iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 4 **Issue:** VIII **Month of publication:** August 2016

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 4 Issue VIII, August 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Enhancing the Android Mobile App Security using SHA-512 Algorithm

E.J. Thomson Fredrik [1], A. Theerthagiri [2]
[1]Associate Professor, Department of Computer Applications, Karpagam University, Coimbatore-21
[2]Final Year MCA Student, Karpagam Academy of Higher Education, Coimbatore-21

Abstract: ***Nowadays most of the smart phones are using Android Operating System. Everyday millions of people are purchasing products using Android mobile App. But smart phone users are very much concerned about the security in Internet transactions. Although the Smart Phone has rich set of features, the vulnerability for security attacks by viruses also increases every day. The security of data stored in a Smartphone is given more importance by Android App developers. Installation of every Android app asks for some critical permission to access our critical files and we have to accept the permissions in order to install that application. We propose an approach to increase security for Android Smart phone using our App. Our approach integrates the Android Security Framework with SHA-512 algorithm to make E-Commerce transactions through Android App more secure. The proposed enhanced security framework enhances the security of Android smart phone from the virus and malware. A novel approach to secure the data on Smartphone's using cryptographic Algorithms is also discussed in this paper.***
Keywords: Smartphone, Android app, Security Framework, Malware, SHA-512 Algorithm

I. INTRODUCTION

Android is a modern mobile platform that is designed to be truly open source. Android applications can use advanced level of hardware and software, as well as local and server data, exposed through the platform to bring innovation and value to consumers [12]. Open source platform needs strong and rigorous security architecture to provide security. Android is designed with multi-layered security that provides flexibleness needed for an open platform, whereas providing protection for all users of the platform designed to a software stack, android includes an operating system, middleware and core application as a complete. Android architecture is designed with keep ease of development ability for developers. Security controls have designed to minimize the load on developers. Developers have to simply work on versatile security controls. [3]

Developers are not familiar with securities that apply by defaults on application. Android is also designed with focused on user's perspective. Users can view how applications work, and manage those applications. Smartphone have assumed an increasingly vital role in the lives of Information and Communication Technology (ICT) users [8]. Today, a constantly increasing number of consumers use Smartphone for a broad variety of tasks and purposes, ranging from social networking to instant messaging, from mobile banking to Global Positioning Satellite (GPS) based navigation. Smartphone serve as special tools for organizing the users' daily lives through productivity applications such as calendars, memos or calculators. [1] The apps downloaded from Google Play like True Caller, Viper, Whatsupp, India Live and billions of other apps have changed the life of a typical android user with 1.2 billion people worldwide using mobile apps at the end of 2012. With these many number of users using the apps downloaded from the Google play, securing the use of these apps is of paramount importance to researchers. Trusted apps are available in Google's market which is self- signed by the developers but Malware has even appeared in Google's market. Two examples are Droid Dream and Droid Dream Light [2]. Both these apps were found on the Android market in early 2011 and both applications steal personal data and are very much like traditional Trojans seen on the desktop.

It is all too common to hear about these bad apps that steal and alter our valuable data and can make our Smartphone non functional, and those discussions always end with one thing -- someone says you need to read an app's permissions before you install it. The existing android security framework is providing only less security for E-payment transactions [7]. Therefore, we propose a method to integrate the Android Security Framework with cryptography algorithm to make E-Commerce transactions through Android App more secure [13]. We also propose an enhanced security framework which will restrict the access of vulnerable apps by checking the behavior of these apps. In section 2, we discuss the existing works in the security of Android smart phones. In section 3, we discuss the security features of Android and its limitations. In section 4, we describe the proposed security Android Framework

©IJRASET: All Rights are Reserved

using SHA-512 algorithm which will protect our data on our personal Smartphone. Finally we draw our conclusions in section 5.

II. LITERATURE SURVEY

In [3], S. Kaur and M. Kaur,2013 presented review paper on 'implementing security on Android application'. In that paper, they described how security can be improved in android based system so that users can safely use the android smart phones.

In [11], Powar and Meshram,2013 reviewed the work on android security framework. They concluded that the increased exposure of open source Smartphone is increasing the security risk. Android provide a basic set of permissions to secure phone. The technique to make Android security mechanism more versatile, the current security mechanism is too rigid. User has only two options at the time of application installation first allow all requested permissions and second deny requested permissions leads to stop installation

In [10], Machigar Ongtang, *et al*,2009 studied on 'Semantically Rich Application-Centric Security in Android'. In their research work, they augment the existing android operating system with a framework to meet security requirements. They proposed secure application interaction (Saint), an improved infrastructure that governs install-time permission assignment and their run-time use as dictated by application provider policy. Saint provides necessary utility for applications to assert and control the security decisions on the android platform.

In [14], Schmidt,A.,D., *et al*, 2008 studied on 'enhancing security of Linux-based android devices'. They presented an analysis of security mechanism in Android Smart phones with a focus on Linux. The results of their analysis can be applicable to Android as well as Linux-based Smartphones. They analysed android framework and the Linux- kernel to check security functionalities. They surveyed well- accepted security mechanisms and tools which could increase device security. They provided details on how to adopt these security tools on Android platform, and overhead analysis of techniques in terms of resource usage 11. Their second contribution focuses on malware detection techniques at the kernel level. They tested applicability of existing signature and intrusion detection methods in android platform.

In [9], Lackorzynski,A., *et al*, 2011 presented 'L4Android: a generic operating system framework for secure smartphones'. In this title they present a generic operating system framework that overcome the need of hardware extensions to provide security in smartphones. They encapsulate smartphone operating system in a virtual machine, this framework allows highly secure applications to run side-by- side with the virtual machine. It is based on a state-of-the-art micro-kernel that ensures isolation between the virtual machine and secure applications13 .In [5], Gibler,C.,*et al*,2012, studied on 'Android Leaks: automatically detecting potential privacy leaks in android applications on a large scale'. They have presented a static analysis framework for automatically searching potential leaks of sensitive data in android applications on a large scale. AndroidLeaks drastically reduces the number of applications and the number of traces that a security auditor must verify manually.

In [15], Tesfay,W.B., *et al*,2012 presented 'reputation based security model for android applications'. They have proposed a cloud based reputation security model as a solution which greatly mitigates the malicious attacks targeting the Android market 19. This security solution uses unique user id (UID) which is assigned to each application in the android platform. This model stores the reputation of Android applications in an anti-malware providers cloud (AM Cloud). The experimental results witness that the proposed model can identify the reputation index of a given application and its potential of being risky or not.

In [4], Feth,D., and Pretschner,A.,2012 proposed 'Flexible data-driven security for android'. They proposed an improved security system beyond the standard permission system. It is possible to enforce complex policies that are built on temporal, cardinality, and spatial conditions in this system. Enforcement can be done by means of modification or inhibition of certain events. Leveraging recent advances in information flow tracking technology, policies can also pertain to data rather than single representations of that data.

III. SECURITY OF ANDROID

Android is a Linux-based mobile operating system programmed with Java and implemented with its own security framework. Android combines OS features like efficient shared memory management, preemptive multi-tasking of processes, Unix user identifiers (UIDs) for each of its programs in execution and file permissions with the type-safe features of Java language and its well-known API library. The resulting security framework is much more like a multi-user server than the sandbox found on the J2ME platforms. Unlike a desktop computer operating system where a user's applications all run under the same UID, Android applications are individually partitioned from each other. Android applications run in distinct processes under distinct UIDs each with different set of permissions. Programs have no permission to read or write each other's files/data or code, and sharing data/files

©IJRASET: All Rights are Reserved

between applications must be done explicitly by the programmer. The Android GUI environment has some novel and distinct security features that help support this isolation of processes.

The permission based model is the basic mechanism for securing access to various files or resources in Android. Although the app permissions are categorized to different protection levels such as Normal, Dangerous, Signature and Signature-Or-System, the assignment of these protection levels of various resources is left to the developer's will and his/her own understanding. This feature of Android Security framework may lead to attacks by malicious software and a number of vulnerabilities in the security framework of Android. When an application is downloaded and installed by the user on Android, the Android framework prompts the user to accept a list of required permissions, the user may grant all of the permissions in order to install the application successfully or deny the permissions to cancel the installation. Practically, there are a number of security issues in such a framework: 1) The user must accept and grant all of the required permissions in order to install the application successfully, 2) once the app is installed and permissions are granted; there is no mechanism for restricting an application to revoke the permissions already granted 3) there is no way of restricting access to the resources based on dynamic constraints as the permission model is based on install-time check only, 4) granted permissions can only be revoked by uninstalling the application.

At the time of installation, the user is presented with a dialog box listing all permissions requested by the app to get successfully installed. These permission requests are defined in an XML File called AndroidManifest.xml, which is shipped with every Android app.

However, this security framework has a few drawbacks [3]:

A. All or No Permission

A user cannot grant single permissions, while rejecting others in order to install the app. Among the list of permissions an app might request a suspicious permission among the other legitimate permissions, will still be able to confirm the installation.

B. Often, the users of the app cannot judge the appropriateness and legitimacy of permissions for the app in question. In some cases it may be well understood, for example when a chess game app requests the privilege to reboot the Smartphone or to send SMS messages. In many cases, however, users will simply not be able to understand the appropriateness of the permission.

C. Functionality, which is supposed to be possible only given the appropriate permissions, can still be achieved with less number of permissions or even with none at all.

IV. PROPOSED SECURITY FRAMEWORK

The objective is to provide security against the Apps which are installed by the end user and is given all the permissions at the time of installation. This enhanced security has the desirable property of not disturbing a regular user in any noticeable way.

Data/Files are not stored in encrypted format within media.

The lack of strong security control of user´s private information that permits malware to access the information stored in the device.

Configurable firewalls are not integrated into Smartphone operating systems.

A. Malware Detection

File operations offered by the proposed Security API should aid in the detection of potentially malicious Apps whose behavior matches that of Malware. Malware recognition is usually achieved by signature matching, heuristic analysis, or comparing hash-values. We provide details in the following sections about how our proposed security approach can provide for these malware recognition techniques.

B. Signature Matching

Our Proposed API shall provide a method to conduct pattern matching using regular expressions. That method will only return true or false for any given pattern described by a regular expression passed to the method. Currently one or multiple byte sequences are found in common signatures which can be linked in various relations, such as logical (AND, OR) or regarding location in a target file. Such relations can be implemented through regular expressions which will ensure user and system data privacy, while still allowing signature-based detection. Different pattern matching algorithms can also be used, which can improve pattern matching performance or compatibility with the use of existing vendor signature databases.

©IJRASET: All Rights are Reserved

C. Hashes

Malware can also be detected using hash functions of SHA-512 Algorithm. Desktop malware is usually spread in a highly decentralized manner via exploitation of software vulnerabilities, and often the same malware may be spread by many people for whom it has been "personalized". This leads to hashes being a less useful approach. On mobile platforms, however, with difficult app vulnerability exploitation and with centralized software distribution, hash-based malware detection gains value.

D. Implementation of SHA-512 Algorithm for Android E-Payment transactions

SHA-512 is a novel hash functions computed with 32-bit and 64-bit words, respectively. They use different shift amounts and additive constants, but their structures are otherwise virtually identical, differing only in the number of rounds. SHA-512 was published in 2001 by the NIST as a U.S. federal standard. In 2005, an algorithm emerged for finding SHA-1 collisions in about 2,000-times fewer steps than was previously thought possible. Since SHA-1 Algorithms are weaker than SHA-2 algorithms. Therefore, SHA-1 algorithms are no longer recommended for applications that depend on collision resistance, such as digital signatures. Since SHA-512 hash algorithm comes under the SHA-2 algorithms, We use SHA-512 for enhancing android smart phones security in this paper.

There are various steps involved in the SHA-512 algorithm. They are listed as follows:

1) Message Padding
2) Append Length
3) Divide the Input into 512 bit blocks
4) Initialize chaining variables
5) Process Blocks
   a) Copy variables to register
   b) Divide one 512 bit block into 16 blocks of 32 bit each
   c) 4 rounds, each round consisting of 20 steps.
   d) Diagram + process P + all chaining variables.

Step 1: Padding

Adding padding bits to the original message is the first step of SHA-512 algorithm. The main objective of this step is to make the length of the original message equal to a value which is 64 bits less than an exact multiple of 512.

For example, if the original message is 900 bits, then we add a padding of 60 bits which makes the message length 960 which is hence 64 bits less than 1024 (1024=512 × 2). The padding consists of a single 1 bit followed by as many 0's bits as required. It is mandatory to add padding bits even if the original message length is itself 64 bits less than the multiple of 512.

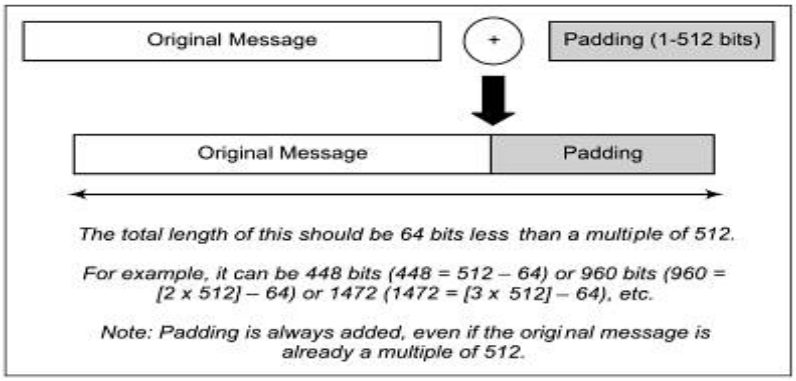


Fig.1 SHA-512 Padding

Step 2: Append Length

The next step after adding padding bits is to calculate the original length of the message and append it to the end of the message after padding. The length of the message is calculated excluding the padding bits i.e. the length of the message before the padding bits were added. For example, if the original message was of 900 bits and a padding of 60 bits was added to make the length 64 bits less than the multiple of 512 then here the length is considered 900 bits instead of 960 bits. This length is now expressed as a 64 bit value and appended to the end of the original message + padding. This process is better explained by the figure. Now if the length of

©IJRASET: All Rights are Reserved

Volume 4 Issue VIII, August 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

the original message exceeds 264 bits then only lower order 64 bits are used here i.e. length mod 264 is calculated in that case. Hence the length of the message is now an exact multiple of 512. This becomes the message whose message digest will be calculated.

Step 3: Divide the Input into 512 bit blocks

The next step is to divide the input message into blocks, each of length 512 bits. Now these blocks become the input to the message digest processing logic.

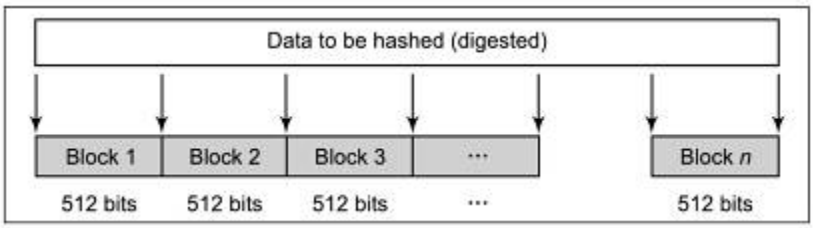


Fig.2 Dividing the Input to 512 Bit Blocks

Step 4: Initialize the chaining variables

There are five chaining variables A through E. These five chaining variables are initialized in this step. MD5 had four chaining variables each of 32 bits (total length will be 4 × 32=128 bits) but in the case of SHA-512, we need a message digest of 160 bits hence there are five chaining variables here making a total of 5 × 32=160 bits. The values for these chaining variables are as shown in the figure 3.

A	Hex	01	23	45	67
B	Hex	89	AB	CD	EF
C	Hex	FE	DC	BA	98
D	Hex	76	54	32	10
E	Hex	C3	D2	E1	F0

Fig.3 Initialize the chaining variables



Fig.4 Android Application of E-commerce System

©IJRASET: All Rights are Reserved

E. Advantages of the Proposed System

Passwords used in the E-Commerce transactions are encrypted with SHA-512 algorithm. Therefore the users of android smart phones are not necessary to scare about the hacking of their passwords. If a novice user has installed any malicious app and has given all the permissions at install time, this Security API will restrict the app from accessing any of the files. Our proposed system uses the latest techniques to detect Malware which includes signature matching and Hash functions using SHA-512 algorithm. This enhanced security has the desirable property of not disturbing a regular user in any noticeable way. In fact, the user need not even be aware that the Security techniques have been applied.

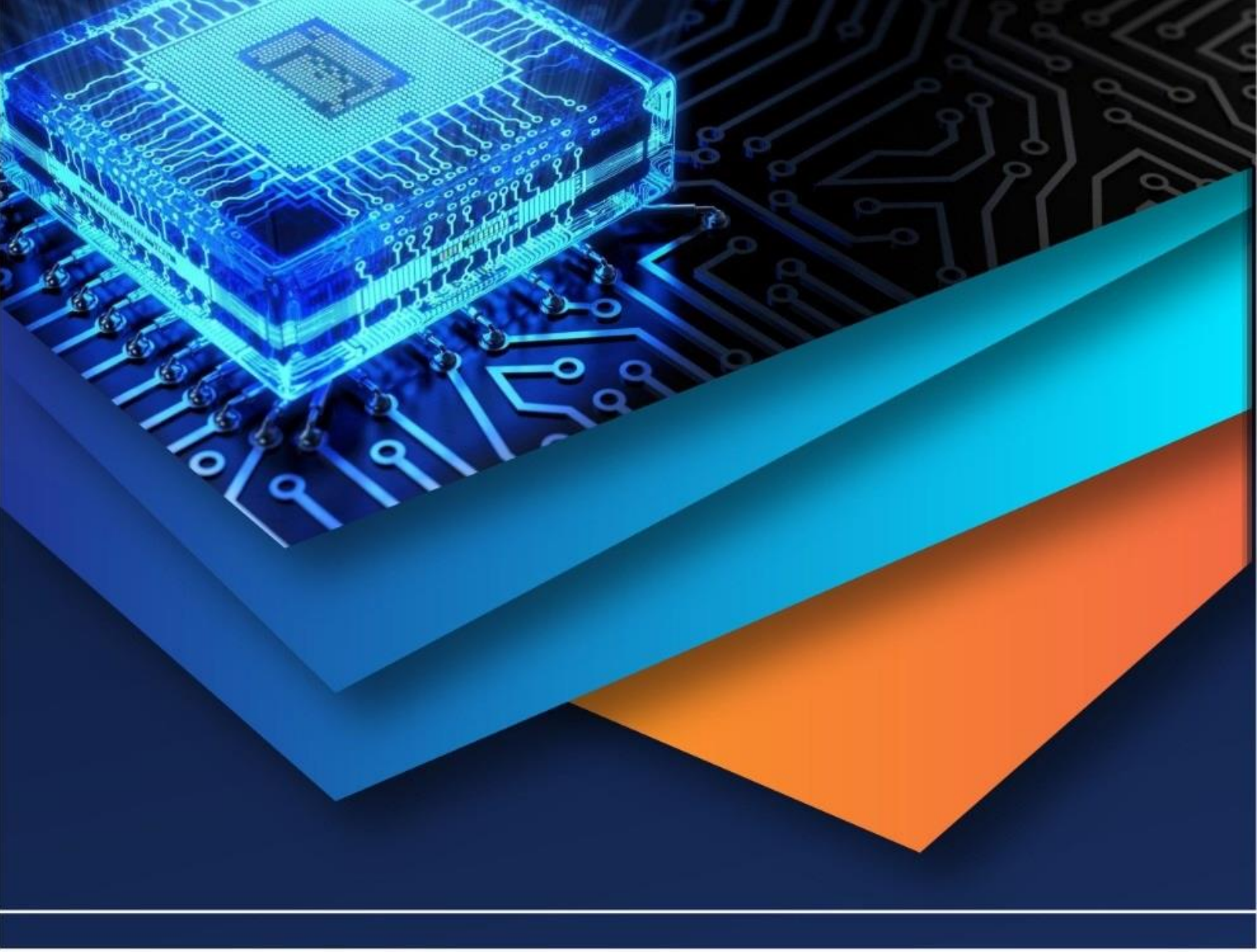
V. CONCLUSION

With the current security architecture, many Smartphone operating systems are vulnerable to attacks because the Smartphone user is instrumental in deciding which applications will be installed on the phone. It is not easy for a user to judge applications by their description. The Android framework is one platform that expects the user to be security conscious and implicitly assumes applications developers are not malicious. Because of this, a user may unknowingly install software that poses a security threat or is not efficient enough to handle the user's privacy issues. Our aim is to provide a system to free the user from making decisions as to which applications to install and to provide protection to the user's personal files and data from any malicious apps downloaded from Google store. Thus, our proposed Security API enables users to install the apps and if the built-in security of Android is not able to prevent the unauthorized access of critical data, then this enhanced security framework will provide necessary safeguards. The proposed enhanced security framework detects Malware and protects the E-Commerce transactions from hackers using the SHA-512 algorithm.

REFERENCES

[1] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev, Chanan Glezer, "Google Android: A Comprehensive Security Assessment", IEEE Security & Privacy, Vol.8, no. 2, pp. 35-44, April 2010.
[2] Dignan,L.,"Malware sneaks by google's android market gatekeepers again,"http://www.zdnet.com/blog/security/malware-sneaks-bygoogles-android-market-gatekeepers-again/ 8696 .
[3] Enck,W., Ontang,M., and McDanial,P., "Understanding Android Security," IEEE Security & Privacy Magazine, 7(1), 10-17, 2009.
[4] Feth D., Pretschner A., Flexible Data-Driven Security for Android, The 2012 IEEE Sixth International Conference on Software Security and Reliability, 41-50 ,2012.
[5] Gibler C., Crussell J., Erickson J. and Chen H., Android Leaks: Automatically Detecting Potential Privacy Leaks In Android Applications on a Large Scale, 5th international conference on Trust and Trustworthy Computing, 291-307, 2012.
[6] Kaur S. and Kaur M., Review Paper on Implementing Security on Android Application, Journal of Environmental Sciences, Computer Science and Engineering & Technology, 2(3), 2013
[7] Kirti.P Lokhande, Avinash.P.Wadhe,"Security in Android File System", International Journal of Advanced Research in Computer Science and Software Engineering, Vol 3, Issue 12, December 2013
[8] Konstantinou,E., and Wolthusen,S., Metamorphic virus: Analysis and detection. Technical report,Information Security Group at Royal Holloway, University of London, 2009.
[9] Lackorzynski A., Lange M., Warg A., Liebergeld S., Peter M., L4Android: A Generic Operating System Framework for Secure Smartphones, 18th ACM Conference on Computer and Communications Security, 39-50, 2011.
[10] Machigar Ongtang, McLaughlin, William Enck , Patrick McDaniel, "Semantically Rich Application-Centric Security in Android", IEEE International Conference on Computer Security Applications,2009.
[11] Powar,S.,Meshram, "Security on Android Security Framework", International Journal of Engineering Research and Applications, Vol 3, Issue 2,2013.
[12] Rafael Fedler, Marcel Kulicke and Julian Sch¨utte 2013 IEEE 8th International Conference on Malicious and Unwanted Software: "The Americas" (MALWARE)
[13] Ruben Jonathan Garcia Vargas "Security Controls for Android", IEEE Fourth International Conference on Computational Aspects of Social Networks (CASoN) ,2012
[14] Schmidt A.D., Schmidt H.G., Clausen J., Camtepe A., Albayrak S. and Yuksel K. Ali and Kiraz O., Enhancing Security of Linux-based Android Devices, http://www.dailabor.de/fileadmin/files/publications/lk2008-android_security. pdf ,2008.
[15] Tesfay W.B., Booth T., and Andersson K., Reputation Based Security Model for Android Applications, Trust, Security and Privacy in Computing and Communications, IEEE Computer Society, 896-901,2012

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)