Contagio Mobile Malware MiniDump[4]. These malwares belong to Virus, Bots, Worms, Trojans, Rootkits, Spyware, Backdoors, Adware categories. There are total 112 malicious applications. Malicious applications are taken in large quantity for better training and detection so that more accurate results can be obtained. Most of the malicious applications consists of Trojan horses/Viruses.

4. FRAMEWORK

For detecting malicious applications, both static analysis and dynamic analysis will be used and then features will be extracted by using both of these techniques and lastly will use WEKA(Waikato Environment for Knowledge Analysis)[15] to train machine learning classifiers and to validate our result using k-fold cross validation. First step will be of extracting features using static analysis then in second step we will use dynamic analysis and lastly machine learning will be used.

4.1 Static Analysis

In this step, permissions of an application will be extracted. Permissions will be extracted from Manifest file by using aapt(Android Asset Packaging Tool). When user downloads application from any of the application store he is presented with a list of permissions during installation of that application. It is upto him whether to accept those permissions or decline them. If user accepts those permissions then application will be installed and if user decline those permissions application will not install. Mostly users accepts all permission without even looking what that permission list includes. Moreover if user find any permission suspicious he has no other option to remove that permission from the list of given permissions, user only can either accept whole list or decline whole list of permissions.

In static analysis, we will extract permissions from Android Manifest.xml file by using aapt and figure out following things:

i. Number of permissions used by each application

ii. List of permissions that are asked by an application

Permissions can be extracted from an application package without executing that application. During static analysis we are finding number of permissions that are used by both benign applications and malicious applications and if there is any difference in the number of permissions they ask. Number of permissions that are asked by both benign applications and malicious applications does not vary that much as shown in Figure 2.

Secondly, we will analyse the list of permissions asked by an application and will find out the difference between permissions asked by a benign application and by a malicious application. We will find out which extra permissions malicious applications will ask. There is also not much difference in the permission list that is used by malicious applications and benign applications. Both use almost same set of permissions but there are some malicious applications that use only single permission.

We used following command to extract features from an application package:

```
./aapt dump permissions <sample file location>
```

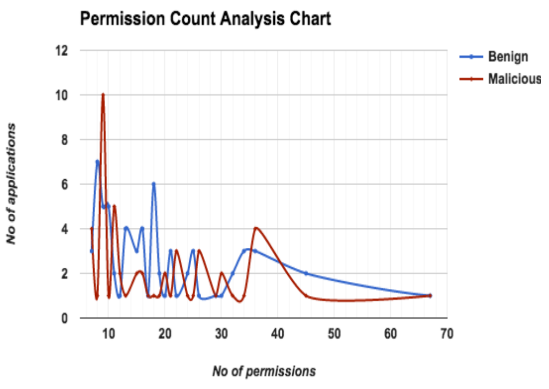

Figure 2: Number of permissions for benign and malicious applications

4.2 Dynamic Analysis

Dynamic Analysis will be done by tracing system calls. Whenever an application executes it makes use of various system calls to perform its functions. Things that will differentiate between benign applications and malicious applications are:

i. Number of system calls that are used by both benign and malicious applications?

ii. List of system calls used by both malicious and benign applications

System calls for both benign and malicious applications are extracted by using Strace (System Tracer) tool[16]. By tracing system calls with the help of Strace we will get list of all the system calls used by a particular application. Then the unnecessary system calls are filtered out and all the necessary system calls will be recorded. Then comparison of those system calls will be made based on above points to find out which system calls benign applications use and which system calls malicious applications use. Figure 3 shows the number of system calls that are being used by both benign and malicious applications.

4.3 Machine Learning

Once all the features are extracted by using both static and dynamic analysis, next step is to provide input of these features to a machine learning tool that will use different classifiers and will validate our method. We will use supervised machine learning for this purpose. Tool that we will use for all this purpose is WEKA. WEKA will be trained first by providing all the extracted features.

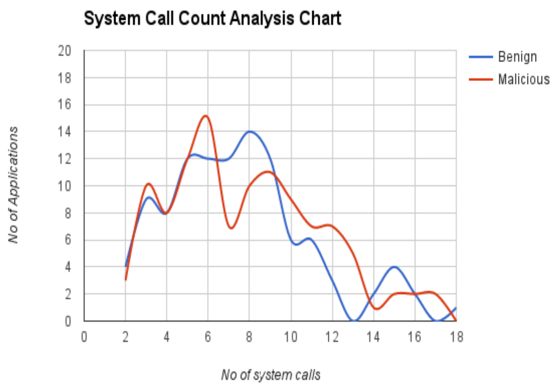

Figure 3: No of system calls used by benign and malicious applications

4.3.1 Classifiers used

We will use following four machine learning classifiers for supervised machine learning:

i. SimpleLogistic

ii. NaiveBayes

iii. SMO

vi. RandomTree

Firstly, we will validate all the permissions that were extracted by static analysis and will check accuracy with each classifier, then all system calls will be validated and their accuracy is checked individually, then as a last step we will combine both of these features and will compare all results.

k-fold cross validation: To check the performance of all the classifiers, k-fold cross validation is used and we took k =10. So, whole dataset is divided 10 times into 10 different learning sets which means 90% of total dataset will be used for training and 10% of total data will be used for testing. With each classifier mentioned in the above table we tested k-fold cross validation with k =10.

4.4 Results Obtained

Once all data is given as input to WEKA, we will use all classifiers one by one to check results.

4.4.1 Validating data obtained from static analysis

First of all, dataset of all the permissions obtained by static analysis is made. Two attributes are taken to make dataset of permissions, those attributes are:

i. Number of permissions that are asked by each application.

ii. Permission's name.

Results that are obtained using all the above classifiers are shown in Table 1.

4.4.2 Validating Data Obtained from Dynamic Analysis: Firstly, dataset of all the system calls which are obtained through dynamic analysis is made. Two attributes that are used to make dataset of system calls are:

i. System call name.

ii. Number of system calls used by each file.

Results that are obtained by using all the above classifiers are shown in Table 2.

Table 1: Comparing results of all classifiers for static analysis

Results / classifiers	Accuracy	True Positive Rate	False Positive Rate
SimpleLogistic	68.94%	0.68	0.29
NaiveBayes	59.81%	0.59	0.41
SMO	69.86%	0.69	0.28
RandomTree	54.79%	0.54	0.47

Table 2: Comparing results of all classifiers for dynamic analysis

Results / Classifier	Accuracy	True Positive Rate	False Positive Rate
SimpleLogistic	57.07%	0.57	0.44
NaiveBayes	60.27%	0.60	0.39
SMO	60.73%	0.60	0.40
RandomTree	62.10%	0.62	0.39

In the above steps, we have analysed permissions and system calls separately on machine learning classifiers, now we will combine both of these parameters and will validate them in a single run on above mentioned classifiers. Table 3 will combine all the results which we obtained after validating permissions and system calls solely and then after combining them.

After comparing all the results, we obtained maximum accuracy when we combined both parameters that is system calls and permissions except in case of SimpleLogistic classifier where maximum accuracy is 68.94% in case of static analysis when we validated permissions solely.

Table 3: Comparing all results

Results / Classifiers Accuracy	Static Analysis	Dynamic Analysis	Combined results
SimpleLogistic	68.94%	57.07%	65.29%
NaiveBayes	59.81%	60.27%	65.29%
SMO	69.86%	60.73%	70.31%
RandomTree	54.79%	62.10%	54.79%

5. CONCLUSION

We have come across various malware detection techniques that uses either static detection techniques that can be easily obfuscated or those that use only dynamic detection techniques, which are also not a complete solution and then made this model which combine features of both static analysis and dynamic analysis and machine learning algorithm. All these techniques are combined so to obtain maximum accuracy in detecting malicious samples.

6. FUTURE WORK

Malware detection techniques should be improved as polymorphic malwares are increasing day by day. As a future work for this model of malware detection, some more dynamic features can be extracted to improve correctness of the system. More samples will be taken and tested so to improve the accuracy of given model. Instead of machine learning classifier various techniques of data mining and artificial intelligence can be used to differentiate between benign and malicious applications. This approach can also be made light weight, so that it can execute on real smartphone and use minimum resources.

7. REFERENCES

[1] IDC data. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[2] Cabir, Smartphone Malware. Available: http://www.f-secure.com/v-descs/cabir.shtml

[3] Google PlayStore. Available: https://play.google.com/store

[4] Malware Repository, http://contagiominidump.blogspot.com

[5] Thomas Zefferer, Peter Teufl, David Derler, Klaus Potzmader Alexander Oprisnik, Hubert Gasparitz and Andrea Hoeller "Power Consumption-based Application Classification and malware Detection on Android Using Machine-Learning Techniques" in FUTURE COMPUTING 2013

[6] Hahnsang Kim, Joshua Smith, Kang G. Shin "Detecting energy greedy anomalies and mobile malware variants" in MobiSys'08

[7] Aubrey-Derrick Schmidt, Rainer Bye, Hans-Gunther Schmidt, Jan Clausen, Osman Kirazy, Kamer Ali Y¨uksely, Seyit Ahmet Camtepe, and Sahin Albayrak "Static analysis of executables for collaborative malware detection on android" in Communications, 2009. ICC '09. IEEE International Conference

[8] Leonid Batyuk, Markus Herpich, Seyit Ahmet Camtepe, Karsten Raddatz, Aubrey-Derrick Schmidt, and Sahin Albayrak "Using Static Analysis for Automatic Assessment and Mitigation of Unwanted and Malicious Activities Within Android Applications" in Malicious and Unwanted Software (MALWARE), 2011 6th International Conference

[9] Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo Garcia Bringas, Gonzalo Álvarez. "PUMA: Permission Usage to Detect Malware in Android", in International Joint Conference CISIS'12-ICEUTE´12-SOCO´12 Special Sessions.

[10] Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones" in: Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI) (Oct 2010).

[11] Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). "Crowdroid: Behavior-based malware detection system for Android" in 2011 ACM CCS Workshops on Security and Privacy in Smartphones and Mobile Devices (SPSM'11), 17-21 October 2011, Chicago, Illinois, USA.

[12] Grace, M., Zhou, Y., Zhang, Q., Zou, S., & Jiang, X. (2012). "RiskRanker: scalable and accurate zero-day Android malware detection." in The 10th International Conference on Mobile Systems, Applications, and Services (MobiSys'12), Low Wood Bay, Lake District, United Kingdom

[13] Portokalidis, G., Homburg, P., Anagnostakis, K., and Bos, H.: "Paranoid Android: Versatile protection for smartphones" in ACSAC'10, Dec. 2010.

[14] Su, X., Chuah, M., Tan, G."Smartphone dual defense protection framework: Detecting malicious applications in android markets" in: Mobile Ad-hoc and Sensor Networks (MSN), 2012 Eighth International Conference on, pp. 153-160 (2012).

[15] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[16] Strace, Available: http://en.wikipedia.org/wiki/Strace