Malicious Applications Detection by Analyzing Manifest Files

Aparna Harikumar
BE Computer, VIIT Pune

Chandrika Kapre
BE Computer, VIIT Pune

Rajeshwari Chandratre
BE Computer, VIIT Pune

Dhaval Adhav
BE Computer, VIIT Pune

Mrs.S.R.Rathi
Assistant Prof.VIIT Pune

ABSTRACT

With the rapid rise in the use of Android worldwide, the harm done due to malicious applications is increasing exponentially. Due to this the data of the Android users is not secure, and we need a system to detect the maliciousness of the applications being downloaded. The project being developed here is a solution to this problem as it detects whether an application is malicious or benign, this will not only provide data security but will also enhance the performance of the user's device. There are a number of systems like DroidMat and signature-based systems, already available in the market that detect malicious applications. But these systems detect that on the base of signatures and by parsing smali files which not only increases the processing time but also does not provide accurate results. The proposed system here that is Malicious applications detection using permissions retrieval detects whether an application is malicious or not based on the keywords present in the manifest file. This method does not affect the performance of the device in any way and gives us better throughput.

General Terms

Security,malware,adware,malicious

Keywords

Android,smart phones,manifest file.

1. INTRODUCTION

With the advent of smartphones, hundreds of thousands of Android apps find their way in the Android market. The Android market is an open platform, and hence anyone can upload the apps developed by them here. The cyber criminals use this feature as a golden opportunity to post their malicious applications in the Android market. The malicious applications may prove hazardous on various levels. Right from collecting user’s personal information to sending text messages to some premium rate numbers, the malware possesses a great threat to the Android users. Before their installation, the Android applications ask for certain permissions like access to the installer’s contact list, gallery or GPS co-ordinates and even for the modification of the SD card content. Some of the permissions might prove unnecessary according to the use of the app. Thus, the malicious applications access the user data and may make transactions in the Android device illegally. As Android is considered as an easy target for the attackers, the privacy and integrity of the users is seriously threatened. There have been many successful attempts to get rid of the malicious content from the phone including the Signature-based Detection, Behaviour-based Detection, DroidMat, etc. Detection based on signature is quite effective in detecting the known malicious applications but is susceptible to evasion. Since the signature is known, it can be easily hacked. It fails to detect unknown malicious applications since its signature is not known. The other technique overcomes the shortcomings of the previous method. Simply put, a behaviour-based detector determines whether a program is malicious by inspecting what it does rather than what it says. But again, it is susceptible to mimicry attacks. Also, it is susceptible to false positives as normal behaviour for complex programs is very complicated. Another system was developed to provide a static analysis paradigm for detecting malware, called DroidMat. They obtained some distinguishable characteristics such as permissions, components and API calls by analyzing manifest files and smali files. This system can discriminate between malware and benign applications. However, the cost of their analysis depends on the size and numbers of smali files. The efficiency reduces highly as we have to access the manifest, as well as smali files. The complexity of this algorithm is quite high. Though there are numerous anti-malware applications available in the market, it is the need of the hour to create a better and an effective way to detect the hazardous malicious applications you have installed in your phone. The main objective of the DMAP is to increase the security of the Android devices and make the users aware of malicious content carried by their phones. This application will reduce the complexity of malware detection by just fetching and reading the manifest files of the Android applications installed and analysing them for potential malicious content. We also display the permissions and features of the user’s phone accessed by the installed Android applications on the device. The integral resource to implement the DMAP is the manifest file of the Android application to be scanned. The manifest file is a XML file carried by every Android application and carries specific information required by the Android system about that application. The contents being the permissions and other resources accessed by the application from the device on which it is installed. This manifest file is scanned for various malicious tags which will be computed upon to determine the degree of the maliciousness of that app. When the user wishes to scan any app, it should be selected explicitly. The Manifest File of the application to be scanned is fetched. Originally in the XML format, this file is now saved in text format and is parsed by the XML parser. The content of this file is analyzed for various permission tags such as uses-permission, uses-library, uses-features, etc. After supervising the file content for potentially malicious tags, these tag contents are extracted and compared with a set of predefined malicious tags, this decides the value of keywords in the manifest file that are malicious. It is followed by computation of the degree of maliciousness. Our application now displays the obtained figure in percentage- the level to which the scanned app is potentially malicious. Some apps being harmless, we leave it to the user whether to uninstall the app or still use it. This system is successful in extracting and displaying all the resources like permissions required and features accessed by the scanned application from the user's Android device. Which gives a clear idea about the reason of the application's degree of maliciousness to the user. In this paper, Section II describes the literature survey that has been done while bringing this idea into being. Section III gives a detailed description of the proposed methodology and Section IV shows the results of the testing done and its analysis..

2. LITERATURE SURVEY

Trojan like malware on Android is recognized by machine learning based framework known as crowdroid[1].It analyzes a frequency of each system call issued by an application at the time of execution of an action which requires user interaction.It differs from a genuine application. Crowdroid builds a vector of Android system calls features.13 features are used by MADAM: multi-level anomaly detector for Android malware [3]in which features are used at both levels kernel and user level.MADAM has a global monitoring approach which detects malware contained in unknown applications that are not classified previously. MADAM is tested on real malware found in wild. [5] proposes a behavior-based malware detection system (pBMDS) that correlates user's inputs with system calls to detect anomalous activities related to SMS/MMS sending. [6] and[7] propose Kirin security service for Android, which performs lightweight certification of applications to mitigate malware at install time. Kirin certification uses security rules that match undesirable properties in security configuration bundled with applications. [8] performs static analysis on the executables to extract functions calls usage using read elf command. Hence, these calls are compared with malware executables for classification. Finally, [9] surveys some security solutions for mobile devices. In Shadow Manifest[10] permission that an app requires are stored by prior execution of the app. Unnecessary permissions are stored along with a mask which are to be revoked by generating an empty resource when an app requests them. COPES(Correct Permission Set)[11] is a tool which uses static analysis to extract a table from Android framework bytecode. This table the set of permissions that an app needs and maps every method of the API to these permissions which are called. So no unnecessary permissions are stored in the table and mapped with API methods. In Apex[12] users are allowed to specify what an app can access. An extended installer is used to set user policies.

Data set	Highest	Lowest
Malicious	106	1
Market	36	0

Average number of permissions by data set, and the highest and lowest number of requested permissions (source internet)

3. PROPOSED WORK

3.1 Architectural Design

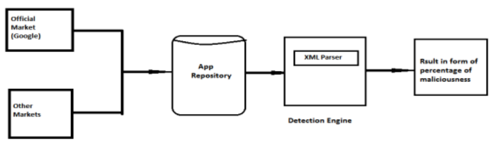


Fig 1 Architectural design

Applications in Android can be downloaded from the official market that is google app store as well as from third party applications providers. All downloaded applications that are all .apk files will get stored on user's device storage let's say this storage space as applications repository.Manifest file for an Android application is a resource file which contains all the details needed by the Android system about the application. It is a key file that works as a bridge between the Android developer and the Android platform. It helps the developer to pass on functionality and requirements of our application to Android. Manifest file is a XML file which must be named as AndroidManifest.XML and placed at application root. Every Android app must have AndroidManifest.XML file. AndroidManifest.XML allows us to define.

3.2 The packages, API, libraries needed for the application

- Basic building blocks of application like activities, services etc.
- Details about permissions.
- Set of classes needed before launch.[source internet]

When a user wants to scan the particular application manifest file will get fetched to detection engine.Here detection engine will start his working.Firstly the XML parser will parse this manifest file and .txt file will get generated for further analysis.Then an underlying algorithm will run, and degree of maliciousness in percentage will be displayed on the users screen.Now its up to the user whether to keep the application or to uninstall it..

3.3 Algorithm

1]Fetch manifest file

Here the user needs to select the application which he wants to scan.As soon as user selects application for scanning manifest file of an application will get fetched

2]Read using XML parser

Manifest files of applications are XML files so parsing is done which will give .txt file so that it can be easily accessed for analysis.

3]Analyze it for potentially malicious tags

In manifest file, various tags are present like

Elements for Application Properties

- uses-permission –specifies the permissions that are requested for the security.
- permission –sets the permissions to provide access control for specific component of the application.
- permission-group – does the same as permission for a set of the components.
- permission-tree – refers the one specific name of the component which is the parent of the set of component.
- instrumentation – shows interaction between Android system and application.
- uses-sdk – specifies the platform compatibility of the application.
- uses-configuration – gives information about set of hardware and software required for the application.
- uses-feature – specifies single hardware and software requirement and their related entity.

• supports-screens, compatible-screens – both these tags deals with screen configuration mode and size of the screen etc.

• supports-gl-texture – specifies texture based on which the application is filtered.

Elements for Application Components

These should be enclosed in <application> container.

• activity – has the set of attributes based on a user interface.

• activity-alias – lets us know about target activities.

• service – has the operation provided by any library or API, running in a background that is not visible.

• receiver – that makes to receive message broadcasted by the same application or by an outside entity.

• provider – provides some structure to access application data.

• uses-library – it specifies a set of library files need to run the application.

For each and every tag values are obtained and analyzed.

4]Comparing with predefined tags of malicious tags

5]Computing the value

6]Displaying the result

Result is displayed in 2 forms

a. degree of maliciousness b. what all permissions application uses

7]Uninstall

If the user wants to uninstall an application through our application, the uninstall choice is given.

4. TESTING AND RESULTS

For our project we tested applications in which there were variety of applications categorically games, chatting applications ,music applications , online shopping applications, food related applications, travelling related applications and so on along with some known malicious applications like Geinimi, DroidDream, CounterClank, Pjapps, asSMS, Jimm Russia, Gold Dream etc. From these sample applications degree of maliciousness is obtained.

Result of testing is as follows

Table 1

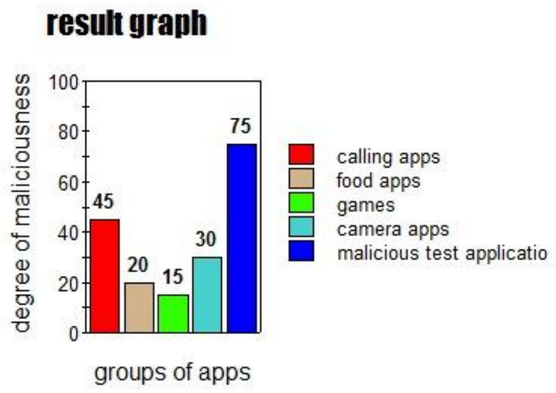
Type	Correct Detection (%)	Incorrect Detection (%)
Malicious	88.7	11.3
Benign	91.6	8.4
Total	90.15	9.85

By our proposed methods some malwares were not detected for example adwares which means while running application we generally find different advertisements, our application fails to capture such advertisement's manifest files or information related to them. As there is often a marginal difference between a benign application and adware. This means that both manifest files appear to be similar, and it is difficult for our proposed method to effectively detect adware based on the manifest analysis.

Table 2 1=Uses permissions 0=Not uses permissions

Permissions	Aptitude workbook	Gallery	Cup cake game	Jabong	Farm heroes game	MalDetect
READ_PHONE_STATE	1	0	0	0	0	0
INTERNET	1	0	1	1	1	0
SEND_SMS	0	0	0	0	0	0
WRITE_EXTERNAL_STORAGE	0	0	0	1	0	1
ACCESS_NETWORK_STATE	1	0	1	1	1	0
RECEIVE_SMS	0	0	0	0	0	0
READ_SMS	0	0	0	0	0	0
ACCESS_WIFI_STATE	1	0	0	1	1	0
WRITE_SMS	0	0	0	0	0	0
READ_CONTACTS	0	0	0	1	0	0
INSTALL_PACKAGES	0	0	0	0	0	0
MODIFY_PHONE_STATE	0	0	0	0	0	0
MOUNT_UNMOUNT_FILESYSTEMS	0	0	0	0	0	0
PROCESS_OUTGOING_CALLS	0	0	0	0	0	0
bluetooth	0	0	0	0	0	0
location.g	0	0	0	0	0	0

ps						
camera	0	1	0	0	0	0
cameara.front	0	0	0	0	0	0
wifi	0	0	0	0	0	0
ACCESS_FINE_LOCATION	0	0	0	0	0	0
ACCESS_CORE_LOCATION	0	0	0	0	0	0
WRITE_CONTACT	0	0	0	0	0	0
multiwindow	0	0	0	0	0	0
BOOT_COMPLETED	0	0	0	0	0	0
SMS_RECEIVED	0	0	0	0	0	0
CONNECTIVITY_CHANGE	0	0	0	0	0	0
USER_PRESENT	0	0	0	0	0	0
PHONE_STATE	0	0	0	1	0	0
NEW_OUTGOING_CALL	0	0	0	0	0	0
UNINSTALL_SHORTCUT	0	0	0	0	0	0
INSTALL_SHORTCUT	0	0	0	0	0	0
Degree of maliciousness in %	12	0	6	22	22	3



Calling apps- Truecaller and apps that access your contact list were scanned and the average result was calculated.

Food Apps- Applications like zomato, burrp, foodpanda, etc were scanned and the average of their degrees of maliciousness was calculated.

Games- Games like candy crush, cricket world, etc were scanned and average was calculated.

Camera apps- all applications that accessed the camera and gallery present on the device were scanned and the average was computed. Eg. Google camera, photo editing apps.

Malicious test applications- Many malicious applications were downloaded and manually made for testing purpose so their average degree of maliciousness was calculated as well.

Basic comparison between all these groups were made and it is depicted using a bar graph. The graph shows the basic difference in the degree of maliciousness.

5. FUTURE SCOPE

The idea presented in the paper devises a method to detect the degree to which an application can access the device resources hence giving the user an idea of the maliciousness level of the application.

This idea can be further expanded to give the user a chance to select the permissions required by the application before downloading it. Which allows the user to secure his or her data and device resources in a proper way. The idea presented in the paper is static, this can be made dynamic as well. Which will ensure that no application on the user's device is hacked by another application. The manifest file of every application is made secure, and the hacker application is not allowed to access any of the manifest files in order to make changes and use device resources.

6. CONCLUSION

This paper has presented an application which can detect a degree of a maliciousness of an application is a cost effective way. Every Android application has a manifest file attached to it, the application discussed in this paper accesses this manifest file and based on the permissions, services, features, etc. present in the manifest file the degree of a maliciousness is calculated. This method is better than the earlier present methods as it requires access to only the manifest file thus reducing the processing time and since it is based on permissions the result is more accurate. Thus the idea presented in the paper gives the user a chance to secure his or her own device, and avoid the breach of security and unnecessary use of device resources.

7. ACKNOWLEDGMENTS

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our highly respected and esteemed guide Prof.S.R.Rathi,Computer department, VIIT Pune, for their valuable guidance, encouragement and help for completing this work. Their useful suggestions for this whole work and co-operative behavior are sincerely acknowledged..

8. REFERENCES

[1] I. Burguera, U.Z., Nadijm-Tehrani, S.: Crowdroid: Behavior- Based Malware Detection System for Android. In: SPSM'11, ACM(October 2011)

[2] A. 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) (January 2011) 161-190

[3] G. Dini, F.Martinelli, A. Saracino, D. Sgandurra: MADAM: a Multi-Level Anomaly Detector for Android Malware

[4] Schmidt, A.D., Peters, F., Lamour, F., Scheel, C., Camtepe, s.A., Albayrak, S.: Monitoring smartphones for anomaly detection. Mob.Netw. Appl. 14(1)(2009) 92-106

[5] Xie,L.,Zhang,X.,Seifert, J.P.,Zhu, S.: pBMDS: a behavior-based malware detection system for cellphone devices. In: Proceedings of the Third ACM Conference on Wireless Network Security, WISEC 2010, Hoboken, New Jersey, USA, March 22-24 2010, ACM(2010) 37-48

[6] Enck, W., Ongtang, M., McDaniel, P.: On lightweight mobile phone application certification. In: CCS '09: Proceedings of the 16th ACM conference on Computer and Communication Security, New York, NY, USA, ACM (2009) 235-245

[7] Ongtang, M., McLaughlin, S., Enck, W., McDaniel, P.: Semantically Rich Application-Centric Security in Android. In: Computer Security Applications Conference, 2009. ACSAC '09. Annual.(Dec 2009) 340-349

[8] Schmidt, A.D., Bye, R., Schmidt, H.G., Clausen, J.H., Kiraz, O., Yuksel, K.A., Camtepe, S.A., Albayrak, S.: Static Analysis of Executables for Collaborative Malware Detection on Android. In: Proceedings of IEEE International Conference on Communications, ICC 2009, Dresden, Germany, 14-18 June 2009, IEEE (2009) 1-5

[9] La Polla, M., Martinelli, F., Sgandurra, D.: A survey on security for mobile devices. Communications Surveys Tutorials, IEEE PP(99) (2012) 1-26.

[10] LakhmiPriyaSekar,Vinitha Reddy Gankidi, SelvakumarSubramanian.,Avoidance of Security Breach through Selective Permissions in Android Operating System.ACM SIGSOFT Software Engineering Notes,September 2012 Volume 37 Number 5.

[11] AlexandreBartel,JacquesKlein,MartinMonperrous.AutomaticallSecuring Permission-Based Software by Reducing the Attack Surface:An Application to Android.

[12] Mohammad Nauman and Sohail Khan. Design and Implementation of a Fine-grained Resource Usage Model for the Android Platform. The International Arab Journal of Information Technology, Vol.8, No.4, Oct 2011