ISSN: 0974-2115
www.jchps.com Journal of Chemical and Pharmaceutical Sciences

Security testing for android application by malware detection on cloud computing

Krishnaveni S [1] Kamala Sevugan[2] Aishwarya Sivakumar[2]
Assistant Professor, Software Engineering, SRM University, Chennai
B.Tech Student, Software Engineering, SRM University, Chennai
***Corresponding author: E-Mail: krishnaveni.s@ktr.srmuniv.ac.in**

ABSTRACT

Out of all the mobile devices available in the market Android is the most popular and maximum sold mobile phone. There is an increasing threat of malware affected applications on Android based mobile device. Some antimalware products available in market has the ability to check malware in its original form. These antimalware products use sandboxing technique to find out malware which can easily be escaped by the hacker's code. Hence, the malware detection method based on permission feature of application can lead to identification of many undetected malwares. We propose a Security Testing for Android apps by malware detection technique, called Permission-based Malware Detection Systems (PMDS). PMDS is implemented as a cloud system that uses a machine learning classifier to automatically identify dangerous behavior of applications based on the combination of permissions they require. We focus on machine learning approach to detect previously unknown and next-generation malware. By using permissions and suspicious API calls some features which would be analyzed and detect malware applications with more accuracy. The application would be classified as good ware or malware accurately. First the permissions are extracted from the manifest file from the apk of an application and the API calls are extracted from code. Weights are assigned to permissions and API calls based on their malicious nature. If it exceeds a predefined threshold, then the application is put down as malware.

KEY WORDS: Malware, Android, Permission, Play Store.

1. INTRODUCTION

In the last few years Smart phones have become very popular. They have the power to do anything that a personal computer can do like they allow users to browse the Internet, send e-mails or connect to a remote desktop. These small computers have the ability to perform complex computing tasks and communicate through different methods (e.g, GPRS, WiFi or Bluetooth). In the last decade, smart phone users have been experiencing problems when installing mobile applications. Earlier, users had to download an application from a website and then, install it on the device. In order to protect the device and avoid piracy, several operating systems, such a Symbian, employ an authentication system based on certificates that causes several inconveniences for the users (e.g, they cannot install applications despite having bought them). Nowadays, due to the enhancements in Internet Connection, there are new methods to distribute applications. Hence, smart phone users can install any application without even connecting the mobile device to the computer. The only requirement to download and install an application is to to have an account of an application store. Apple's App Store implement this new model first and was very successful. Other mobile device manufacturers such as Google, RIM and Microsoft have followed the same technique developing application stores accessible from the device. These facts have increased the number of developers for mobile platforms and the number of mobile applications. According to Apple1, the number of applications available on the App Store is over 350,000, whilst Android Market2 has over 200,000 applications. In Apple devices the App Store is the single official way to obtain applications, whereas Android allows its users to install applications that have been already downloaded from alternative markets or directly from Internet or through side loading. According to their feedback to the US Federal Communication Commission's July 2009, Apple applies a rigorous process to review applications consisting of at least two reviewers. In contrast, Android relies on its security permission system. Unfortunately, users don't have any security consciousness and they do not tend to read required permissions before installing an application in their Smartphone. In spite of both App Store and Android Market including clauses in the terms of services that urge developers to not submit malicious software, both have hosted malware in their respective stores. To solve this problem, they have developed tools for removing these malicious applications remotely. Both models are insufficient to ensure user's safety and new models should have been included in order to improve the security of the devices. Machine learning techniques have been widely applied for classifying applications which is mainly focused on generic malware detection (Burguera, 2011; Shabtai, 2011; Holmes, 1994; Christodorescu, 2005). Besides, several approaches (Yu Feng, 2014; Hanumantha Rao, 2011), have been recommended to classify applications specifying the malware class; e.g, trojan, virus, worms; and, even the malware family. With regards to Android applications, there is a lack of samples for malware. Anyway, the number of samples is increasing continuously.

Background: Android mobile platform is becoming more popular hence the maximum number of Smartphone users is Android users. Android is estimated to have 675,000 apps in the official Google's Android Market and downloads

in excess of 25 billion. Malware developers have discovered many techniques to escape the Antivirus scanning. A recent study reveals 79.6 % best case to 20.2 % worst case detection from the cumulative growth of collected Android mobile malware samples. Major outbreaks AnserverBot (Starting Sept. 2011) and DroidKungFu (starting June 2011) are highlighted. Both are actively evolving to evade detection from existing anti-virus software. A cumulative study of the 1260 Android malware samples from 49 of the 52 then discovered families revealed 86% are legitimate apps repackaged with malicious payload > 92% turned compromised phones into network or SMS controlled bots 36.7% leveraged root-exploits 45.3% enable stealthy premium-rate SMS sending or phone calls 51.1% harvest user information including user account and stored messages. Android malware remain undetected for up to three months. Google's Bouncer introduced in Feb. 2011 uses dynamic analysis and can be prevented. Several third party Android Marketplaces exist, most without app scrutiny Mobile Antivirus engine detection rates are very less and Ineffective against unknown malware. The above motivates our machine learning based approach that uses static analysis. The main advantages are Malware cannot modify its behavior during analysis proactive and flexible.

Literature Survey: Mihai Christodorescu, Somesh Jha (Christodorescu, 2005), presented Semantic aware malware detection. It is based on semantic characteristics of the program which are data flow and control flow. The binary program is disassembled first. Control flow graphs are created for each and every function of program, and then IR (intermediate representation) is created using instruction transformer. Templates are created by oneself for comparison with malicious program. The instructions in template are mapped with the instructions in program based on a certain rules. They both match if the variables of template node match with the program node expression. It checks that the template variables should have similar update pattern as of program expression, although they may not have the same values. Advantages are that it can handle different transformations/obfuscations like reordering of code or instructions. Register renaming can also be detected since it is doesn't check the variable names in template instructions but it is based on matching function. Garbage insertion can be detected. It has low value of false positives denoting that the tendency of detecting the good ware program as malware is very less. The disadvantages are it needs all the IR instructions in template to be of same form as that in program. For example, if the template has an operation like multiplication operation in the instruction and program has equivalent left shift operation, then they are not matched. It enforce that the arranging of memory updates of program should be same as that of template. Yu Feng, Saswat Anand presented Apposcopy: Semantics-Based Detection of Android Malware through static Analysis. In first step it constructs ICCG (Inter component call graph) of application which shows different components, broadcast receivers, activities and services of application. In the next step, it does static taint analysis that is it tracks the flow of data from source to sink. Advantages are High accuracy it can detect obfuscations like Change of component, method, field names and Method invocations to the classes in android are redirected using proxy methods. Disadvantages are cannot detect unknown malware family. Cannot detect obfuscations it cannot detect malware instantly. Daniel Arp, Michael Spreitzenbarth (Arp, 2014), gave a method DREBIN which is an effective and explainable detection of Android malware. This method enables identifying malware on smart phones itself. It combines static analysis of all the features of application and then apply machine learning to classify the applications as malware or benign. It also provides explanation for malware detection. Advantages are it gives only few false alarms. Since the accuracy of method is high, it is good for detecting unknown malware. Disadvantages are cannot detect transformations that are not detectable by static analysis. For example, the Reflection and byte code encryption.

Problem Definition: To classify an application as malware or goodware on Android mobile. In Permission based Android Malware Detection technique given by Win Zaw, Zarni Aung, it does static analysis based only on permission request features of application. Based on permissions requested by any application on the android device, it is categorized as malware or good ware application by machine learning technique. This technique leads to detection of many undetected malwares. Also number of false positives is high.

Mathematical Modeling and Algorithm

Let S be the set {I, O, P, F, S} where,
O: Output set
I : Input set
P: Process set
F: Failure cases
S: Successful cases
1) Output set O = Application predicted as benign or malware.
2) Input set I = Set of Android applications.
3) Process set: P = {P1, P2, P3, P4, P5}
P1=Extraction of Application features from AndroidManifest.xml:

Extract permissions and filters from Android manifest.xml. The extraction has to be done in a timely manner and in constrained environment and represent the features as sets of strings. Figure 1 illustrates the extraction of Application features.
P2 = Extraction of features from disassembled code.
Extract API calls from disassembled code.
P3: Assign weight to permissions, intent filters and API calls based on their malicious behavior.
P4: Fix a threshold by analyzing different malware and good ware applications features with help of Virus total Service.
P5: Add weight of all the permissions, API calls and malicious intents of application to be tested. If the weight exceeds the threshold, then it is classified as malware and it is recommended to check the permissions requested by the app while installing. Check the necessity of requested permissions against the app requirement. If app doesn't need some particular access and still is requesting for it, then it is suspicious app.
Set S: Success cases
The system is able to detect malware correctly.NP Complete: The problem of reliably identifying a bounded-length virus is NP-complete, since the malware application can be identified and detected in fixed and short time.
Set F: Failure cases
The system incorrectly classifies a benign app as malware or a malware as benign app.
System Design and Implementation: The Permission-based Malware Detection System (PMDS) design. The clients, whether Android devices or Android marketplaces, has a Permission Checker that extracts permissions by an application and sends them to a server-side application where the application is evaluated as malicious or benign. PMDS is implemented as a cloud system. Such systems have several additional advantages. One such important advantage is that the cloud service is no longer bound to the Android architecture design and its limitations, allowing a more detailed analysis that are poised to improve detection rates. As mentioned above, the next main advantage is that fewer resources are spent on the device itself, which in turn will improve the battery lifetime in comparison to an on-device scan. As shown in Fig 6. The Permission-based Malware Detection System (PMDS) is poised of a client-side application, whose task is to extract the permissions of an application and send them to the server-side component. The server is the most important component of our technology, as it is tasked with organizing the behavior of the given application instance as either benign or malicious and, then, signaling the results to the client-side application as fast as possible. In order to do so, we will rely on machine learning classifiers.

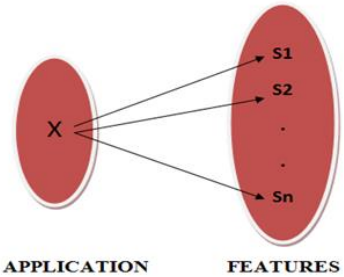

Figure.1.Mapping of application to features

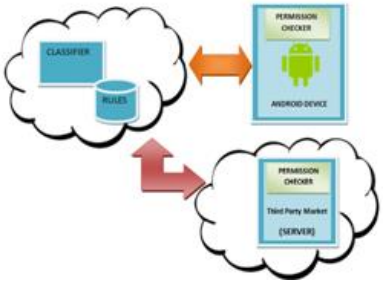

Figure.6.Cloud Architecture for Permission-based Malware Detection System

System Implementation
Server-side Application: The server-side application is the core of our Permission based Malware Detection System (PMDS) as it is the component responsible for classifying an application's behavior as either benign or malicious. In our prototype of the server-side application, we use Python to automatically extract the permissions declared by an application in its Android Manifest. xml file of the APK package .We piggyback on the Android Asset Packaging Tool (aapt) which is a part of the Android SDK. After extracting the permissions, the program automatically save the information in the Weka's Attribute-Relation File Format (ARFF).Our prototype then uses Weka (Holmes, 1994) to train multiple classifiers in order to detect new and unseen malware. To create the classifier's dataset, the permissions declared in the Android Manifest.xml file of an application is automatically extracted using the Android Asset Packaging Tool also known as "aapt". Then, the classifier automatically labels the application behavior, as either benevolent or potentially malicious, according to the combination of permissions the application requires. The permissions requested by an application is represented as a binary vector of Boolean values, one for each permission, where TRUE stays for the presence of that particular permission while FALSE stays for its absence. In our database, we also store the application behavior, saved as either benign or malicious.
Client-side Application: The client-side component is responsible for extracting the permissions declared by an application and sends them to the server-side application. Retrieving the list of applications currently installed on an

Android device, irrespective of whether they are user or system applications can be achieved by Package Manager Class.

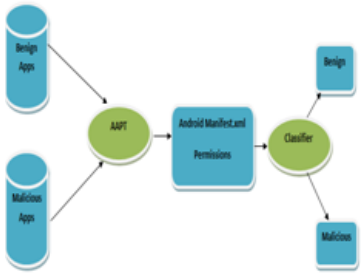

Figure.7.PMDS is poised of a client-side application

EXPERIMENTS

We tested our system against a collection of sample Android applications. Comparing our system result with Permission based malware detection applying J48 classifier (Burguera, 2011), from the response of classifiers, relevant confusion matrices were developed. The following four metrics define the members of the matrix.

- **True Positive (TP**): Number of correctly identified goodware applications.
- **True Negative (TN**): Number of correctly identified malware applications.
- **False Positive (FP):** Number of wrongly identified malware applications.
- **False Negative (FN**): Number of wrongly identified goodware applications

Following are the parameters of evaluation:

True Positive Ratio (TPR):

TPR =TP/TP + FN

TP is the number of malware cases correctly classified. FN is the number of malware cases misclassified as legitimate software.

False Positive Ratio (FPR):

FPR = FP/FP + TN

FP is the number of benign apps incorrectly detected as malware.TN is the number of legitimate apps correctly classified.

Accuracy: The complete number of the classifier's hits divided by the number of instances in the whole dataset.

Accuracy = (TP + TN) *(TP + TN) / (TP + FP + TP + TN)

EXPERIMENTAL RESULTS

First, we extract the necessary features to analyze from sample applications (goodware and malware). Then, we built a dataset in (.arff) file format from the extracted features. We use these two datasets to distinguish between malware and good ware applications by using machine learning approaches.

Following are the numerical results:

Table.1.The TPR and Accuracy of our system are higher than that of permission based malware detection. The FPR is lower than Permission based malware detection

Detection Method	TPR	FPR	Accuracy
Permission based detection	0.88	0.24	**81.25%**
Detection based App features (permission+API calls)	0.89	0.21	**84.2%**

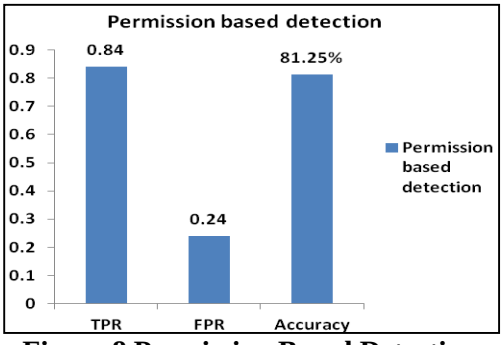

Figure.8.Permission Based Detection

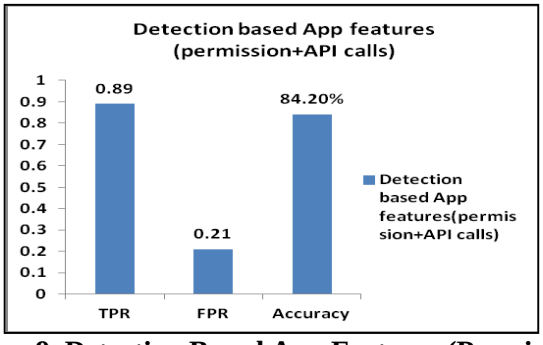

Figure.9. Detection Based App Features (Permission +Api Calls)

CONCLUSION

In this paper, Our proposed a new method of detection of malware on Android based on application features which make use of Permissions, API calls, and Intent filters is proven to be better than only permission based malware detection. It yields better TPR and accuracy than Permission based malware detection. Most of the anti-malwares in industry are signature based which maintains large database of signatures of different malwares and keep on updating the signature regularly. In comparison to this, our method is behavior based which overcomes these disadvantages and detects an application as malware or good ware based on its features. It also gives an indication to check the permissions requested by application while installation that whether they are really required or not.

REFERENCES

Arp D, Spreitzenbarth M, ubner M.H, Gascon H, Rieck K and Siemens C, Drebin, Effective and explainable detection of android malware in your pocket, in NDSS, Internet Society, 2014.

Burguera U.Z, Nadijm-Tehrani S, Crowdroid, Behavior- Based Malware Detection System for Android, In: SPSM'11, ACM, 2011.

Christodorescu M, Jha S, Seshia S, Song D and Bryant R, Semantics-aware malware detection, in Proc. IEEE Symp.Security Privacy, 2005, 3246.

Hanumantha Rao K, Srinivas G, Damodhar A, Vikas Krishna M, Implementation of Anomaly Detection Technique Using Machine Learning Algorithms, Internatinal Journal of Computer Science and Telecommunications, 2(3), 2011.

Holmes G, Donkin A and Witten I.H, Weka, a machine learning workbench, 1994, 357-361.

Holmes, Geoffrey, Donkin, Andrew, Witten, Ian H, Weka, A machine learning workbench, Proceedings of the Second Australian and New Zealand Conference on Intelligent Information Systems, 1994, 357–361.

Shabtai U, Kanonov Y, Elovici C, Glezer Y, Weiss, Andromaly, a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems, 38(1), 2011, 161-190.

Venkatraman M and Devaradjane G, Computer Modeling of a CI Engine for Optimization of Operating Parameters Such as Compression Ratio, Injection Timing and Injection Pressure for Better Performance and Emission Using Diesel-Diesel Biodiesel Blends, American Journal of Applied Sciences-Published by Science Publications New-York USA (Scopus Indexed), 8(9), 2011, 897-902.

Yu Feng, Saswat Anand, Isil Dillig, Alex Aiken, Apposcopy:Semantics-Based Detection of Android Malware through Static Analysis, in ACM SIGSOFT Int, Symp, 2014.