International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 4, April - 2014

Intrusion Detection Prototype System for Massive Mobile Applications

Chethana S Patil[1]
PG Student, Dept. Of CS&E
Acharya Institute of Technology,
Bangalore, India

Nagesha A G[2]
Associate Professor, Dept. Of CS&E
Acharya Institute of Technology
Bangalore, India

***Abstract* – Mobiles have gained widespread usage& in smart phones many interesting applications are made available through Google Play. Android is one of the major Smartphone platforms today. The intense increases in mobile apps too many threats migrate from conventional PC client to android mobile device. Smartphone applications can steal users' personal sensitive information and send it out across their back .Smartphone's store various personal data such as phone numbers, location information, contact information, sms, passwords. The sensible private information is abused highly without users notice. In fact that majority of the users are not proficient in mobile security. To improve security status of current mobile apps MOBAPP-SAFE prototype is proposed process to valuate mobile Apps based on cloud computing technology. Mobapp-Safe prototype helps to identify whether the mobile App is safe from malwares or they contain malwares. When compared with conventional method, such as permission based method the Mobapp--Safe prototype system associates the dynamic and static analysis methods are used to check the Android Apps. In the implementation Static Android Analysis Framework (SAAF) and Android security evaluation framework (ASEF) the two methods static and dynamic analysis methods respectively is adapted to examine the Android Apps and determines the total time needed to estimate mobile App market which contains all the Apps &It also gives information what type of private data App is leaking. Prototype provides deeper security analysis & the estimation results show it is feasible to use cloud computing for all stored Apps to authenticate regularly to clean out malware apps from the mobile app markets.**

Keywords: Android operating system, malware, cloud computing, Cloud stack, redis key store.

I. INTRODUCTION

Smartphone's are powerful and well connected becoming increasingly common now days & other kinds of mobile devices like tablets has also risen significantly. The fact is accompanied by the n-number and variety of mobile applications is made available. The android operating system is available as an open source tool it is widely popular among developers and various Smartphone manufacturers make use of it & which enables them to add Apps so this strength becomes challenge for providing security. Smartphone applications can leak users' sensitive information and send it out at the back. The worldwide Android Smartphone market raises the proficiency of security and privacy issues however, in traditional Android Smart phones uses permission based approach which is not enough to ensure the security & it is time consuming. It presents a challenge to the engineers for developing security solutions and methods for the platform So in order to overcome security issues and to provide security Mobapp-safe prototype is developed.

A. Mobile App Threats

The intense increase in mobile apps on Mobile devices the Apps uses Internet trends too many PC software's are migrating to the mobile device. Some malicious behaviors of Android malwares in mobile apps are usually motivated by controlling mobile device without the users intervention such Malicious behaviors are as follow:

- Privilege escalation causing flaw in design.
- Leakage private data.
- Dial premium numbers were it increases the call rates.
- Botnet activity lead denial of service attacks.

B. Root causes for Android malicious mobile apps origin

- Android operating system allows users to introduce any mobile apps from the third-party market that may make no efforts to check the safety of the software in app that they distribute.
- Easy and Flexible to port an Windows-based botnet existing client apps to Android based Smartphone's.
- Application developers in android can upload their apps without any intervention check of trustworthiness& certificate authority.
- A number of mobile applications have been modified in the app market and the malwares have been introduced in application and spread through unofficial repositories. The sophisticated malwares in app detect the presence of an sohpiscated environment and adapt their behavior pattern e.g. Create hidden background processes when they find security layer.

II. PROBLEM STATEMENT

To improve current security status of mobile applications & to avoid leakage of the sensible private information which is abused highly such as phone numbers and other personal information without users notice by third party applications & to detect malware apps.

III. RELATED WORK

Android Apps security analysis is a important as more Smartphone users exists and it is hot topic. In existing system static analysis and dynamic behavior analysis and machine learning techniques are used by too many researchers to carry out security analysis.

A. Static analysis methods

Static analysis is security analaysis which is carried out during installation time of android apps and many static tools are available. Stoway tool is used to detect an over privileges in Android application. It checks whether data flows through the stated specification. Android access control policy was determined through techniques and it fails to detect reliable permission information and the stoway tool was proposed by Felt et [1].

Barrer et al[4] made an security analysis using static method called permission based method it is a novel methodology which applies Map organizing self algorithm were relationship proximity was preserved greatly in complex dataset to present to present relational view. The disadvantage of permission method it fails detects sophisticated malwares.

Nadji et al [5] proposed static tool airmid, which uses collaboration between the smartphone devices and network sensors to identify the malicious traffic. They used three mobile malwares to test the correctness of airmid. Airmid's carry out remote repair and it consists of a device attribution system and it is server-based infection detection system. The airmid methods as disadvantage the permissions are coarse grained do not provide sufficient insight of the actual, permissions. Potential malicious activities can be performed after installation of App.

B. Dynamic behavior analysis

Dynamic behavior analysis is security analysis where the analysis of android mobile App is done at the run time .So it belongs to the run time app monitor category. Many dynamic analysis tools are available in existing system dynamic analysis detects the abnormal execution patterns through behavior signatures. The Author proposed Paranoid Android [6] a security analysis to detect the malicious app. a system can carry out malware analysis in the mobile phone replicas. This method as disadvantage it causes serious problems such as accessing call logs, contact information, personal details. Zhou [7] author proposed security analysis method DroidMOSS which make use of fuzzy hashing technique to localize and to detect the changes from app behavior in mobile. The disadvantage of it consumes more time.

C. Machine learning

Schmidt et al[8] Proposed a machine learning solution to detect malicious app based on monitoring events occurring on kernel. Read elf tool was used to read static information held by files and after applying read elf tool to both normal apps and mailcious apps, they used the calls of the functions and names appearing to form their safe app data set and malicious data set. The main disadvantage is it is costly, time consuming & high sensitive information data leaks are not detected.

IV. THE PROPOSED METHODOLOGY

The cloud computing platform a methodology is proposed to valuate mobile apps for checking security status and for the improvement of current security MOBAPP-SAFE, a prototype system is proposed to check the whether the mobile app is safe free from malwares or they contain malwares (mobile app which is malware free). Mobapp-Safe associates the active and passive analysis methods to check an Android mobile apps and estimation time is reduced and the total analyze time is an acceptable level. In the implementation, the two methods dynamic and static analysis methods are used as follows ASEF (Android Security Evaluation Framework) and SAAF (Static Android Analysis Framework) to examine the Android apps to detect malicious app and estimate the total time needed to check the mobile app markets were all the Apps are stored are safe or no & mobile app market owner get useful data to clean out the unsafe mobile apps and it gives the information about the leakage of sensitive data.

A. *system architecture*

The system architecture gives the Mobapp-safe prototype conceptual design which as work flow that defines the structure and behavior of a system. Architecture is a formal description of a system in which it is organized in such way that supports reasoning about the structural properties of the system. It also defines the system components or building blocks and provides a plan from which the system developed. The architectural design process is concerned with establishing the basic structural framework for a system. The major components of the system and communications between these components are identified by the system architecture as follows:

1) Infrastructure for cloud platform: Infrastructure is provided by the cloud stack based on cloud computing platform it can be used to carry out security analysis task. Cloud stack is used to manage a VMware based computing servers. The whole Cloud Stack manages the network servers, storage and compute all nodes that makes cloud infrastructure were Mobapp-safe prototype is placed.

2) Work principle: Mobapp-Safe is a prototype which is used to check an Android apps is virulence or non-virulence and if virulence what type of private sensitive data App is leaking information will be provided based on cloud computing. Using some customized tools. The architecture of system is shown in Fig1. Mobapp-Safe is an prototype system which can be used to carry out security analysis of android apps. When you upload an unknown Apk file to Mobapp-Safe for analysis, it will check the key in store whether the upload apk file is already undergone analysis and its result is stored in database. The Comparison of apk file is based on the hashing technique as the key to query in the redis key value store. The redis version is 2.1.3 is used if the key is matched in redis, then the response is returned as result to submitter.

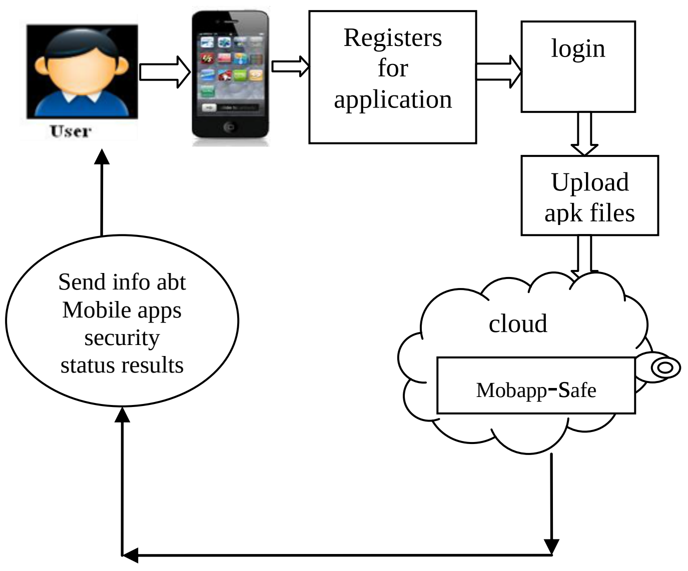

User
Registers for application
login
Upload apk files
cloud
Mobapp-Safe
Send info abt Mobile apps security status results

Fig.1 System Architecture

If it is new Apk file then key is not matched. In that case, the Apk is stored in database. After that, a daemon invokes the frameworks, such as ASEF and SAAF to collect the log files and store them in specific directory to carry out security analysis. Also the prototype inserts the key to redis and the value is updated with the result directory in database.

B. *DESIGN & IMPLEMENTATION*

Entire Mobapp-safe prototype system can be divided into the following modules:

1) User registration / login platform: Any user can register for the platform with an email id and phone number. The platform generates the user id and password for the user which will be active for 15 days and these details will be sent to the user via email and sms which was used at the time of registration. Once the user logins for the first time with the password sent the user will be asked to change the password and after successful change of password, user can again login to application and gains full access to the platform.

2) Platform analyzing application: Once the authorized user logins to the platform & the user can upload the apk file from an Google app or the browser app which is then sent to the platform for the security analysis. The platform applies SAAF and ASEF algorithm which completely checks for the file to find malicious app which contain malwares .It also finds the vulnerabilities and reports the status to the user which is also stored in the database. If the app to be analyzed is already present in the Database the status is directly passed to the user. The platform analyzing application module plays a very vital role in finding whether app is safe or unsafe and it collects data why the app is a unsafe and what type of private data it is leaking from the android based mobile so user can come to know security status. The Working of Mobapp-safe is shown in fig. 2.

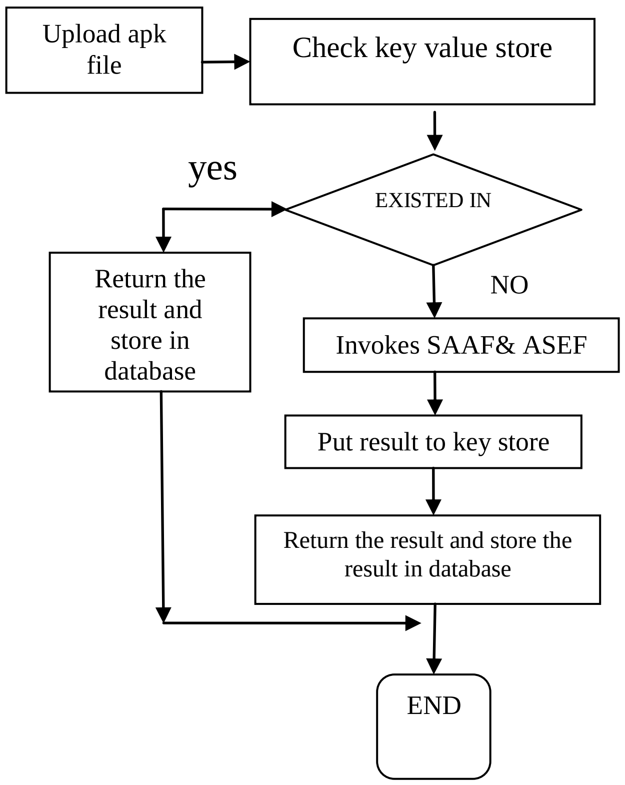


Fig.2 The process of Mobapp-Safe prototype

The frameworks used in the module are:

a) ASEF: ASEF is an automatize framework which can carry out security analysis for Android application in mobile whether the app is safe or unsafe When an unknown apk file is uploaded to ASEF for analysis As shown in fig 3 it as three phases: active, passive, interpret firstly it will start the passive phase were it is initial phase normalization of app will be carried then it will send that data to active phase behavioral data will be collected to run test cycle and then it send to the interpret phase were data will be parsed then launch an Android Virtual Machine(AVD) were ADB logging and traffic sniffing using TCPDUMP is done and install the application on it. Then ASEF will start to install the application to be examined for analysis and send a number of behaviors it will be collected to simulate human integration on the mobile application. Meanwhile ASEF also compares the log file of CVE library with activites connected to internet with API of safe Google browser of android virtual machine. After that certain numbers of behaviors are sent to virtual machine randomly then the test circle is ended and the application will be removed. ASEF will start to analysis the log file and the traffic generated by app through internet. ASEF uses API Google Safe Browsing to find out URLS of the app trying to reach whether they contain malwares or not framework also checks the existed vulnerabilities with a known vulnerabilities list to examine whether the application has some dangerous Vulnerabilities.

b) SAAF: SAAF is a framework which is used for security analysis it is static analyzer for Android apk files. It can extract the data of apk files and decode the apk file content to smali code and then it will divide the program into small chunks of smali code. SAAF unpacks these APK files in the following way in order to carryout analysis the data-flow analysis and further analysis operations are done:

- The analyst loads an Android application (apk file).
- SAAF unpacks the contents of the app and generates smali files for all classes, using the android-apk tool. Working directly on the byte code enables us to obtained a detailed view of the code it overcomes limitations of tools that rely on decompile the byte code to Java code.
- SAAF then parses all the smali files and creates appropriate objects for representation of its contents. we process the Manifest file of app, basic blocks of the methods, fields, and all opcodes are collected to analyze the permissions of apps to match behavior patterns to detect malicious app which is leaking private data.

c) *Performance Evaluation by ASEF Framework:* In order to measure performance metrics how much time ASEF framework takes time to analyze an mobile app, record can be done by writing script The apps are installed in three category Android Smartphone's. The beginning of a program the time stamp is used and ASEF framework is used to analyze all different Android mobile apps downloaded from app market. The result is shown in Fig. 3, what the time it takes to analyze one application varies from 60 s to 140 s, and the average time is taken 100 s. It means that we can finish the security analysis and acquire the results in less than 1 min on average whether the app is malware free can be known. It follows 6 steps to analyze: Preparing app, start collecting log files, service step were installation is done and testing is carried out by collecting all the behaviors of app. As shown in Fig.3.

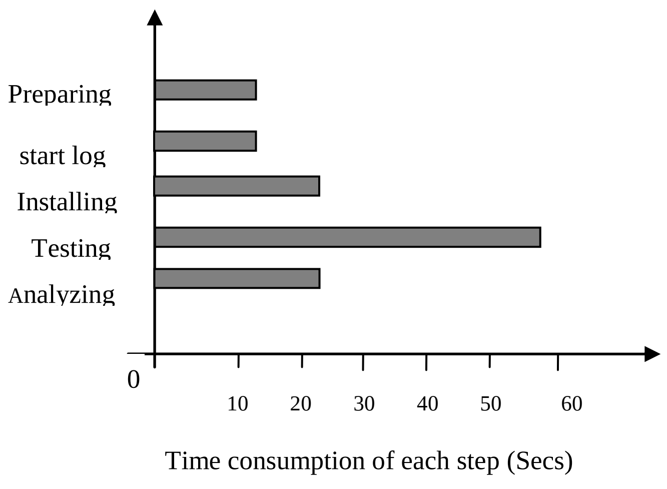


Fig.3 ASEF framework time consumption analysis

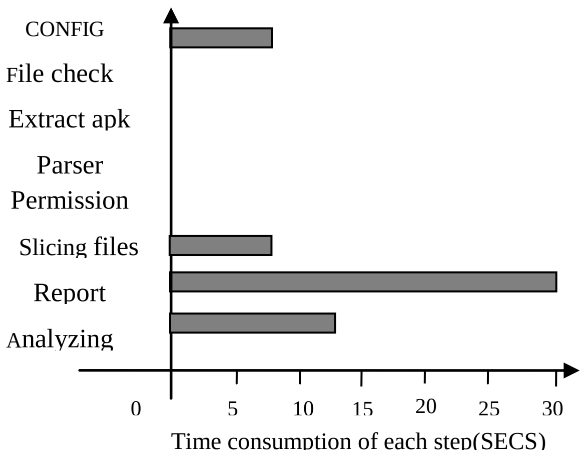


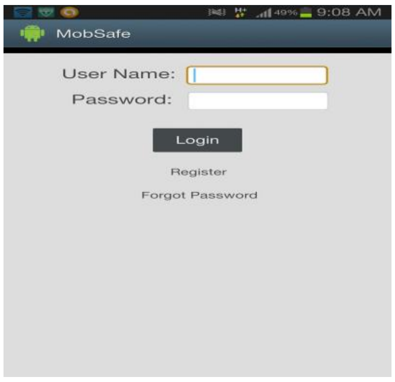
Fig.4 SAAF framework time consumption analysis

d) *performance evaluation by SAAF framework:* To calculate performance Metrics in security analysis of app whether it is free from malwares we make use of SAAF framework for Android apps downloaded from App markets for checking the performance static smali code of this framework is used From Fig. 4 above, we can see that the time consuming step of SAAF framework is the slicing files into chunks step, and the second is the permission check step. The average time of analyzing one app consumed by SAAF framework is less than 1secs. Time depends on the complexity of app but SAAF framework complete the analysis in acceptable time duration. In this way security analysis can be carried out in quicker manner to submit response to users.

3) *Admin user for approval tasks and generate reports:* Since the user has authorized gain to the application for only 15 days, after this time the user can request for extension and the admin user approves such request. Since large numbers of apps are uploaded for analysis the admin user can see all the analyzed files with the status and also generate the reports in the graphical format for the same data.

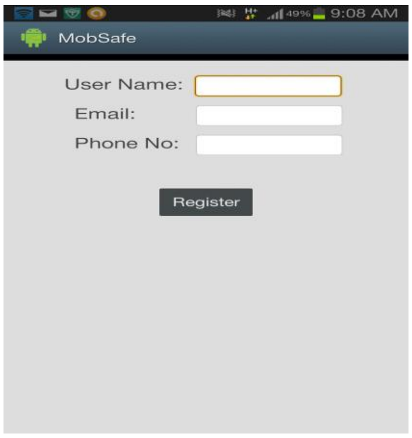
V. RESULTS

1) *User login window:*



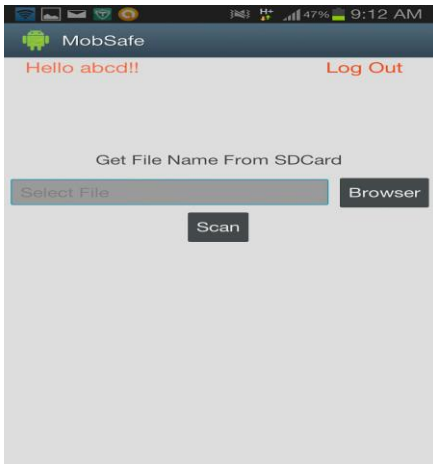
This is user login window where users can login to check App by using username & password if the credentials are True then user can access the platform.

2) *User register window:*



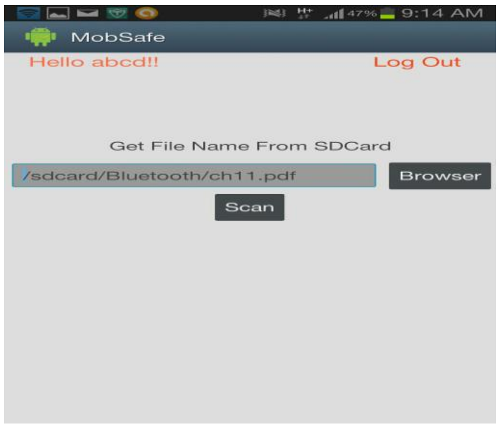
This window shows If the user is using the app for the first time the user should get registered for the platform with an user name email-id and phone number.

3) Upload apk file window:



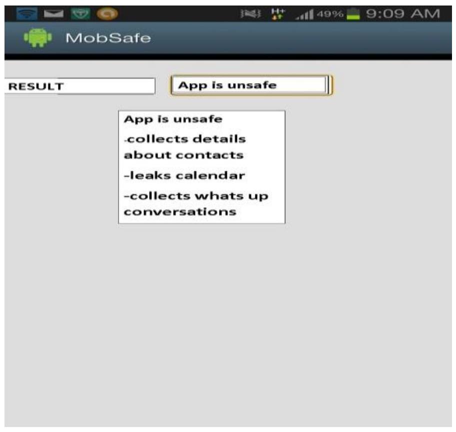
This window shows Upload apk files to verify the apps are safe unsafe.

4) Apk file is uploaded window:



This window shows apk from mobile file is uploaded.

5) Result window App is unsafe:

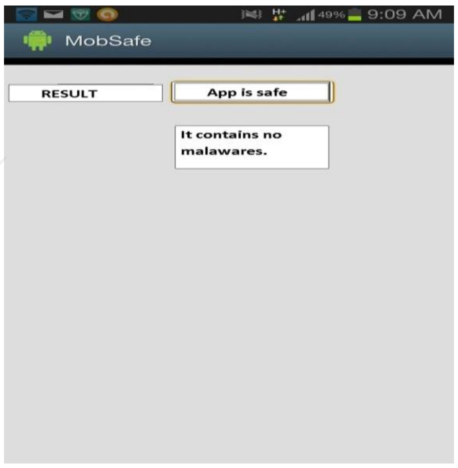


App is unsafe

App is unsafe
-collects details about contacts
-leaks calendar
-collects whats up conversations

This window shows the result apk file is analyzed whether the app contains malwares as result shows the uploaded apk file is unsafe.it is unsafe as its leaks private data.

6) Result window App is safe:

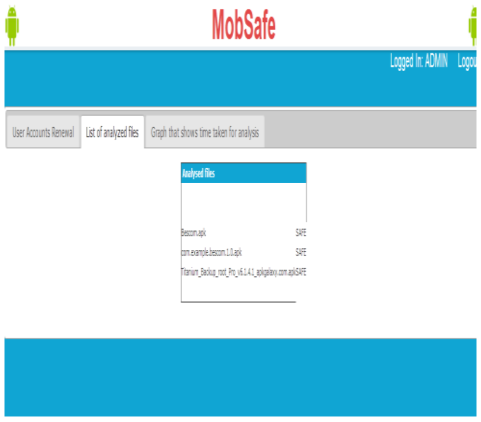


App is safe

It contains no malawares.

This window shows it analyzes and checks the app whether it is safe or it contains malwares.

7) Admin window:

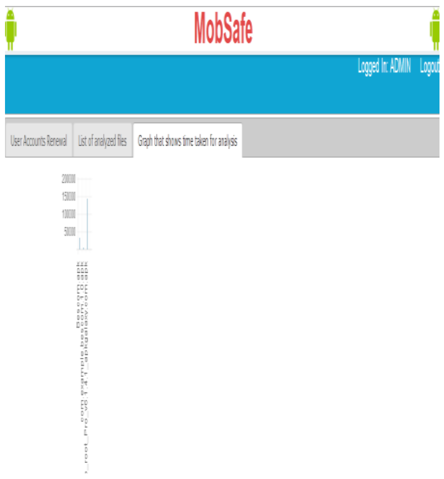


Analysed files

Bescom.apk SAFE
com.example.bescom.1.0.apk SAFE
Titanium_Backup_root_Pro_v6.1.4.1_apkgalaxy.com.apkSAFE

This window shows admin can login see the user account status& list of Analyzed files.

8) Admin Analyzed Report:



This window shows admin can see graph of all analyzed apk files and time consumed by apps to undergo security analysis

VI CONCLUSION

The proposed methodology is to improve security status & verify the security of Android mobile apps to indentify whether mobile apps contain malwares or no based on cloud computing technology. The memory and time consumption during the security analysis is less & classifies new malware under few seconds with less percent of impact on performance & power consumption is also less. The prototype system Mobapp-Safe can be implemented for security analysis of mobile Apps were static code and active behavior of App is analyzed ASEF and SAAF are the two methods which carry out active analysis and passive analysis can be used to examine the Android Apps and calculates the total time needed to examine all the stored Apps which are collected in a mobile App market and regularly to filter out malware Apps The scope of the mobapp-safe prototype system it can solve practical real life problems such as it can protect & promotes the android markets. Developers and owner of App markets would benefit from integrating Mobapp-safe prototype. They can also protect and promote a market being targeted by an attacker. Before downloading any app into smartphone, a user can see what this prototype has to say about the behavioral analysis of the App.

REFERENCES

[1] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, Android permissions demystified 18th ACM Conference on Computer and Communications Security, Chicago, USA, 2011, pp. 627-638.

[2] Q. Feng, Android software security and reversing engineering analysis, Feb. 2013.

[3] http://techcrunch.com/2013/05/29/mary-meeker-2013- internet-trends/, May 29, 2013

[4] Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji, A methodology for analysis of permission-security models and its application to Android, in Proc. 17th ACM Conference on Computer and Communications Security, Chicago, USA, 2010, pp. 73- 84

[5] Nadji, J. Giffin, and P. Traynor, Automated remote repair for mobile malware, in Proc. 27th Annual Computer Security Applications Conference, Orlando, USA, 2011, pp. 413-422. J. Wu, On Top of Tides (Chinese Edition), Beijing: China Publishing House of Electronics Industry, January 8, 2011.

[6] Sango Lee and Da Young Ju International Journal of Security and Its Application Vol.7, No.5(2013.

[7] Gartner http://www.gartner.com/it/page.jsp?id=2153215, September 11, 2012.

[8] Schmidt, R. Bye, H. G. Schmidt, J. Clausen,O. Kiraz, K. A. Yuksel, S. A. Camtepe, and S. Albayrak, Static analysis of executables for collaborative malware detection on Android, in Communications, ICC'09, IEEE International Conference on, Dresden, Germany, 2009

[9] List of mobile software distribution platforms, http://en.wikipedia.org/wiki/List of digital distribution platforms for mobile devices, July 19 2013. .

[10] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, A study of android application security, San Francisco, USA, 2011.