Countermeasures for Security Vulnerability in Android

Ipta Thakur, Guide-Shaily Jain

Abstract*— The high speed penetration of Smartphone's in the market with Android as the leading operating system makes the need for malware analysis on this platform an urgent and concerning issue. In our project we capitalize earlier approaches for dynamic analysis of location based and other suspicious permissions and classes which can cause vulnerability. Our framework has been demonstrated by analyzing the permissions those are vulnerable. Array list will be created on the basis of the permissions and names of classes, and then checked for vulnerabilities using automated approach and then assured through the manual cross checking for vulnerability.

***Index Terms*—Android Security, Malware Analysis, Dynamic Analysis, Vulnerabilities.**

I. INTRODUCTION

With the advancement in technology, the Smartphone market is growing by leaps and bound but there is one O.S that has been enjoying the best of all i.e. Google's Android which contributes 75 % of Smartphone's shipped in Q1 2013 according to IDC stats. The reason for this rise is open source which gives a complete new freedom to developers as well as users. As android is open source so developers can upload their application without any certificate check whereas they can upload self signed applications which can be done without any help or assistance. Smartphone's are becoming ubiquitous and entities ranging from a university student to big organizations rely on them for their personal and private information. All these above reasons make the malware analysis an immense part of today's android security world. In summary we make the following contributions:

A. Point 1

We describe the design and implementation of permission based analysis to detect the vulnerability for the malware attack. Our analysis is able to automatically describe low level o.s specific and high level android –specific behavior of android malware by observing and analyzing system call invocations including IPC and RPC interactions ,carried out as system calls underneath.

B. Point 2

Based on the observation of different permissions and classes which can cause vulnerability, an array list is created which contains all the permissions and names of the classes.

C. Point 3

After the in depth observation, again we create the second level of the array list which contain the list of files which we want to scan ,which we think them to be vulnerable for the attack .We name the tool as patternDroid , We analyzed maximum of 25 malware samples belonging to different Android malware families. PatternDroid is able to automatically and faithfully check the different files for the vulnerability and creates the list of the vulnerable files sets. The same is done manually for the vulnerability check.

D. Point 4

We developed a web interface through .NET framework which is written in c# window based called patternDroid which breaks the applications (.apk) into the number of files stored in the destination folder. Then each of the file is scanned using patternDroid & checks for the permission .Then ratio is calculated, using the formula NO. OF PERMISSIONS MATCHED/NO. OF FILES. The no. of permissions matched are then tested manually in static and dynamic analysis, hence we get the percentage of vulnerability in the malware sample.

II. ANDROID FRAMEWORK

Android o.s is divided into 5 main layers:

A. Linux Kernel

This is the core layer which contains the kernel; this includes all low level device drivers of the hardware included in the android device.

B. Libraries

It contains all the code written in c/c++ that provides the main features for e.g. SQLite for data storage & web libraries to access web browser and other web related services etc.

C. Android Runtime

This provides a set of core libraries by which the developers write android applications using java language. It also includes DVM (i.e. It has its own virtual machine).android applications are compiled into dalvik executables.

D. Application Framework

This contains classes and services which the developers can extend and reuse. In this we are provided with managers which enable application for accessing specific data related with that manager for e.g. telephony manager is for handling network connection, notification manager is for displaying alerts etc

Revised Manuscript Received on June 06, 2013.
Ipta Thakur, CSE, Chitkara University, Delhi, India.
Shaily Jain, CSE, Chitkara University, Baddi, India.

E. Applications

This is the top layer. All applications downloaded and installed from android market and written by developers are located in this.

Source: Android documentation

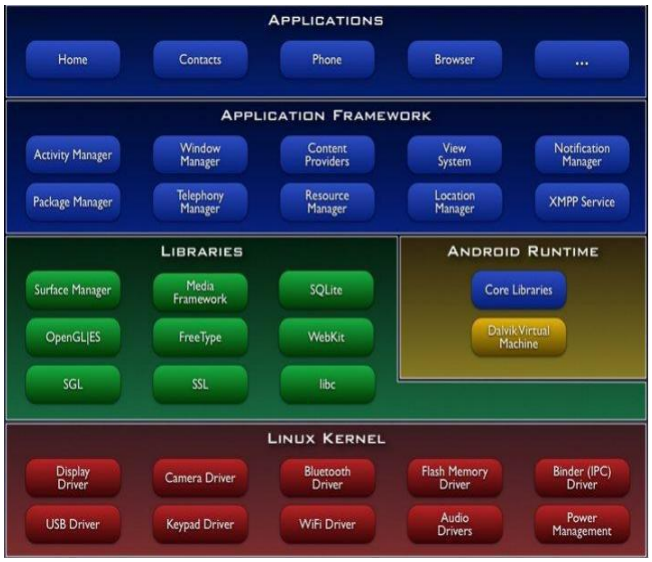

Figure1. Android Framework

III. MALWARES

A. Android Malwares

Malicious software and malicious code which are technically known as malware refers to a software which is deliberately added to the system with the perspective of disrupting or damaging the services , data ,application and o.s. by compromising the confidentiality, integrity and availability of the system.

B. Reversing the android malware[13]

In the forward process java code is compiled using java compiler javac and *.classes are produced which are further processed by DX which converts the *.classes files to *.dex files, but in the reverse process *.dex files disassembled to *.classes files and further to *.java files by decompiler. This is all done to understand the services, code, parameters and files which were added or modified by the malicious software. By reverse engineering we study the working of the malware that how actually the malware is imposing the threat into the system. This is done in two steps:

a. Static analysis

This includes the behavior analysis like for eg. If on the installation of application money is being charged to us for the sms, any email being sent which can be seen by us observing the sent box, any new network connection made, files being tampered, any ports which are opened etc .These observations can lead us to determine serious threats in the application.

b. Dynamic Analysis

This includes the code analysis. We can never get the full 100% code but still most of the code can be get in binary form which can be converted to readable format by the debuggers and decompilers. As there is no standard certification and self signed certificates make a way through them that is why malware coders embedded their malicious code with the original one and then self sign it again. Other than this different packers are used which compress the code which is again done by the malware coders as compressed code is harder to detect.

IV. FUNCTIONALITY OF VULNERABILITY [3]

A. Phone Identifiers being misused

Identifiers such as phone no. , imei no. (Device identifier), imsi (subscriber identifier) & ICC-ID (sim card serial no.) are leaked by the mobile devices to unknown servers.

B. Physical location Exposed

In this the latitudinal, longitudinal parameters, city/country & geographic coordinates of the mobile device can also be leaked.

C. Telephony Services Misused

In this SMS messages to premium rate numbers & voice call abuse are done.

D. Background Audio/video

Smartphone's use the record audio/video facility without the legitimate control .Eavesdropping Problem On microphones and camera is also a part of this.

E. Use of socket API

Java sockets give invitation to malicious entities as they act as an open interface to external services. So applications that use socket class for network connection are at risk.

F. Android specific vulnerabilities [3]

a. Leaking Information to logs

Log messages can be read if the application has the permission READ-LOGS. And this can prove to be harmful if the log has sensitive and private information.

b. Leaking information via IPC

If the target component of an application is not specified it can receive intent broadcast from any other application which can be malicious in nature.

c. Unprotected Broadcast Receivers

Broadcast receivers are been used by the applications to receive intent messages. These receivers define some filters or permission types. If receivers are not protected a malicious application can forge messages.

d. Intent Injection attacks

Start of a service and start of an activity are done with the help of intent messages, an IIA occurs if intend address is derived from untrusted input.

e. Delegating control

Applications can use a "pending intent" and can transfer actions to others applications. The pending recipient applications cannot change the values, but can still fill in missing fields so if address intent remained unspecified the remote application can redirect an action with original application's permission.

f. Null checks On Ipc input

Applications process information from other applications (intent messages) frequently. Null references cause

applications to crash and can be used as denial of service.

g. SD card use

If applications have access to read and write data on the SD card then it can read or write any other applications data on SD card.

h. JNI use

Application uses functionality in native library using java native interface, as these methods are not written in java they have inherited danger.

V. ANDROID SECURITY MODEL

Based on Linux platform and programmed with java, android has a unique security model. Android application run as different process and user id's unlike in desktop application where all applications run having the same UID. Fortunately it doesn't have a single main entry point but their entry points are based on the registration of Activities, Services, Broadcast receiver and content providers with the system. Android code supports self signed certificates which can be easily created by the developers without any one's assistance.

A. Android components based on security[1]

a. Intents

They are for sending data between android processes. They themselves don't enforce any security policy, but they are the ones which pass through the security boundaries of the system. Intent filters do not solve the purpose of security from the perspective of intent receivers. Although intent receivers need to handle opposed or unfriendly callers but intent filters do not have any mechanism to filter the malicious intents. They help the system to find the right handler for the intent but provide no input filtering system. Even categories can be added to intents to be more selective about what code will be handled and what will be restricted.

b. Activities

Only one activity runs on the system at one time, all other activities are suspended at that time. For reuse, replace or improvement of components activities allow application's to call each other. Activities run in their own process and uid's so they don't have access to caller's data except from the data in the Intent. Activities cannot only depend on intent filter tag to stop malicious intents. Activity implementers can depend on permission checks as a security mechanism. If android: permission attribute in an <activity>declaration is being set, this will prevent the programs which don't have the specified permissions from starting the activity directly. If the manifest permissions are specified then the input of the caller is always validated. Developers should avoid of putting data of interest to an attacker into the intents which are used to start activities.

c. Broadcast

Intents are the messages which are broadcast to the broadcast receiver. A manifest permission can restrict that who can send intent to the receiver. Activity manager validates that if the incoming sender has the specified permissions before it delivers the intent. Permissions are the means by which one can ensure that the intents are received from right sender but they don't affect the properties of the message in the intent.

d. Services

Processes running in background are services for e.g. music running in the background. They can also be secured by adding permissions in the android manifest file. Now calling a service like scheduling Mp3 to play is not a sensitive task but task such as storing passwords and private messages, here the services need to be validated.

e. Content Providers

They are used for storing and sharing data. Application uses them to expose their data to the rest of the system. There are separate read and write permissions for reading and writing on the provider. These read and write permissions should be declared directly in Android Manifest.xml. These tags are named as android: read Permission and android: write Permission.

e. I. Avoiding SQL Injection[4]

They can be easily avoided by parameterized queries which distinguish query logic from data explicitly. Update (), delete (), query () of content provider and managed query () of Activity supports parameterization.

f. Mass storage

Android devices have limited memory on internal storage. So to support larger files SD cards come to escape us from this problem but SD card's format is VFAT, which is an old standard and therefore don't supports access control of Linux so the data here is unprotected. Confidential and sensitive data should be stored in encrypted format.

VI. ANDROID PERMISSIONS [4]

Google protect its android users with the help of the permissions which are imposed on every application. Whenever a user tries to install an application, a set of permissions are encountered which the application request. These set of permissions contains all the resources which will be used by the application like for e.g. READ_CONTACTS permission will have access to read entries in a user's phonebook and they also give an insight to the user of the sensitive and harmful resources (if used)and alert him before installing it. The permissions are contained in the manifest file. It is very important for the permissions to match with the application's goal for e.g. A calendar application should have no need of permissions like READ_CONTACTS and SEND_SMS. As the application is installed once then the permissions cannot be reviewed again .So it becomes an important part to examine the permissions carefully. Custom permissions can also be declared and created. As for e.g. SEND_SMS and USER_LOCATION can be collaborated to form a single permission SEND_LOC_SMS. There are only four protection levels for permissions: normal, Dangerous, Signature, and SignatureorSystem.[6]

VII. EVALUATION

On the basis of patternDroid we analyzed the whole set of samples. In the first column we place the samples belonging to different malware and non-malware families. In the second column for each sample i we create two sets that is Si –no. of permissions scanned by the automated tool to be suspicious, Ti – Total no. of files in that sample of malware. The ratio is taken in the second column. In the third column we create Ai on the basis of behavior analysis of stimulator, we manually evaluate that from Si what is the actual no., which is

really being a threat to the system. So we place Ai/ Si in the third column and thus percentage of vulnerability is calculated and placed in the fourth column.

SAMPLES	no. of permissions scanned/no. of files	Behavior analysis on the basis of stimulator	Percentage of vulnerability
campass	3/5	0/3	0
cemara	9/6	0/9	0
Employee_Records	56/12	20/56	35.714
googlem	3/5	0/3	0
icalendar	18/4	8/18	44.44
NotesAnd	48/22	2/48	4.166
Sample	8/4	4/8	50
timersendsms	4/4	2/4	50
videoviewplay	4/4	0/4	0
wifidirectdemo	22/8	3/22	13.63
campass	3/5	0/3	0

VIII. CONCLUSION

By this tool the malware analysis platform has been given a new automation. Instead of manually analyzing each and every file and identifying the suspicious threats this tool gives us the freedom of direct scanning the files and directly knowing that the application will cause harm to the system. Other than analyzing the applications on the basis of the set of permissions prepared by us for the malicious permissions one can also have the independency to manually fit in the suspicious keyword. The final result which we get state the individual files with the number of permissions in each of them which match the malicious permissions and also name the particular permission.

REFERENCES

1. WILLIAM ENCK, MACHIGAR ONTANG and PATRICK MCDANIEL, Proceedings of the 20th USENIX Security Symposium, August, 2011, "Understanding Android Security" [online available]: http://www.css.csail.mit.edu/6.858/2012/readings/android.pdf
2. SASCHA FAHL, MARIAN HARBACH, THOMAS MUDERS, MATTHEW SMITH, LARS BAUMGARTNER,BERND FREISLEBEN CCS '12 Proceedings of the 2012 ACM conference on Computer and communications security , Oct 18, 2012, "Why Eve and Mallory love Android: An analysis of Android SSL(In) Security" [online available]: http://www2.dcsec.uni-hannover.de/files/android/p50-fahl.pdf
3. KIRANDEEP, ANU GARG, The International journal of Engineering and Science, March, 2013, "Implementing Security on Android Application" [online available]:
4. http://www.theijes.com/papers/v2-i3/Part.Vol.%202.3%20%282%29/I0232056059.pdf
5. JESSE BURNS, Black Hat 2009, "Mobile Application Security On Android".
6. WILLIAM ENCK,DAMIEN OCTEAU, PATRICK MCDANIEL and SWARAT CHAUDHRI, In *Proceedings of the 20th USENIX Security Symposium*, San Francisco, Ca, August 2011. " A Study of Android Application Security".
7. ADRIENNE PORTER FELT, ELIZABETH HA, SERGE EGELMAN, ARIEL HANKEY, ERIKA CHIN, DAVID WAGNER, "Android Permissions: User Attention, Comprehension and Behavior".
8. Asaf Shabtai, Yuval Fledel, and Yuval Eloivici, IEEE May-June 2010 ,Security & Privacy, IEEE (Volume:8 , Issue: 3), "Securing Android-Powered Mobile Devices Using SELinux".
9. Shabtai, A, Fledel, Y, Kanonov, U, Elovici, Y ,March-April 2010, Security & Privacy, IEEE (Volume:8 , Issue: 2) "Google Android: A Comprehensive Security Assessment".
10. Miller, C., July-Aug. 2011, Security & Privacy, IEEE (Volume: 9, Issue: 4) "Mobile Attacks and Defense".
11. Stavrou, A, Voas, J. ; Karygiannis , T. ; Quirolgico , S. Feb .,s 2012, Computer (Volume:45 , Issue: 2), "Building Security into Off-the-Shelf Smartphones"
12. Alessandro Reina, Aristide Fattori, Lorenzo cavallaro,6th European Workshop on Systems Security (EuroSec)Prague, Czech Republic, April 14, 2013, A system call centric analysis and stimulation technique to automatically reconstruct android malware behaviors.
13. Iker Burguera , urko zurutuza , simin nadjm tehrani , SPSM '11 Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, 2011,Crowdroid:behavior based malware detection system for android.
14. Mahmud AB RAHMAN, The HoneyNet Project 10th annual workshop, 2011-03-21, Reversing android malware.

AUTHORS PROFILE

Ipta Thakur is pursuing mtech from Chitkara University. Her research interest security related to Smartphone's and cloud computing.

Shaily Jain is presently teaching at Chitkara University. Her research interest is network security.