International Journal of Computer Applications (0975 – 8887)
Volume 137 – No.1, March 2016

Android Security Vulnerabilities Due to User Unawareness and Frameworks for Overcoming Those Vulnerabilities

Tauseef Ibne Mamun
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh

Lamia Alam
Department of Computer Science and Engineering
Military Institute of Science and Technology
Dhaka, Bangladesh

ABSTRACT

With the popularity of Android smart phones everyone finds it convenient to make transactions through these smart phones. And the users of these smart phones, in most cases unaware of different types of threats. The purpose for this survey paper is to conduct a survey on users to get the information about the security vulnerabilities they are creating unknowingly, bringing forward some security frameworks for these threats & giving a basic knowledge to the new comer to the android about android OS architecture and the threats to this architecture.

General Terms

Android security architecture, Security vulnerabilities occur due to user unawareness, Security frameworks

Keywords

Android Security Framework, Security Vulnerabilities, User Unawareness, Android App Permission, Malware Detection, Survey on User Awareness.

1. INTRODUCTION

A smart phone is an intricate combination of a mobile phone and a computing platform with powerful computing system and high speed connectivity. In the market of smart phones, android dominates the market with 78% share. *S*mart phones have become indispensable part of our daily lives in recent years, since they are involved in keeping in touch with friends and family, doing business, accessing the internet and other activities. Andy Rubin, Google's director of mobile platforms, has commented: "There should be nothing that users can access on their desktop that they can't be access on their cell phone" [1]. We are keeping data which are private in nature inside our smart phone for easy access. Since users keep a huge amount of data in our smart phone, the hackers are targeting our smart phones more and more. In this paper a survey is done to see what vulnerabilities occur due to a user's unawareness. Some security frameworks are also discussed which will help to remove these vulnerabilities if these frameworks are adopted in an android phone.

2. ANDROID SECURITY ARCHITECTURE

Android seeks to be the most secure and usable operating system for mobiles by providing security measures to protect user data, system resources and it isolate applications. Google provides following security features to achieve these objectives

- Robust security at the operating system level through the Linux kernel
- The OS is sandboxed, preventing malicious processes from crossing between applications
- Secure interposes communication
- User defined permissions.
- Application signing

Despite of this attempts, it fails to address the issue of infection all together

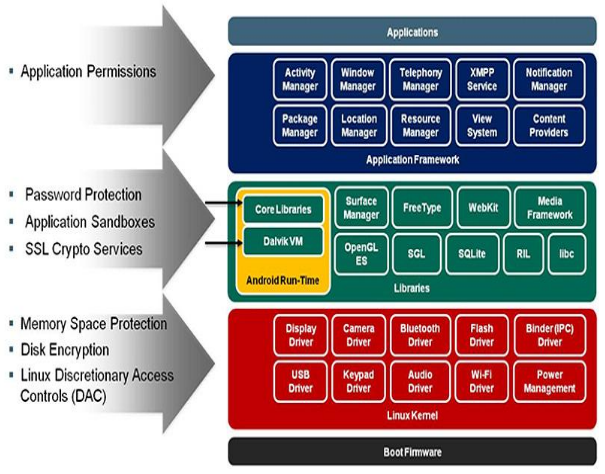

Figure 1: Summarizes security provided at various levels of Android. Every level assumes that level below is properly secured.

In the Linux Kernel, the main purpose of memory space protection is to prevent a task from accessing memory without proper access permissions. Without memory protection, memory segment like code and data segment are vulnerable to memory related bugs and code injection attacks. Disk encryption ensures that files are always stored on disk in an encrypted form. **Security Enhanced Linux** (SELinux), an access control implementation for the Linux kernel which was introduced recently has prevented multiple vulnerabilities, and now it has been strengthened even more to meet the needs of enterprise customers that have strict security requirements. All process run above the Linux kernel is restricted by the application sandbox.

In the Libraries, The Android platform takes advantage of the Linux user-based protection as a means of identifying and isolating application resources. This approach is different from other operating systems (including the traditional Linux configuration), where multiple applications run with the same user permissions. This sandbox is dissimilar than the sandbox found on the J2ME or Blackberry platforms.

3. ANDROID THREATS

Security breach on this architecture may come in two ways from outside offensive activities; attack due to user unawareness and attack due to system defects. Most attacks exploit vulnerabilities of the smart phone. The threats we see appearing on mobile are rootkits, Trojans, and even botnets. Since new malwares are appearing almost on a daily basis and it is hard to replace depicted secured device with a latest secured device in this pace by a user; user awareness can go in a great length to stop outside interferences to the current device.

4. SURVEY ON USER AWARENESS

For research purpose we conducted a survey on a focus group of 20 people while they are using android device. We asked following questions

a. How often you install third party applications (third party means applications which are not from Google play store)?

b. Have you rooted your android device at least once?

c. How often you connect to an unsecured WI-FI network?

d. Do you give your android device remote access to your PC?

e. Do you maintain unsecured Bluetooth connection?

f. When you disconnect an external device from your android device how often you check for virus in the device after disconnection?

g. Do you click on unknown emails and any spam contents in social media?

h. When you install an application do you check all the permissions on the permission list?

i. Do you know the risk of third party sites?

The summary result of our survey on user awareness is shown in the following figures (for details: - https://www.surveymonkey.com/results/SM-GTHMR3CQ/):

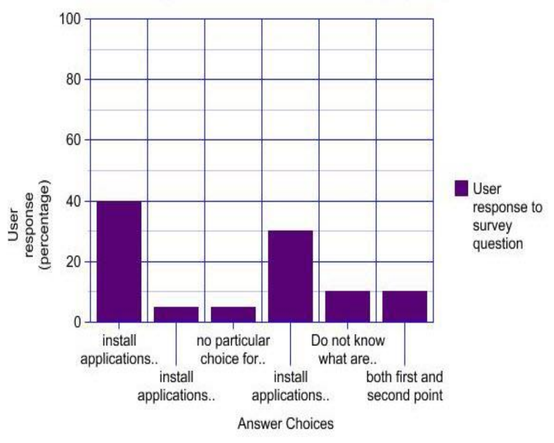

Figure 2: How often you install third party applications?

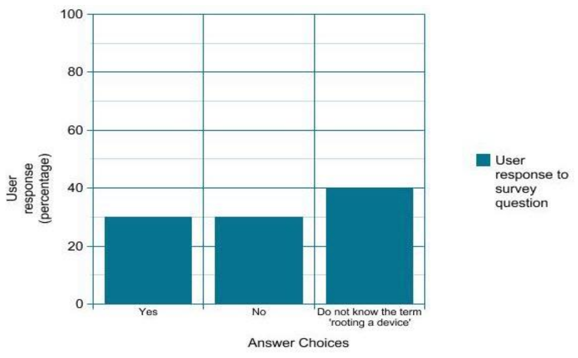

Figure 3: Have you rooted your android device at least once?

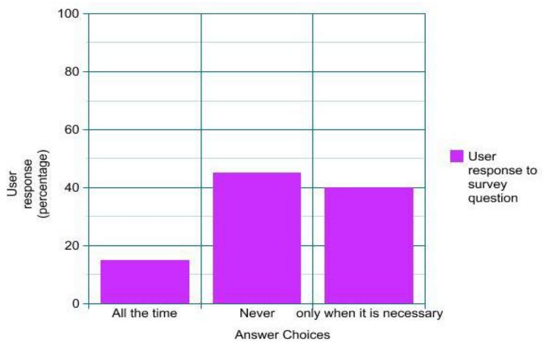

Figure 4: How often you connect to an unsecured WI-FI network?

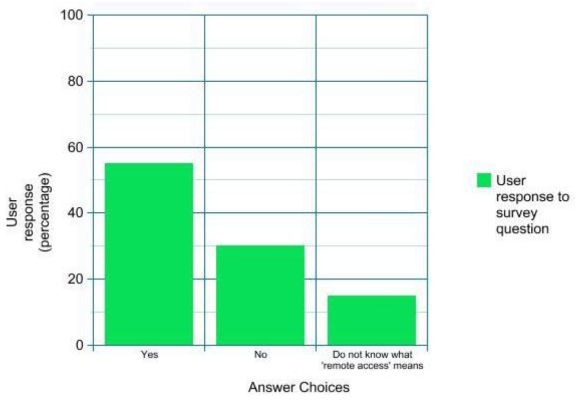

Figure 5: Do you give your android device remote access to your PC?

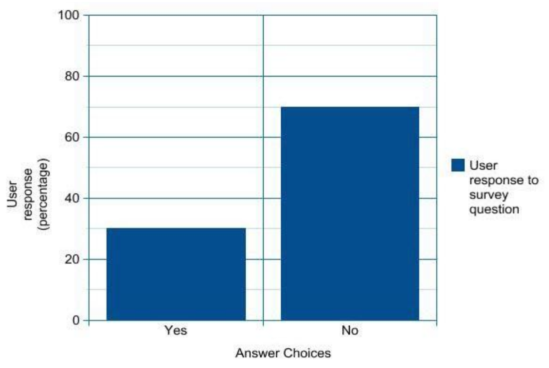

Figure 6: Do you maintain unsecured Bluetooth connection?

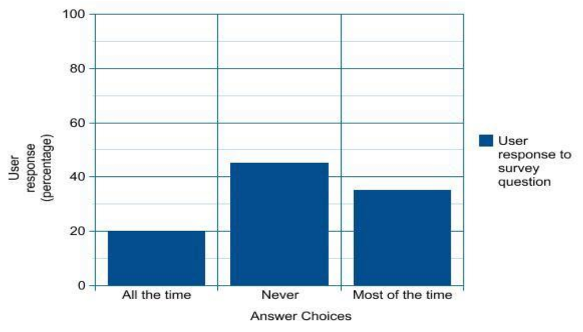

Figure 7: When you disconnect an external device from your android device how often you check for virus in the device after disconnection?

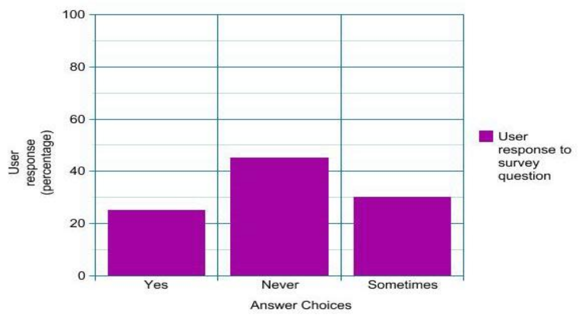

Figure 8: Do you click on unknown emails and any spam contents in social media?

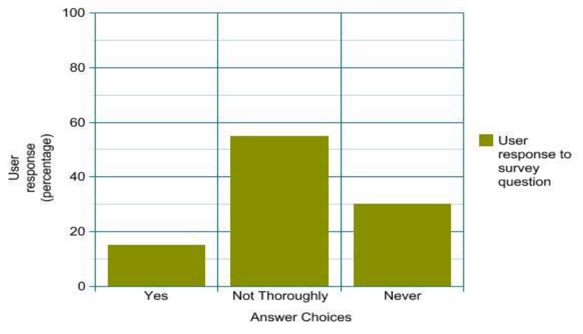

Figure 9: When you install an application do you check all the permissions on the permission list?

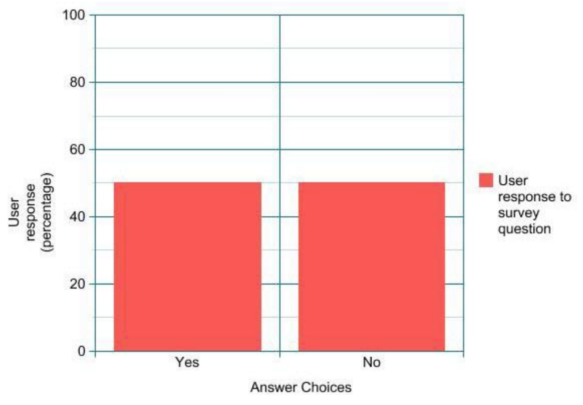

Figure 10: Do you know the risk of third party sites?

5. VULNURBILITIES DUE TO USER UNAWARENESS

After going through the survey from focus group; numerous vulnerabilities were identified which are caused due to user unawareness. This vulnerabilities occur a when a user

- installs third party applications
- roots a device
- connects to a unsecured WI-FI network& maintaining unsecured Bluetooth connection
- gives remote access to PC
- connects to SD card and external device
- clicks on spam emails/sms/mms
- grants unnecessary permission to an application

Studying these vulnerabilities, an android device is prone to these following attacks due to user unawareness

1. **Spam**: Spam is generally sent in SMS, MMS and email. VoIP and Instant Messaging (IM) have also become common ways for spamming. It is seen from our survey that 25% people click on unknown links or spam contents from their e-mail or messenger.
2. **Malware**: Users are not always aware of downloaded applications' functions. Even if applications have acquired explicit user consent, users may be unaware that the applications are executing malicious code[2].A case study shows that,in AAMU after connecting to the WiFi using Intercepter-NG, after running the scan command it showed all the devices with IP addresses that are connected to AAMU WIFI. After that, the application started to collect packets that is sending and receiving through this WIFI. In most case, this application can collect the information like user name and password.
3. **Peripheral Interfaces Attacks**: Smart phones usually have many peripheral interfaces, such as Wi-Fi, Bluetooth, USB, etc. While peripheral interfaces can increase smart phones communication capabilities, unfortunately, they also become a popular steppingstone for outside attacks. In our survey it is seen that most of the people do not connect to unsecured Bluetooth connection. So it is possible that users are much aware about this type of security risk.
4. **Data hijacking**: Granting unnecessary permission gives attackers full control over the contents (e.g. photo gallery) and information (e.g. location) of a phone. Private and confidential data can easily be hijacked by an attacker.

5.1 Permission

Permissions are the rights that a specific application has that allow it to perform certain actions on a device. Examples of these actions include taking pictures, using the GPS, reading contacts, or making phone calls. All applications have their permissions available for users to check; many users do not check the permission properly and thus cyber criminals can exploit user information for their personal gain. In our survey, only 15% people said that they check all permission before installing an app.

5.2 Third Party Installation

One of the main advantages of android is that as it is an open source operating system, one can easily install applications which do not belong to the google play store from the internet. In 2012, researchers uncovered an increase in the number of malicious domain accounts related to Android apps. From approximately 3,000 domains in January 2012, the number jumped to almost 8,000 by the end of the year. These malicious domains host suspicious .APK files or files containing data needed in Android app installation. Just an example of these malicious apps is the recent fake versions of the popular Candy Crush app with features that can be abused by cyber criminals [3]. By using these features, they can get hold of your important data and aggressively push ads onto your device. In the survey, all the people in our survey keep personal data and private documents in the mobile. We have seen from our survey that most of the people often install app from third party websites.

5.3 Rooting

Rooting an Android phone simply means to gain administrative privileges on the system. For malware publishers, if they got the root control; they could design malwares to get users' private information and credential. [4] Researchers from University of California, Berkeley, have chosen 6 most popular Android systems from 2010 to 2011 to count the days that root exploits are exposed, and shown that the percent of time with known root exploit are very high. The least one is 74%. It means that the root exploit is exposed only one day per five days. [5] It is observed from our survey that most of the people even do not know the exact meaning of "Rooting."

5.4 Unsecured Connection

We have already stated that majority people do not maintain unsecured Bluetooth connection. Most of them know threats of android security mainly come from the attacks during data transformation. Malicious Applications also make unauthorized actions when Android exchanging data through the technology such as messaging, wireless network and Near Filed Communication. 45% People from our survey told that they never connect to unsecured Wi-Fi network though 40% people told that they use unsecured network when they find it necessary. 55% people said that they do give their android phone remote access to their PC. 9 out of 20 people said they do not check for virus in their smart phones after disconnecting from an external device.

5.5 Accessing Unknown Contents

"Stagefright" is the nickname given to a potential exploit that lives fairly deep inside the Android operating system. Stagefright that's used to process, play and record multimedia files. Some of the flaws in android allow for remote code execution and can be triggered when receiving an MMS message, downloading a specially crafted video file through the browser or opening a Web page with embedded multimedia content. On a finding, researchers of FireEye [6], a security company found out an example of such exploit. Users just have to click on the featured link in the email and the malicious .apk (Android Package File) is downloaded. Researchers at FireEye went through HTTP requests and found nearly two-dozen URLs serving up the .apk, some disguised as LabelReader.apk. According to them this malware isn't entirely new. It surfaced earlier and is known for deceiving users into paying for cleanup of other non-existent infections on their device. As long as the user pays the fee, the phone will purportedly remain uninfected with malware

6. LITERATURE SURVEY

Number of frameworks has been proposed to encounter android security vulnerabilities and many re-search works have been done regarding this perspective. We have discussed few of them here to explain how threats can be minimized facing with the vulnerabilities. We will represent them in tabular form in the paper.

Paper Name	Abstract View
Scandroid: Automated security certification of android applications [7]	SCANDROID is a tool proposed by Fuchs, et al. for reasoning automatically about the security of Android applications to understand the flow of information from one component to another component. -SCANDROID's analysis is modular to allow incremental checking of applications as they are installed on an Android device. Based on information flow, it can make security-relevant decisions automatically. -It can decide whether it is safe for an application to run with certain permissions judging the permissions enforced by other applications. -Though SCANDROID is among one of the first program analysis tool for Android, it suffers from the limitation of security policy express ability. If a policy writer does not define certain constraints before executing the policy, an information flow will not be explicitly added to the set of constraints and the framework will consider it to be safe.
Semantically Rich Application-Centric Security in Android[8]	Secure Application INTeraction (Saint) framework proposed by Ongtang, et al. is a modified infrastructure that governs install-time permission assignment and their run-time use as dictated by application provider policy. -Saint's install-time policy regulates granting of application defined permissions. More specifically, an application declaring permission P defines the conditions under which P is granted to other applications at install-time. -Saint's runtime policy regulates the interaction of software components within Android's middleware framework. Any such interaction involves a caller application that sends the IPC and callee (B) application that receives that IPC. The IPC is allowed to continue only if all policies supplied by both the caller

	and callee are satisfied. -This framework is not user-centric as it gives the option of policy specification to the application developers and not to the user.
A methodology for empirical analysis of permission-based security models and its application to android[9]	D. Barrera, H. Güne¸ S. Kayacık, P.C. van Oorschot, A.Somayaji study on ‘a methodology for empirical analysis of permission-based security models and its application to android’. -In the paper, the proposed methodology is of independent interest for visualization of permission based systems beyond current Android-specific empirical analysis. -They provide some discussion identifying potential points of improvement for the android permission model, trying to increase quality where required without increasing number variety of permissions or overall complexity.
Android Permissions Demystified[10]	In this paper, Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, David Wagner determined the general causes of over-privilege and provided ideas about how to avoid it. -A permission map is developed that can be used by existing or future tool to further study permission usage in Android. It determines whether Android developers follow least privilege with their permission requests. -An applicant analysis is done to compare manual and automatic analysis that provides opportunity to calculate false positive rates and eventually gage how accurate the remaining automatic analysis was. -A tool Stowaway is built that detects over-privilege in compiled Android applications. Stowaway determines the set of API calls that an application uses and then maps those API calls to permissions. The authors used automated testing tools on the Android API in order to build the permission map that is necessary for detecting over-privilege.
Application collusion attack on the permission-based security model and its implications for modern smart phone systems[11]	C. Marforio, A. Francillon, S. Capkun show technique in which permission based mechanisms are used on mobile platforms allows attacks by colluding applications that communicate over explicit and covert communication channels. -These security bugs allow applications to indirectly execute operations that those applications, based on their declared permissions, should not be able to execute. Example operations include disclosure of user’s private data (e.g., phone book and calendar entries) to remote parties by applications that do not have direct access to such data or cannot directly establish remote connections. -They further showed that on mobile platforms users are not aware of possible implications of application collusion. Application permissions should be displayed to the users differently, reflecting their actual implications.
Survey on android security framework[12]	S. Powar, Dr. B. B. Meshram described android security framework in this paper. 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.
AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale[13]	AndroidLeaks, a static analysis framework is presented by C. Gibler, J. Crussell, J. Erickson and H. chen in this paper for automatically finding potential leaks of sensitive information in Android applications on a massive scale. -AndroidLeaks drastically reduces the number of applications and the number of traces that a security auditor has to verify manually. The authors evaluated the efficacy of AndroidLeaks on 24,350 Android applications from several Android markets. AndroidLeaks found 57,299 potential privacy leaks in 7,414 Android applications, out of which are manually verified that 2,342 applications leak private data including phone information, GPS location, WiFi data, and audio recorded with the microphone.
CRePE: Context-Related Policy Enforcement for Android[14]	This is a context based security system for android. A context can be defined by the status of some variables (e.g. Location time, temperature, noise, and light), the presence of other devices, a particular interaction between the user and the smart phone, or a combination of these. -CRePE allows context-related policies to be defined either by the user or by trusted third parties. Depending on the authorization, third parties can set a

	policy on a smart phone at any moment or just when the phone is within a particular context, e.g. within a building, or a plane. -To protect users' privacy, the current security models restrict trusted third parties' control on mobile phones. Typically, only the device manufacturer and the telephone company have a small control on the smart phone. There are no mechanisms to allow other authorized parties. By implementing this framework we control applications like Wi-Fi or Bluetooth.
TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones[15]	TaintDroid is an extension to the Android mobile-phone platform that tracks the flow of privacy sensitive data through third-party applications. -TaintDroid assumes that downloaded, third-party applications are not trusted, and monitors–in real time. How these applications access and manipulate users' personal data. Analysis of applications' behavior requires sufficient contextual information about what data leaves a device and where it is sent. -TaintDroid automatically labels (taints) data from privacy-sensitive sources and transitively applies labels as sensitive data propagates through program variables, files, and inter process messages. When tainted data are transmitted over the network, or otherwise leave the system, TaintDroid logs the data's labels, the application responsible for transmitting the data, and the data's destination.
On Lightweight Mobile Phone Application Certification[16]	Kirin provides: -a methodology for retrofitting security requirements in Android. As a secondary consequence of following the methodology, it identified multiple vulnerabilities in Android, including flaws affecting core functionality such as SMS and voice. -a practical method of performing lightweight certification of applications at install time. This benefits the Android community, as the Android Market currently does not perform rigorous certification. -practical rules to mitigate malware. These rules are constructed purely from security configuration available in application package manifests.
PREC: Practical Root Exploit Containment for Android Devices[17]	PREC performs classified system call monitoring by separating the system calls originated from high risk third-party native code from the system calls issued by the less dangerous Java code. (e.g., third-party native libraries) and execute those system calls within isolated threads. -PREC can detect and stop root exploits with high accuracy while imposing low interference to benign applications. After extracting the system calls from the high-risk native code, it needs to build a normal behavior model for the app before it is released to the market. -The behavior model is then transferred to the smart phone device for runtime root exploit detection. A new lightweight and robust behavior learning scheme based on the self-organizing map (SOM) technique is used in this process.
Apex: Extending android permission model and enforcement with user-defined runtime constraints[18]	Apex is an extension to the Android permission model that is more user-centric in allowing applications to access the device resources. -Apex allows users to specify detailed runtime constraints to restrict the use of sensitive resources by applications. It is designed to overcome the limitation that the Android framework grants all the permissions to an application, which the application requests at install time. -There are some limitations in the Apex framework. In the current Android architecture, the application developers assume that all the permissions that their application requests will be present in the manifest file. The developers often do not handle the unexpected security exceptions that are thrown when an application requests to access some resource(s) but the application does not have the required permissions to access it.

Crowdroid: behaviour-based malware detection system for Android[19]	In this paper, I. Burguera, U. Zurutuza, S. Nadjm used earlier approaches for dynamic analysis of application behaviour for detecting malware in the android platform. The detector is embedded in framework for assortment of traces from limitless number of real users supported crowd sourcing. -This framework has been demonstrated by analysing information collected in the central server using two sorts of data sets: those from artificial malware created for test functions, and people from real malware found in the world. -The technique is shown to be an effective means of analytic the malware and alerting the users of a downloaded malware. This method is avoiding the spreading of a detected malware to a larger community.
Andromaly: a behavioural malware detection framework for android devices[20]	The proposed framework realizes a Hostbased Malware Detection System that continuously monitors various features and events obtained from the mobile device and apply Machine Learning anomaly detectors to classify the collected data as normal or abnormal. -They developed four malicious applications and check Andromaly's ability to detect new malware based on samples of known malware. -They evaluated many combinations of anomaly detection algorithms, feature choice methodologies in order to find out the combination that yields the best performance in detecting new malware on android.
MADAM: a multi-level anomaly detector for android malware[21]	MADAM can monitors android at the kernel-level and user-level to notice real malware infections using machine learning techniques to differentiate between normal behaviours and malicious ones. The primary prototype of MADAM is able to notice several real malware found in the world. The device is not affected by MADAM due to the low range of false positives generated after the training phase.
Static analysis of executables for collaborative malware detection on android[22]	The contribution of this title is twofold. First, A.D. Schmidt, R. Bye, H.G. Schmidt, J. Clausen, O. Kiraz, K. Yuksel, A. Camtepe, and S. Albayrak, perform static analysis on the executables to extract their operate calls in android environment using the command readelf. Method call lists are matched with malware executables for classifying them with part, Nearest Neighbour Algorithms and Prism. Second, they present a cooperative malware detection approach to improve results.
Reputation based security model for android applications[23]	In this work, W. B. Tesfay, T. Booth, and K. Andersson proposed a cloud based reputation security model as a solution which greatly mitigates the malicious attacks targeting the Android market. -This security solution takes advantage of the fact that each application in the android platform is assigned a unique user id (UID). The solution stores the reputation of Android applications in an anti-malware providers' cloud (AM Cloud). -The experimental results witness that the proposed model could well identify the reputation index of a given application and hence its potential of being risky or not. -A given anti-malware provider is able to keep track of the number of their users, who are running any given Android application. If a given vendor's application has under ten users, the reputation would be extremely low (unknown reputation). The reputation would increase (good reputation) as hundreds, thousands and even millions of users run a given application.

7. CONCLUSION

This paper presented the existing research proposals for removing vulnerabilities caused due to user unawareness. It was found that the prime threat is install time granting access without reading the permission list. Fortunately from API level 23 Google introduced run time permission granting option. But runtime granting permission may be tedious to the user. So whether it would be fruitful is still uncertain. After studying the frameworks in this paper, there is a future scope to build a new framework for tackling multiple threats to android phone.

8. REFERENCES

[1] http://news.bbc.co.uk/2/hi/technology/7266201.stm

[2] Choosilp, Wichien, and Yujian Fu. "A Case STUDY OF MALWARE DETECTION AND REMOVAL IN ANDROID APPS."

[3] http://blog.trendmicro.com/trendlabs-security-intelligence/the-hidden-dangers-in-third-party-app-sites

[4] Tse, Daniel, X. Liu, Christopher Nusaputra, B. Hu, Y. Wang, and M. W. Xing. "STRATEGIES IN IMPROVING ANDROID SECURITY." (2014)

[5] www.acumin.co.uk

[6] http://www.konsultek.com/10/cyber-attacks-2/fireeye-discovers-emails-carrying-malware-in-android-devices

[7] Fuchs, Adam P., Avik Chaudhuri, and Jeffrey S. Foster. "Scandroid: Automated security certification of android applications." *Manuscript, Univ. of Maryland, http://www. cs. umd. edu/avik/projects/scandroidascaa* 2, no. 3 (2009).

[8] Ongtang, Machigar, Stephen McLaughlin, William Enck, and Patrick McDaniel. "Semantically rich application-centric security in Android."*Security and Communication Networks* 5, no. 6 (2012): 658-673.

[9] Barrera, David, H. Güneş Kayacik, Paul C. van Oorschot, and Anil Somayaji. "A methodology for empirical analysis of permission-based security models and its application to android." In *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 73-84. ACM, 2010.

[10] Felt, Adrienne Porter, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. "Android permissions demystified." In *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 627-638. ACM, 2011.

[11] Marforio, Claudio, and Aurélien Francillon. *Application collusion attack on the permission-based security model and its implications for modern smartphone systems*. Department of Computer Science, ETH Zurich, 2011.

[12] Powar, Swapnil, and B. B. Meshram. "Survey on Android Security Framework." *International Journal of Engineering Research and Applications*3, no. 2 (2013): 907-911.

[13] Gibler, Clint, Jonathan Crussell, Jeremy Erickson, and Hao Chen.*AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale*. Springer Berlin Heidelberg, 2012.

[14] Conti, Mauro, Vu Thien Nga Nguyen, and Bruno Crispo. "CRePE: Context-related policy enforcement for Android." In *Information Security*, pp. 331-345. Springer Berlin Heidelberg, 2011.

[15] Enck, William, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones." *ACM Transactions on Computer Systems (TOCS)* 32, no. 2 (2014): 5.

[16] Enck, William, Machigar Ongtang, and Patrick McDaniel. "On lightweight mobile phone application certification." In *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 235-245. ACM, 2009.

[17] Ho, Tsung-Hsuan, Daniel Dean, Xiaohui Gu, and William Enck. "PREC: practical root exploit containment for android devices." In *Proceedings of the 4th ACM conference on Data and application security and privacy*, pp. 187-198. ACM, 2014.

[18] Nauman, Mohammad, Sohail Khan, and Xinwen Zhang. "Apex: extending android permission model and enforcement with user-defined runtime constraints." In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pp. 328-332. ACM, 2010.

[19] Burguera, Iker, Urko Zurutuza, and Simin Nadjm-Tehrani. "Crowdroid: behavior-based malware detection system for android." In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pp. 15-26. ACM, 2011.

[20] Shabtai, Asaf, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. "“Andromaly”: a behavioral malware detection framework for android devices." *Journal of Intelligent Information Systems* 38, no. 1 (2012): 161-190.

[21] Dini, Gianluca, Fabio Martinelli, Andrea Saracino, and Daniele Sgandurra. "MADAM: A Multi-level Anomaly Detector for Android Malware." In *MMM-ACNS*, vol. 12, pp. 240-253. 2012.

[22] Schmidt, Aubrey-Derrick, Rainer Bye, Hans-Gunther Schmidt, Jan Clausen, Osman Kiraz, Kamer Yüksel, Seyit Camtepe, and Sahin Albayrak. "Static analysis of executables for collaborative malware detection on android." In*Communications, 2009. ICC'09. IEEE International Conference on*, pp. 1-5. IEEE, 2009.

[23] Tesfay, Welderufael Berhane, Todd Booth, and Karl Andersson. "Reputation based security model for android applications." In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pp. 896-901. IEEE, 2012.