ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 4, April 2016

Malware Detection in Android based on Rules and Risk factor of Permissions

Nishali Deshmukh, Bhagyashri Deshatwar, Nivedita Kadam, Aishwarya Patil

***Abstract*—** Mobile malware are growing in scale and complexity as smartphone usage continues to rise. This malware have many side effects on android system.In recent years a number of approaches for Android malware detection have been proposed, using permissions, source code analysis, or dynamic analysis. There has been continuous development in malware detection systems. This paper is one such approach aimed at development of malware detection system and determined to develop a Android Malware detection system. It includes creation of android application for malware detection. The system classifies application as malicious and non-malicious on basis of parameters like package name, valid digital certificate and risk factor of permissions requested by the application. The android application with permissions having high risk factor are malicious. The results of classification are then conveyed to the users . This system is simple, efficient to use and easily available. It reduces threat of malware and thus provide security to Android system.

Keywords- malware, Android, permissions.

I. INTRODUCTION

There has been a steady rise in android mobile devices these days because of their user friendly nature. It has become a basic necessity of human other than food, water and shelter. Android platform is the most popular and famous OS among smart phone users. These devices run various applications which are used to perform various tasks .For example application SMS is used to send messages, social networking applications like whatsapp, hike, instagram are used for interactive conversation and information transfer, etc. In order to provide the respective services these applications need to access the some user information like their phone contacts, their images, media storage, etc. This facility puts the mobile user's data to risk as there is a fear that this data may be misused for unethical purpose. It means that information from the device can be used to perform malicious activity. Malicious software i.e. malware in an application may steal confidential data of user and upload it on its server which is threat to users security, it may redirect the web browser to unwanted sites, it may pass faux messages to all the contacts of user's device. The malicious application through their permissions may have access to personal information like bank account details, passwords, etc. This information is extremely sensitive and could prove danger if used for ill motives. Hence, there is a need for malware detection to avoid all these above problems. There are various malware detection techniques present which are used to solve this problem.

Manuscript received April, 2016.

***Nishali Deshmukh**,Information Technology, Bharati Vidyapeeth's College of Engineering for Women, Pune, India, 8806678577.*
***Bhagyashri Deshatwar**,Information Technology, Bharati Vidyapeeth's College of Engineering for Women, Pune, India, 9922657395.*
***Nivedita Kadam**,Information Technology, Bharati Vidyaeeth's College of Engineering for Women, Pune, India,9730702199.*
***Aishwarya Patil**, Information Technology, Bharati Vidyaeeth's College of Engineering for Women, Pune, India,8600453615.*

II. RELATED WORK

Malware detection-android Permissions:A perspective combining Risks and Benefits. This methodology is used to find out feasibility of using both the permissions of an app requests, the category of the app and what permissions are requested by other android apps in the same category to inform users whether the risks of installing an app is propotional with its expected benefit. It proposes several risk signals also which evaluate them using two datasets, one consists of 158,062 Android apps which come from android market, and other contains of 121 malicious apps.[1]

Using probabilistic generative models for ranking risks of Android apps: In this paper, authors have introduced the notion of risk scoring and risk ranking for Android apps, to improve risk communication for Android apps, and identify three desiderata for an effective risk scoring scheme. They have proposed to use probabilistic generative models for risk scoring schemes, and identify several such models, ranging from the simple Naive Bayes, to advanced hierarchical mixture models. Experimental results conducted using real-world datasets show that probabilistic general models significantly outperform existing approaches, and that Naive Bayes models give a promising risk scoring approach.[2]

RiskRanker, a Scalable and Accurate Zero-day Android Malware Detection: In this paper, authors have proposed a proactive scheme to spot zero-day Android malware. Without relying on malware samples and their signatures, their scheme is motivated to assess potential security risks posed by these untrusted apps. Specifically, they have developed an automated system called RiskRanker to scalably analyze whether a particular app exhibits dangerous behavior (e.g.,launching a root exploit or sending

background SMS messages). The output is then used to produce a prioritized list of reduced apps that merit further investigation.[7]

Crowdroid:Behavior-based malware detection system for Android- The next research is on crowdroid: Behavior-based malware detection system for android. This framework has been demonstrated by analyzing the data collected in the central server using two types of data sets: Those from artificial malware are created for test purposes, and those from real malware found in the wild. The method is shown to be an elective means of isolating the malware and alerting the users of a downloaded malware. This shows the potential for avoiding the spreading of a detected malware to a larger community.[5]

III.METHODOLOGY

It is our goal to perform classification of an application as malicious or nonmalicious, given its .apk file which includes the code of application and list of permissions that application requests. In this section, we describe how to extract the information from .apk file necessary for classification.

The following information need to be extracted from .apk file-

- package name
- certificate information
- permissions requested

A. Information Extraction-

The information extraction needs the .apk file to be decompiled. As we now know that .apk file is just a zip file containing all program resource file, we can now get java code from .apk files with ease. Following are the steps to decompile .apk file.

Step 1:

Make a new folder and put .apk file in it (which you want to decode). Now rename the extension of this .apk file to .zip (e.g.: rename from filename.apk to filename.apk.zip) and save it. Now you get classes.dex files, etc. At this stage you are able to see drawables, but not xml and java files, so continue to step 2.

Step 2:

Now extract this zip apk file in the same folder (or NEW FOLDER). Now download dex2jar from this link" http://code.google.com/p/dex2jar/" and extract it to the same folder (or NEW FOLDER). Now open command prompt and change directory to that folder (or NEW FOLDER). Then write dex2jarclasses.dex and press enter. Now you get classes.dex.dex2jar file in the same folder. Then download java decompiler from link "http://java.decompiler.free.fr/?q=jdgui" and now double click on jd-gui and click on open file. Then open classes.dex.dex2jar file from that folder. Now you get class files and save all these class files (click on file then click "save all sources" in jd-gui) by source name. At this stage you get java source but the xml files are still unreadable, so continue to step 3.

Step 3:

Now open another new folder and put these files and put .apk file which you want to decode then download apktool v1.x AND apktool install window(both can be downloaded at the same location) and put in the same folder. Download framework-res.apk file and put in the same folder (Not all apk file need framework-res.apk file).Open a command window and navigate to the root directory of APKtool and type the following command: apktool if framework-res.apk. Then write the command apktool d "fname".apk ("fname" denotes filename which you want to decode).Now you get a file folder in that folder and now you can easily read xml files also.

Step 4:

Now just copy contents of both folder(in this case both new folder)to the single one and we obtain the complete source code.

The information about package name, certificate information and the permissions requested by the application can be obtained through the files extracted from the .apk file by above means.

B.Scan of apk file

The scanning of an .apk file begins with scan of following in a prioritized order-

- Package name scan
- Certificate scan
- Permissions scan

Package name scan

In this scan, the package name of the application to be checked with the database where the package names of malicious applications is already stored. Since the package name of an application is unique, if the name matches with the name from database, then application would be considered malicious else nonmalicious.

Certificate Scan

In certificate Scan, the presence of digital certificate in the .apk file of an application is checked.It is mandatory for a valid application to have a digital certificate. If the valid certificate is not present, then the application is considered to be malicious. Each valid android application's .apk file is verified using a digital certificate. Here we need to check the presence of x.509 standard in .apk file. An X.509 certificate binds a name to a public key value. The role of the certificate is to associate a public key with the identity contained in the X.509 certificate. Authentication of a secure application depends on the integrity of the public key value in the application's certificate. If an impostor replaces the public key with its own public key, it can impersonate the true application and gain access to secure data. To prevent this type of attack, all certificates must be signed by a certification authority (CA). A CA is a trusted node that confirms the integrity of the public key value in a certificate. A CA signs a certificate by adding its digital signature to the certificate. A digital signature is a message encoded with the CA's private key. The CA's public key is made available to applications by distributing a certificate for the CA. Applications verify that certificates are validly signed by decoding the CA's digital signature with the CA's public key. Thus, the certification of .apk using CA is necessary for validation of application

C. Permissions scan

Each android application requests permissions of the android device. Sometimes, it requests for the unnecessary permissions, which are not needed to be used by the application. These permissions could be misused to harm the user. Hence, there is a need to perform permission scan. There are two ways in which permission scan could be performed-

Risk factor indication

Some important android permissions are given a risk factor value and this risk factor value is based on the risk of permission to intervene with and access the confidential information of the mobile phone Total Risk factor is sum of risk factor value of all permissions requested by the application. The applications which have a value more than a threshold risk factor are considered to be malicious while applications with value less than threshold, to be nonmalicious. Some of the risky android permissions are given below-

Device & app history permission

An app can do one or more of the following with this permission:

- Read sensitive log data
- Retrieve system internal state
- Read user's web bookmarks and history
- Retrieve running apps

SMS permission

An app can use user's device's text messaging (SMS) and/or multimedia messaging service (MMS). An app can do the following with this permission-

- Receive text messages (SMS)
- Read text messages (SMS or MMS)
- Edit text messages (SMS or MMS)
- Send SMS messages; this may cost user money

Phone permission

An app can use user's phone and/or its call history. Depending on the plan, user may be charged by their carrier for phone calls. Phone permission gives access to following:

- Directly call phone numbers; this may cost user some money
- Write call log (example: call history)
- Read call log
- Reroute outgoing calls
- Modify phone state
- Make calls without user's intervention

Location permission

An app can use the user's device's location. Location access may include:

- Approximate location (network-based)
- Precise location (GPS and network-based)
- Access extra location provider commands
- GPS access

Media permission

An app can use files or data stored on the device. Photos/Media/Files access may include the ability to:

- Read the contents of USB storage (example: SD card)
- Modify or delete the contents of USB storage
- Format external storage
- Mount or unmount external storage

Bluetooth permission

An app can control Bluetooth on user's device, which includes broadcasting to or getting information about nearby Bluetooth devices.

Rule generation

This facility helps provide user intervention in the application. A user can create a rule which involves selection of some permission from a list of most requested android permissions. This rule concept helps perform quick scan. During the scan, if atleast one of the permissions requested by the application is present in the rule, then that application is considered as malicious else nonmalicious. This feature of rule generation helps user to forbid the permissions from gaining access to sensitive or protected data from mobile phone and thus helps maintain the confidentiality of information.

IV.CONCLUSION

There has been a tremendous growth market of Android devices and thus increase in threat to android mobile devices due to malicious applications. These malicious applications thus could invade the privacy of user or cause some intentional/unintentional harm. Hence, malware detection in android is important .This paper proposes to scan or detect malware by performing package name scan, certificate scan and permission scan using risk factor and rules.

V.REFERENCES

[1] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "Android permissions: A perspective combining risks and benefits," in Proc. 17th ACM Symp. Access Control Models Technol., 2012, pp. 13–22.

[2] H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, C. NitaRotaru, and I. Molloy, "Using probabilistic generative models for ranking risks of Android apps," in Proc. ACM Conf. Comput. Commun.Security, 2012, pp. 241–252.

[3] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," J. Mach. Learn. Res., vol. 7, pp. 2721–2744, Dec. 2006.

[4] A.-D. Schmidt, R. Bye, H.-G. Schmidt, J. Clausen, O. Kiraz, K.A.Y€uksel, S. A. Camtepe, and S. Albayrak, "Static analysis of executables for collaborative malware detection on Android," in Proc. IEEE Int. Conf. Commun., 2009, pp. 1–5.

[5] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid:Behavior-based malware detection system for Android," in Proc. 1st ACM Workshop Security Privacy Smartphones Mobile Devices,2011, pp. 15–26

[6]A. P. Felt, K. Greenwood, and D. Wagner, "The effectiveness of application permissions," inProc. 2nd USENIX Conf. Web Appl. Develop., 2011, p. 7.

[7] Michael Grace, Yajin Zhou, Qiang Zhang, Shihong Zou and Xuxian Jiang," RiskRanker: Scalable and Accurate Zero-day Android Malware Detection", in MobiSys'12, June 25–29, 2012.