Secured SMS Receiver System for Android Devices

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Technology
in
Computer Science and Applications

Submitted By
Naman Sharma
Roll No. 601303020

Under the Supervision of
Mr. Gurpal Singh Chhabra
Lecturer

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

JUNE 2015

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Secured SMS Receiver System for Android Devices"**, in partial fulfillment of the requirements for the award of degree of Master of Technology in *Computer Science and Applications* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mr. Gurpal Singh Chhabra* and refers other researcher's work which are duly listed in the reference section.

The matter presented in this report has not been submitted in part or full to any other University or institute for the award of any degree.

Naman Sharma

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Mr. Gurpal Singh Chhabra)
Lecturer
Computer Science and Engineering Department

Countersigned By:

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. S. Bhatia)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

First of all, I would like to express my gratitude to **Mr. Gurpal Singh Chhabra, Lecturer,** Computer Science and Engineering Department, Thapar University, Patiala for introducing me man in the middle attacks and for all his guidance and support. This thesis work was enabled and sustained by his vision and ideas. I have been amazingly fortunate to have an advisor like him who gave me the freedom to explore new ideas on my own and at the same time the guided me to recover when my steps faltered. His patience and support always helped me overcome many crisis situations and successfully complete this dissertation.

I am also thankful to the entire faculty and staff of Computer Science and Engineering Department (CSED) and my friends who devoted their valuable time and constantly supported me in all possible ways towards completion of this work. I thank all those who have contributed directly or indirectly to this work.

Lastly, I would also like to thank **my parents** for their years of unyielding love and encouragement. They have been my backbone and have supported me in all walks of life.

Naman Sharma

(601303020)

Abstract

Now-a-days the place of SMSs has been substituted by various chat applications like “Whatsapp”, “Hike” etc. No one uses SMSs that frequently to pass messages from one to another because of the arrival of the free data services and cheap data packs from the ISPs. But SMS still plays a very vital role in our day to day lives and its theft is increasing concerns to secure it. No airline or bank or subscription services send their transaction details through any web application but through SMS. Now, what if these important SMSs could be traced by some malicious person or a machine which is dedicated to do some evil activities? Let us take an example of a normal routine bank transaction, first the details are filed online that are public and anyone can have the access to i.e. the debit card number, the CVV number at the back and the name of the holder. Now in the next step bank sends an OTP i.e. a onetime password which is to be used one per transaction or valid for a certain time span that is very short. This password is sent so as to keep the security system safe and dynamic, but, what if someone is able to read it through an external application? He/she can do a lot of damage to the pocket of the user to which the card belongs. SMS theft is a phenomenon in which all the SMSs that are received or sent through a mobile phone device are stolen by some malicious user or a spy. This in the hacking world is called as information leak attack. In this work we are trying to build a secure environment for the SMS transactions that are being stolen especially from the devices using the open source android operating system. The android operating system is vulnerable obviously because of its open source nature and thus we are trying to build a concept in the form of a sample application to illustrate the vulnerabilities as well as the methods to curb those vulnerabilities.

Table of Contents

List of Figures and Tables

CHAPTER 1

INTRODUCTION

A computer or a cyber-crime refers to a crime or a malicious act of causing harm to an individual or property of an individual by taking the aid of a computer or a network of computers. Life of almost most of the people in this world is dependent on computers and so on the internet. When people share their lives i.e. their personal information on the network or keep it in the storage of a device then there is a vulnerability of theft of information. One such attack is the information leak attack.

As the name suggests information leak attack refers to all the attacks related to theft of information from a device or by intercepting the information on a network. Information theft is very dangerous and can cause a lot of damage and disturbance if applied at a sensitive site such as bank transactions.

Suppose, a transaction took place between a bank and a customer online, which means crucial data transfer from one node to another. Now, if someone is able to leak this information while it is on its way or afterwards at the client's device then he/she may be able to know all the credentials of the customer and so will be able to access the bank account of the corresponding customer.

Information leak attack can also be used to hamper privacy. Communication can be leaked and all the personal information can thus be known. Moreover, all the browsing details of a person can be leaked and this is what google is doing.

1.1 CYBER SECURITY

To protect the system against these cyber-crimes one need to protect its systems as well as its network. This phenomenon is called as cyber security.

There are certain questions which need to be answered before actually securing the system:

- What information should be protected?
- What are the risks to the information and how much risk can be accepted?
- What measures are required?
- Do the security measures work?
- Which network is concerned?

- What protocols do the network uses?
- How strong is the firewall?

1. What information should be protected?

The information which should be protected can be something which is confidential and should have a limited access. Moreover the information which is personal like pictures files etc. should also be protected

2. What are the risks and how much is acceptable?

The risk or vulnerability involved in a system is dependent upon the security deployed as well as the criticality of information which is stored in a system. One can or one should have backup of the information which one is storing in a system. The acceptable limit of risks must be set in order to be prepared for it.

3. What measures are required?

Here comes the role of mitigation tactics which should be deployed in case of any threat to the information as well as the preventive measures that should be taken care of so that the information should not be vulnerable to any attack.

4. Do the security measures work?

Now after deploying the security measures it should be checked or validated if their existence is any fruitful to us. If the measures which are deployed, in order to protect the information are not able to protect it from the security threats than they are of no use.

5. Which network is of concern?

The network architecture or the machine framework which is worked upon is equally important. This information helps in building the security measures in accordance to the network vulnerabilities.

6. What protocol do the network uses?

The protocols are again very much helpful as some are stable and secure but compromises on speed as TCP on the other hand some are fast but not stable as UDP. So, the correct knowledge of network protocols should be known so that one can design the security model accordingly.

7. How strong is the firewall?

Now, firewall is a term that can be defined as the shield which is required by a system or a network to protect itself. Firewall helps in detecting and preventing intrusion attacks into a system or a network.

1.1.1 Deployment of cyber security

Cyber security can be deployed into various levels i.e. from a small system to a large network. Before studying the security measures to be deployed one should know about the threats or vulnerabilities at various levels.

1.1.1.1 Vulnerability

Vulnerability can be defined as the shortcomings that lead threats to turn into attacks. Various types of systems that can be vulnerable are:

1. Financial systems
2. Consumer IT products
3. Aviation
4. Big corporates
5. Government

1. Financial systems: Big banks which store their data on large servers for ease of access to them as well as to legitimate customers through online media are vulnerable to attacks as here it's about big money.

2. Consumer IT products: Personal information of people is also a very valuable source and thus is vulnerable to hacking. Many cases of Trojan intruding into personal information of people have come across in which the intruder or the attacker either spy on the person or just steal his or her personal information for malicious use.

3. Aviation: The aviation industry is the most vulnerable targets of terrorists. The 9/11 incidence where two airplanes crashed the WTC they were hacked by the terrorist organization Al-Qaeda. So aviation industry needs to be protected from intrusion.

4. Big Corporates: Big corporates have data that is crucial and secret for the overall functioning and growth of that company. Big IT companies has tons of codes which are to be kept confidentially and if leaked than they could just shatter.

5. Government: Governments of various nations have various secret defence as well as social reforms policies which are to be disclosed at a particular point of time. Terrorists or communal rioters may take advantage of this secret information by stealing it before-hand. Moreover the rival nation's military can also take advantage of situations like this.

1.1.1.2 Preventive measures

There are many preventive measures which can be taken as a step against cyber threats. Some of them are:

1. Firewall protection
2. Anti-virus
3. Anti-spyware
4. Limited sharing of information on the internet.
5. Not interacting with strangers over the internet.
6. Not allowing physical access of personal IT devices to strangers e.g. PC, PDA, Smart phone etc.

1.2 INFORMATION THEFT ATTACKS

There are two types of information leak attacks and this classification has been done on the basis of the location where the information is leaked.

1. Information theft over a network or communication media.

Here the information is stolen in between the nodes i.e. when some node say server transmits some data on the channel and the client is waiting to receive it then the information is intercepted on the way.[40]

2. Information theft at a particular site.

Now the second type of information leak can be performed on particular sites such as a standalone mobile device or a computer system where various crucial files are saved.

Information theft over a network has been exemplified with an illustration of ARP spoofing and a preventive measure has also been suggested by me, which has been discussed in the next section. Information theft at a particular site has been discussed in context to SMS theft attack in android devices further in this report.

1.3 ARP SPOOFING IN COMPUTER NETWORKS

In computer networks the man in the middle attack takes place with the help of spoofing i.e. forging the identity or we may say pretending what one is not. Suppose there is a bank transaction to be made and a website for a bank is opened say xyzbank.com. Now here there can be two things i.e. either the website which is opened in a genuine one or it may be a spoofed one, in both the cases the transaction will happen but only one transaction will be legitimate to the bank or the customer. In the other transaction the money or the critical details or both may land up in wrong hands i.e. the man in the middle of the customer and the bank. This MITM attack can be performed with the help of spoofing. Now spoofing can be of many types and in the next section we'll discuss in brief about different types of spoofing.

1.3.1 Address Resolution Protocol

ARP (Address Resolution Protocol) is a technique in which MAC or hardware address of the target machine is resolved with the help of logical or IP address of the same. Within a network packets can't be transmitted with I.P. addresses, that is why one needs to have MAC or hardware addresses of the machines which are to be contacted. [8]

The whole process works like this:

1. The machine which intends to send a packet to another machine broadcasts the packet with the I.P. address of the target machine over the Data Link Layer.
2. Now every machine within the network receives the message and checks it.
3. Now if the packet is relevant to them they would keep it but if not they discard it. When the machines check the I.P. address of the target machine whose MAC address is to be resolved they match it with their own I.P. address.
4. The machine which finds its own I.P. address in the target I.P. block first updates its ARP cache with the corresponding MAC address of the sender and then reply a unicast packet to the sender with its own MAC address.
5. Now the sender machine after receiving the ARP reply first updates its ARP cache and then the intended packets are transmitted.

ARP cache or table of a terminal is the one which helps in storing the MAC addresses corresponding to every IP with in the network with which a terminal has interacted.

This helps in efficient utilization of the network resources as well as saves time because every time the address is not resolved and directly referred from the table itself. [31] Figure 1 shows a sample ARP Table.

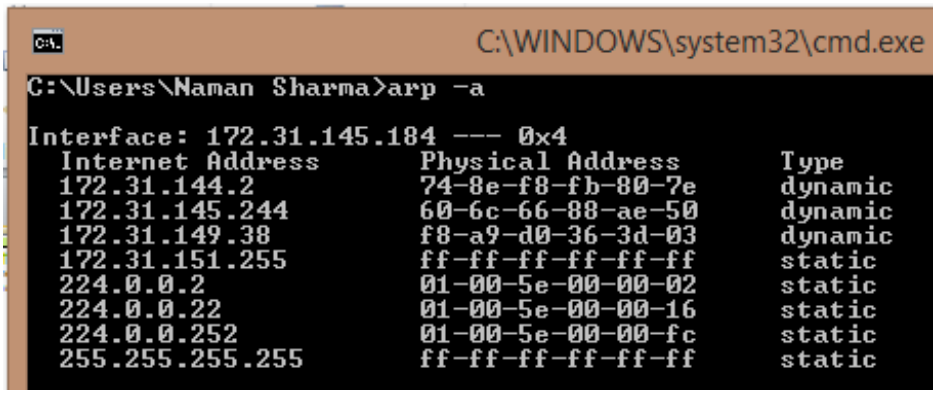


Figure 1 ARP Table Snapshot

Now, this ARP table which maintains the record of all the terminals in the network i.e. the I.P. addresses of every terminal and their corresponding MAC addresses is very much prone to attacks. As this table decides to which terminal the data has to be sent, so if someone tampers with it and change the corresponding MAC address the data can be stolen within the network very easily. This act of tampering with the ARP table to spoof or to duplicate the MAC addresses is called ARP spoofing. Figure 2 shows the pictorial representation of the algorithm.

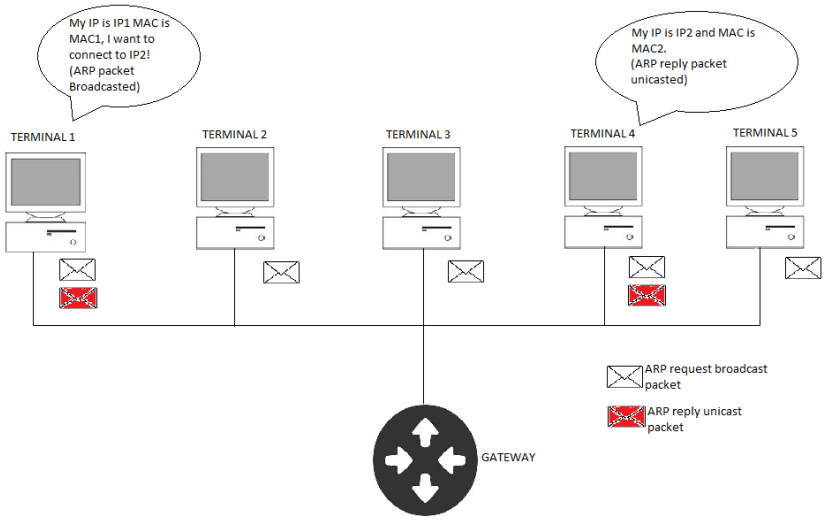


Figure 2 ARP- graphical illustration

1.3.2 ARP Spoofing algorithm in LAN

Man in the middle attack can be implemented very easily by ARP Spoofing. As this work is illegal so it can only be implemented and tested in a dummy network.[22]

This spoofing of MAC addresses can sound very naïve technique of spoofing but it can be harmful at the same time. One of the most prevalent examples of this type of spoofing is M.I.T.M. (Man in the middle) attack. If any malicious user has access to the network then he/she can very easily attack the network and very easily steal information which may be confidential or crucial.[25]

Steps to perform M.I.T.M. attack through ARP Spoofing:

1. First use a utility called "Wireshark" which can see the motion of every packet which is flowing through the network. Now with this utility ARP reply packets can be traced to any terminal or to the specific terminal to be poisoned.
2. ARP packet has four fields in it: Sender's MAC and I.P. address as well as that of the receiver's. [45]
3. Now if attacker wants to sniff every packet that will be transmitted between the two machines then he can send a fake ARP packet to the message sender which will update the MAC address corresponding to the receiver's I.P. address and that address will be of the attacker's machine, then it can forward those packets to the destination itself. [37]

Now to curb this vulnerability in the ARP, I proposed some changes in the ARP itself by introducing a Centralized ARP Server.

1.3.3 Centralized ARP server to detect/prevent MITM attacks

Generally what happens in the ARP is that the request is always a broadcast and the reply is unicast. Now, if every ARP reply packet is also broadcasted over the network and an independent ARP server is introduced which is not involved in any type of communication but receives every ARP broadcast packet and every ARP reply packet then the problem may be minimized.[23]

Here the CAS i.e. Centralized ARP server keeps the record of every ARP packet that is moving in the network. Now if the terminal A with IP address say IP1 wants to communicate with another terminal B with IP address IP2 then the terminal A will broadcast an ARP packet which is received and checked by every terminal but the CAS will save that packet i.e. the IP-MAC couple of terminal A. Afterwards when the terminal B will create an ARP packet to reply terminal B then also that packet will be

broadcasted and again the CAS will store the IP-MAC binding of terminal B and also every other terminal will save that mapping. Now this mapping will be stored in a file in every terminal along with the ARP cache and no other packet shall be received to overwrite it.

This change will make the ARP algorithm more efficient as when any two nodes are communicating with each other than every other node in the network will get to know the IP to MAC mapping of those terminals and if any third terminal wants to communicate with one of them i.e. either A or B than they can easily find their MAC address in their cache and communication will be faster. Moreover if every terminal has the IP-MAC mapping of some terminal that can be easily cross verified if tampered with.

1.3.3.1 Flowchart of the proposed system

Figure 3 shows the flow of actions for the ARP spoof prevention algorithm.

1. A packet is received by the terminal, then its target address is checked by the receiver.
2. If the target address is a broadcast address, then the packet is accepted at this stage otherwise it is discarded from here only.
3. Then if it is considered that the target address is the broadcast address its mapping is verified with the CAS i.e. the centralized ARP server.
4. If the mapping is duplicate then the packet is discarded straight away, else if the mapping is new then the CAS and the ARP cache is updated.

A pseudo code of the algorithm corresponding to the flowchart below has been provided in the next section.

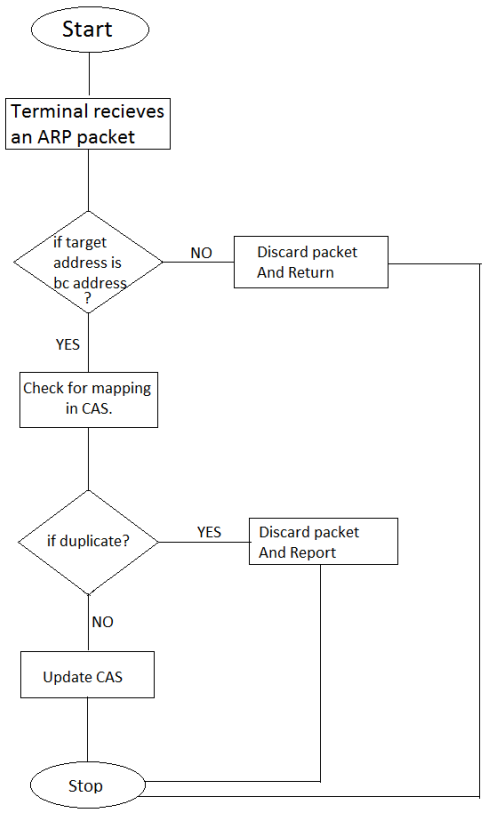


Figure 3 Flowchart for ARP functioning with CAS

1.3.3.2 Algorithm for the functioning of ARP with CAS

Here in this algorithm the working of the modified ARP along with the CAS is shown.

1. A terminal receives an ARP packet.
2. It checks the target MAC address of the packet

```
if(broadcast address)
{
        then check for mapping in CAS.
        if(duplication detected)
        {
                report;
                return;
        }
        elseif(not in CAS)
        {
                update CAS;
```

```
        }
    }
    elseif(not a broadcast address)
    {
        discard;
        return;
    }
```

3. Now after the updating in the CAS the mapping information is stored in the ARP caches of the terminals as well as in the permanent ARP file.

Now, here the duplication means, the IP-MAC mapping which terminal has received it compares the same with every entry in the table, if the same IP address is mapped to some different MAC then duplication will be detected. Figure 4 shows the pictorial representation of the working of the ARP with the CAS i.e. the Centralized ARP Server. Here the ARP reply packets are also being broadcasted and the CAS also receives them so that it could check for any duplicate mapping.

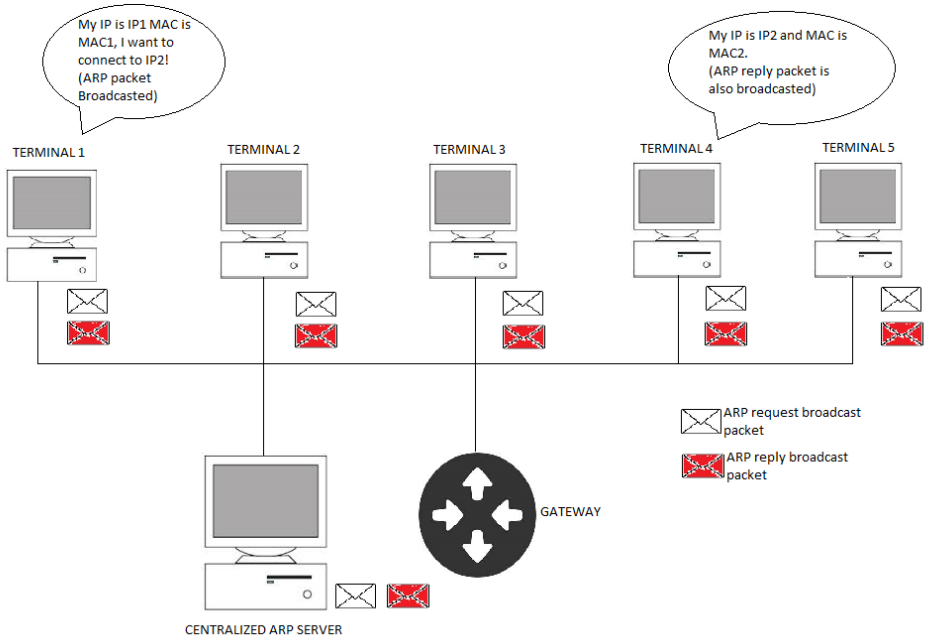


Figure 4 Modified ARP with CAS

After accomplishing the ARP spoofing prevention algorithm, I tried to lay stress towards the SMS leak attacks which takes place when the SMSs which are stored in the data base of a mobile phone device are leaked with the help of some exploits. Now-a-days the most vulnerable and used operating system is the open source android

operating system. In the next few section vulnerabilities which causes the leak of SMSs in a mobile phone device using the android operating system will be discussed.

1.4 SMS THEFT ATTACK ON ANDROID DEVICES AT A GLANCE

If all the classes of android APIs are studied thoroughly than there are vulnerabilities that can be exploited out of it. Anyone can write a high level code and thus play along. Whenever an application is downloaded from the google play store for instance then it seeks for rather notifies the permissions which are required by the application. Now these applications can access any of the applications whose access permissions has been accepted by the user who downloaded the application. Now these applications can even have the access to the whole device and the crucial data or private data residing on it. By cleverly using the inbuilt java classes in android APIs any android device can be fully controlled. For a better understanding of the SMS leak attack in android device one need to have the basic understanding of the architecture and related APIs of the android system.[4]

There has been a lot of increment in the malware content among the android devices that could be detected and recorded in the database of the McAfee antivirus corporation from Intel. The database of detected malware on android devices was maintained from the year 2004 to 2012 and a tremendous increase in the malware content was seen in the year 2012. Figure 5 shows the graph for the rise of android malware over the years.

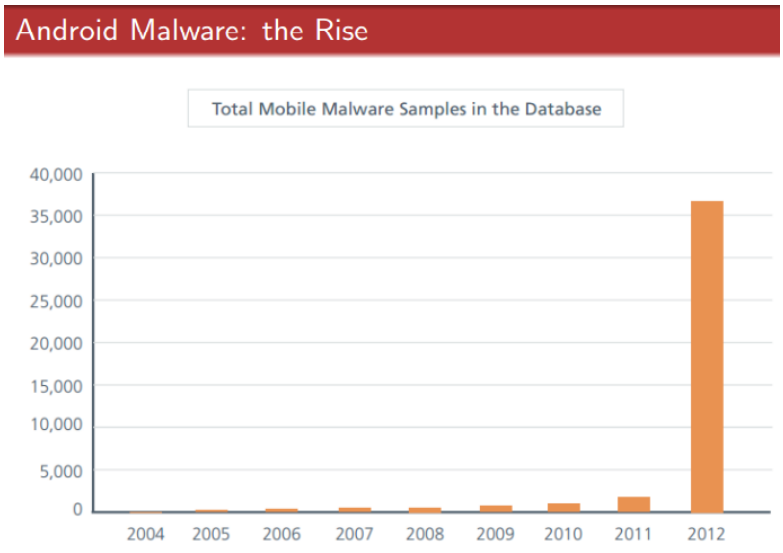


Figure 5 Graph showing rise in Android malware

1.4.1 Android open source operating system.

Expecting a shipment of 1 billion Android devices in 2017, cyber criminals have naturally extended their vicious activities towards Google's mobile operating system: threat researchers are reporting an alarming increase of detected Android malware from 2012 to 2013. [3] Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies. Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android. The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008. On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 Jelly Bean. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance. The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.[20]

1.4.2 Android architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.
Android operating system works with the help of Java APIs written for android so they now become android APIs. These java classes interact with the Linux kernel level code written in C with the help of JNI i.e. the java native interface. Figure 6 shows the pictorial representation of the android architecture. At the base it is the Linux kernel, then above this there are system libraries, then the application framework and at the top most layer comes the applications and the widgets.[20]

1. Linux kernel – This is the base of the Android operating system and is responsible for all the core functioning of the android device as it has all the drivers running in it. It includes all the hardware drivers such as camera, audio, memory, power, binders etc.

2. Libraries – Now they are the files which are responsible for the application development task in android devices, without them no application could work as importing these makes the work easier as well as possible. Libraries are of two types i.e. static or compile time libraries and dynamic or runtime libraries. Runtime libraries are responsible for the virtual machine o which the android program runs i.e. the Dalvik Virtual Machine.

3. Application framework – Application framework provides the basic classes required to access various functionalities of the android device. They are written in java and interacts with the base libraries written in C with the help of JNI i.e. Java Native Interface.

4. Applications and widgets – At the top most level there are applications and widgets which are used to access and control various functionalities of the android device with a user interface which interacts with the APIs. There are about 1.5 million android apps present in the market i.e. on google play store and the rest which are not even registered on the play store also exists.

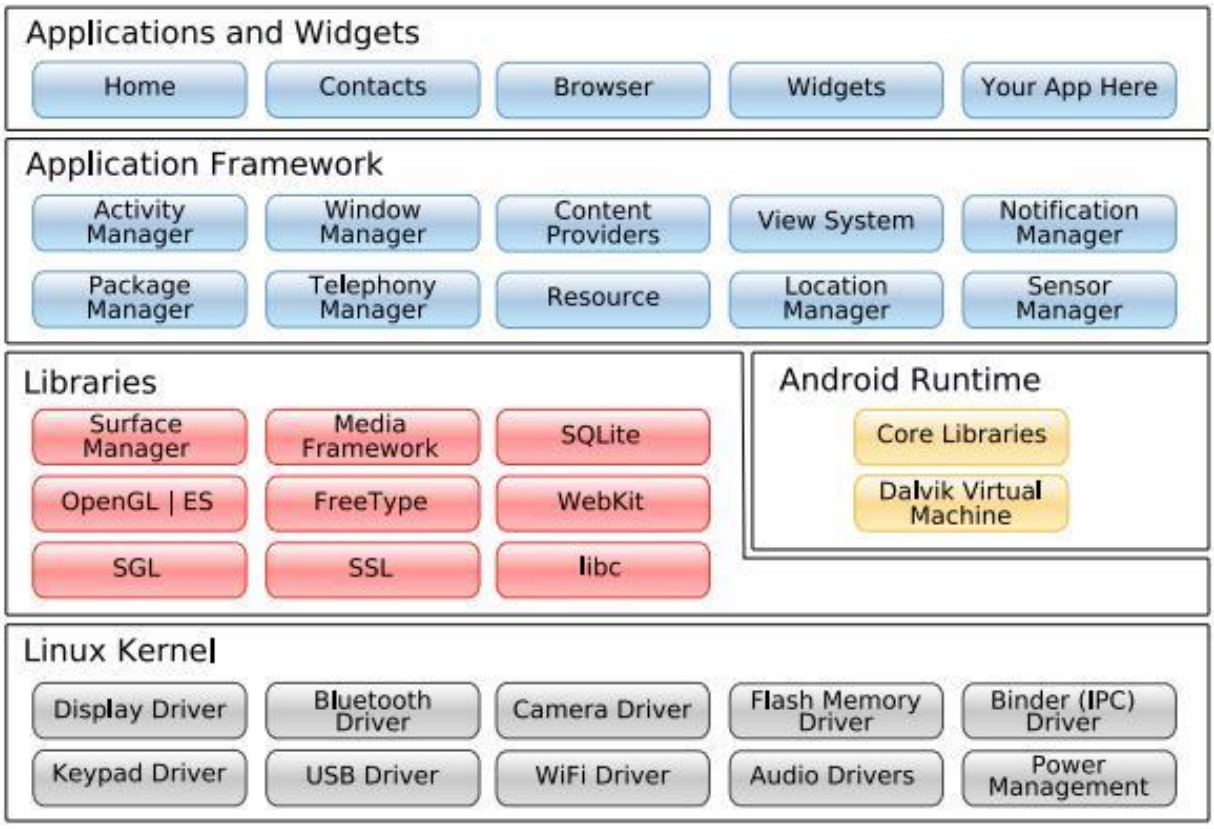


Figure 6 Android architecture

1.5 SHORT MESSAGING SERVICE (SMS) IN ANDROID DEVICES

In Android, SmsManager API or devices Built-in SMS application can be used to send SMS's.[42]

Telephony class is used to access the messages when the Intent.SmsIntents.SMS_RECEIVED_ACTION is set.

```
Telephony.Sms.Intents.SMS_RECEIVED_ACTION
```

Here the following conditioned is checked:

```
if(intent.getAction().equals(Telephony.Sms.Intents.SMS_RECEIVED_ACTION))
```

After this condition is satisfied any message can be received and used to show anywhere.

SmsManager API

```
SmsManager smsManager = SmsManager.getDefault();

smsManager.sendTextMessage("phoneNo", null, "sms message",
null, null);
```

Built-in SMS application

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);

sendIntent.putExtra("sms_body", "default content");

sendIntent.setType("vnd.android-dir/mms-sms");

startActivity(sendIntent);
```

SEND_SMS permission

```
<uses-permission android:name="android.permission.SEND_SMS" />
```

Apart from the above method, there are few other important functions available in SmsManager class. These methods are listed below – [5]

Table 1 Function of SMS API in android

Sr.No.	Method & Description
1	**ArrayList<String> divideMessage(String text)** This method divides a message text into several fragments, none bigger than the maximum SMS message size.
2	**static SmsManager getDefault()** This method is used to get the default instance of the SmsManager
3	**void sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, PendingIntent sentIntent, PendingIntent deliveryIntent)** This method is used to send a data based SMS to a specific application port.
4	**void sendMultipartTextMessage(String destinationAddress, String scAddress, ArrayList<String> parts, ArrayList<PendingIntent> sentIntents, ArrayList<PendingIntent> deliveryIntents)** Send a multi-part text based SMS.
5	**void sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent)** Send a text based SMS.

As android is made up of Java APIs working on Linux kernel, the open source nature of the operating system as proved to be a boon and a bane at the same time. Android devices are prone to many attacks because of the open source nature of the OS installed on them them. Privacy over android devices is a cumbersome task because of the open source nature. If the Apple's iOS and Blackberry's OSX is looked upon in terms of security then Android has a lot to improve, but again if evolution is the consideration then these operating systems are far behind android. At cheaper prices an android user gets access to many a functionality which can only be dreamt of by an iOS user. Now to steal messages from an android device one requires a basic knowledge of some of the objects and classes which are related to the inter application communication of the android native applications which are installed from day zero. [24]

1.6 MOTIVATION BEHIND THIS WORK

Today's online world is very much prone to the critical data theft attack, there is a ton of malware available which is hampering the personal information as well as the critical information such as bank account details, service subscription details etc. The concept of safe surfing has lost its meaning and every device which is connected to a network or stand alone is prone to some or the other kind of attack. Now, how and why these attacks are happening? People now-days are so busy that they don't have time for life and thus they are ready to leverage everything which makes their lives easy and convenient. People using the internet ten years ago were only dependent on technology in certain aspects like ease of communication calculation entertainment etc. but people using the internet now-a-days are dependent on it to a very large extent. E-shopping, e-banking, e-ticketing, e-socializing, e-sharing have made the lives of people so much dependent over the internet that they are ready to leverage every single detail of their lives. They share personal stuff on social media like Facebook, Instagram etc. They share their bank details or the credit card details with e-portals and payment gateways. Now in this era where the internet has taken all means of conveyance the vulnerability of the malicious activities concerning the use of internet by people has risen up to the brim and thus some mechanism need to be devised to control the security breaches into secure systems so that the "e-life" of the people could become secure. One of the most vulnerable security breaches that are

taking place now-a-days is in the mobile devices running the android open source operating system. As the android operating system is open source and build up on Java APIs over Linux kernel the vulnerabilities are many and it is a very cumbersome task to identify all the vulnerabilities and to curb them all. But still if some more crucial areas are worked upon then it may become possible to safeguard an android device against attacks. One such vulnerability in android devices is because of SMS theft attack. Now one may think that SMSs now-a-days are seldom used by the people as there are several messaging apps that are being used has replaced it, but what is being forgotten here is that the application which are being used as an alternate to SMS are not legitimate enough as SMS. No bank or service subscription ever uses "Whatsapp" or "Hike" to send confirmation or transactional messages. SMS services have their importance till date as the most critical information i.e. bank transactional messages, one time passwords, verification codes etc. all are received by us or sent to us by the service provider through SMS itself. So, it is required that a solution should be provided in the form of an application which should prevent any theft of the messages or should at-least detect which applications have the permissions to access SMS of the device.

1.7 OBJECTIVE OF THIS WORK

After counting upon all the vulnerabilities in android device regarding SMS, an application is required which should tell, which of the applications are in the grey area of permissions for accessing the SMSs on the device when they don't even require them for the functioning of their core features. A good android application is one which takes system permissions which are minimal and necessary for the functioning of core features of the same. For making such a utility application we need to identify classes in the Android API which are responsible to provide access to the app permissions of the corresponding device.

1.8 ORGANIZATION OF THESIS

Chapter two of this thesis report consists of the literature that has been surveyed, most found on SMS in android devices, SMS theft vulnerability in android devices and android open source operating system at a glance. In the Chapter three of the report, problem has been formulated on SMS theft attacks on Android devices and some

reasons have been listed for the same. In android devices SMS theft attack has been shown by using the android APIs. Finally in the last chapter i.e. chapter four some solutions has been proposed for the problem elaborated in chapter three i.e. theft attacks in mobile devices. For mobile devices using android open source operating system, an application has been made i.e. a secured SMS receiver which helps in receiving the SMSs on a mobile device and keep them in a secure file. In this document the terms information (SMS) theft or leak will be used interchangeably.

CHAPTER 2

LITERATURE SURVEY

Many research works have been surveyed, to find the vulnerabilities in the android operating system, regarding the privacy thefts and the SMS shortcomings in the android APIs. A whole lot of experience of gaining such vast knowledge about the android operating system and the SMS API corresponding to that has been listed here in this section.

Felt A.P. et al. (2011) laid stress on the permissions that the third party application demands FOR when the installation process is taking place. Whenever one downloads and install an application from the Google's App Store than it gets a notification on the screen about the permissions or it may say the system rights which are required by a particular application when it gets installed over the respective device and which are required for its full fledge working. Now one should be very careful while installing any application on an android device, let's say an application is installed which works as a torch i.e. it uses the flash of the device's camera and while installation it asks for the permission for accessing the contacts and messages, this means that the application comes under the fishy zone and wants to access user's data by providing a service which is worthless in front of the leverage given by the user.[2]

Te-En W. et al. (2012) said that "SMS is a common and basic functionality in traditional mobile and smartphone. Hence, it often can be used for many purposes which include shopping, banking, nail, and other services. For example, banks use mobile and smartphone for payment rely on two factor authentication. The payment credential document texts in the smartphone which makes smartphone become the target for attackers. Otherwise, Premium-Rate SMS is also a critical privacy issue. One often receives the legal premium-rate SMS messages which deliver the valuable and useful contents such as stock quotes and other technical supports or services. However, the charge of premium-rate SMS messages usually makes the sender has a heavy phone bill. Not only Premium-rate SMS but also Premium-rate calls which can cost a lot per minute, and premium-rate SMS messages can cost a lot per message.

Thus, it may be known that the SMS misuse will make phone owners lose assets, rights and interests.”[41]

Vargas R.J.C. et al. (2012) concluded that he Android devices needs to be hardened for business environments, adding security controls and configurations that meet the business security policies. The vulnerabilities in Android can compromise the information stored in the device, so it is necessary to add encryption systems, better permissions management, host firewall, log activities and disable unnecessary services. This paper shows a guide to add security controls to the most spread version of this operating system. These controls help to avoid attacks that get control of the Android system, like the privilege escalation, and reduce the information an attacker can get during a system scan or network sniffing. It is necessary to design specific policies for the different environments where these devices are used, and it will be covered in future works. It is also necessary to design pentest schemas and measure these security controls in the future work.[33]

Erturk E. (2013) found that recent types of Android malware resemble their desktop based predecessors rather than being genuinely created for a specific operating system. Malware and adware are better understood in the context of their commercial objectives and countries of origin (in particular China and Russia) as well as distribution. Therefore the analysis and recommendations should not be shaped merely by program logic because there are supplementary geographic and commercial factors. On one hand, many free apps rely on advertising to support their development. On the other hand, as can be seen from the sample malware, certain apps have crossed the line from merely displaying ads to pushing (or forcing) products to the user, harvesting private data for future use (e.g. spam or other use), and even extracting fraudulent revenues. It is possible for hackers to rent premium rate numbers anonymously (for generating dialing or SMS fraud) in Russia and other Eastern European countries whereas this is not possible in many other countries. Secondary app markets, where many malware have been found, seem to have grown also as a result of language factors in the case of China and other East Asian countries. The official Android Market (Google Play) is still blocked as a result of government restrictions in China. This has given rise to many secondary local app markets there.[10]

Jeter L. et al. (2013) reviewed and concluded that solutions for Android users currently exist from a number of vendors, including Lookout and Droid Security, who are both mobile-focused companies, and F-Secure, Norton, and McAfee, who each have large presences in the desktop anti-malware market. Although potentially effective, anti-malware software is traditionally a reactionary solution, with the average time of 48 days from a new attack's release until it is discovered and included in future virus signature patches. An alternative manner of virus detection involves the review of behavior signatures, however this approach is much more resource-intensive and its effectiveness is reduced due to the limited resources provided to applications by the Android framework. It should also be noted that cloud-based and desktop-based anti-malware scanning solutions are also being researched by various groups to alleviate the burden placed on portable devices' resources.[18]

Savola R.M. et al. (2013) analysed security objectives on the basis of risk analysis results. The basic building blocks for security objectives are authentication and authorisation, integrity, and confidentiality controls. Access control plays an important role in the target system. Moreover, considerations of software and system quality are important. Many of the risks identified can be mitigated through assurance of configuration correctness.[35]

Ahmed M.S. et al. (2013) agreed that iOS are more advantage compared to Android operating System in term of security based on comparison that have made. However, there are few basic security points to keep our data safe on the respective mobile device are:

• Always update your Smartphone OS, irrespective of it being an Android or an iOS, whenever any application patches or OS upgrades are released. • If the device is being used by a stranger, use a Passcode to lock your device in order to avoid data leakage. • Do not jail-break, root, or modify the OS files. • Install an antivirus and firewall software to detect and stop any infection. • Install device-tracking applications to find the phone whenever it is lost or stolen. • Regularly backup or synchronize your settings and other personal information in order to avoid the loss of data due to theft. • Try to learn about the application's reputation before installing it. • Control the types

of data that can be accessed via mobile devices in order to determine your exposure should a device be compromised. • Don't let arbitrary third-party applications run on your device platforms; these applications are a go-to tool for hackers. • Utilize Mobile Device Management software to create an encrypted password-protected sandbox for sensitive data and enforce device-side technical policies.[20]

Hu W. et al. (2014) presented a novel approach to detect APP-Repackaging Android malware. Their method first extracted API calling sequences from a given Android application and then constructed the method invocation graph. After generating sub-graphs based on the method invocation graph, they calculated each subgraph's threat score according to the sensitive APIs invoked in the sub-graph. Finally, they labelled the sub-graph whose score exceeds the given pre-defined threshold as malicious. Their system evaluation showed that the approach they've proposed has a good performance on APP-Repackaging Android malware. The real-world case study revealed the power of detecting Android malware in practical environment without any training sample or signature. The advantage of the approach is that it only utilizes static analysis that is lightweight. [44]

Allix K. et al. (2014) concluded that the recent and steady rise of Android malware over the past four years has lead to a rapidly growing automation in the malware creation process. Due to the specific nature of development of Android applications, important artifacts leak out and can provide some insights about their creators. We have analyzed the available data through this perspective. For our large-scale study, we have considered over 500,000 Android applications, which included both benign applications and malware.[16]

Meshram P. D. et al. (2014) said that on android smartphones, lots of private information of users is stored in databases. Android relies on the Sandbox to protect codes and data of an app from other apps, whilst it offers the ContentProviders to share databases as necessary. Client apps intending to share databases of other apps need to present the same URIs as the database identifiers and database information as well as permissions required by server apps. Attackers trying to attempt any

unauthorized and illegitimate access can use reverse engineering to extract all of the necessary information. Proguard does not guarantee prevention of information extraction. In addition, assigning permissions to control accesses is not mandatory, which leads to security loopholes. In other words, attackers can easily access private information on Android Smartphones with no particularly complicated analysis or alteration or forgery of server apps.[29]

Savola R. M. (2014) proposed a risk-driven security engineering and metrics development assumes the development of security objectives and controls based on the prioritized risks. They analysed security objectives for an Android platform utilized for a public safety and security of mobile network based on iterative industrial risk analysis results. There were many interdependencies between the original risks. Furthermore, security objectives and controls show a different pattern of interdependencies. The original risk analysis results should be preserved in further actions to offer appropriate emphasis.[34]

Yubo S. et al. (2014) illustrated how to exploit the SMS as an attack vector to deploy malware on Android system. They began by providing a detailed description of RSMT messages, why they can be selected as the exploit carrier and hide the malware behaviour. They made use of the OpenBTS combined with USRP for the sending of RSMT messages. They then investigated the security model of Android system. So me defects of Android system are used to realize our target. Furthermore, we deploy real scene attack on some smartphones. [39]

Guo-Hong S. (2014) analysed the Android program design and development for audio/video file procurement, based on the architecture introduction of Android platform. It mainly applies OpenCORE and MediaRecorder class. After the test, their program could successfully realize the procurement and playing of the audio/video files with fast execution speed and good user experience. [38]

Sugiono E. et al. (2014) Over 10 Android major releases (41 minor releases) until 28th July 2014, 16,7% of 455 Android reported vulnerability originated from Android. From those records, we found that Android Team is getting faster in patching vulnerability because Android TTP showed a declining trend. We also found

that Android reported vulnerability showed a declining trend. Based on that, we can say that Android security is getting better.[11]

Mulliner C. et al. (2014) concluded, with increasing demand of stronger authentication mechanisms, online services adopted SMS-based one-time passwords to mitigate phishing and other attacks. Services adopting SMS OTPs are not limited to banking and other financial services, but include email providers and popular online games. Lately, SMS OTP have come under heavy attack, especially by mobile phone trojans that are specifically designed to intercept and forward OTP authentication and authorization credentials to criminals. We presented the virtual dedicated channel, a solution that secures SMSbased OTPs against SMS stealing mobile phone trojans. Our solution is completely backwards compatible and only requires minimal changes on the mobile phone side. Thus, our solution is easy to deploy since it leaves the infrastructure at the service provider and the OTP message format unchanged. SMS-based OTP is one of the most user friendly multi-factor authentication mechanisms today that does not require an additional device. We believe our solution provides the means to secure SMS OTPs against attacks and thus helps to prevent online account theft and fraud.[7]

Kim D. et al. (2014) said, recently various solutions have been launched to prevent the damage resulted from the smishing that causes overcharging via micropayments by intercepting SMS. However, there is a limitation to solve the problem only on the application layer due to the characteristic of android platform. For this reason, they proposed an idea to deal with the problem by means of the modification of the framework and the native SMS app. They demonstrated the idea based on the result of the analysis on the framework by implementing them. In addition, they considered the rooted environment. Therefore, they also suggested an idea to verify the app with its public key of the certificate on the framework layer and get users notified of the compromised app. This can guarantee that the interception of important SMS that is used for the micropayments is completely prevented without affecting any other third party apps related to SMS. Furthermore, it is expected that the method to adjust the order of receivers on the framework layer is available to diverse cases if there is a problem caused by priority order.[9]

Chan P. et al. (2014) In this paper, they proposed a static mechanism for detection of Android applications by extracting permissions and API calls as feature set. They applied different machine learning techniques in our approach. The experimental results have shown that the method can reach an accuracy rate of more than 90%. And they get the conclusion that the feature set "Permissions + API" can be better for classification than the feature set only contains "Permissions" generally. It may be due to the complementary of API calls and permissions. [28]

Alam M. et al. (2014) proposed a context-aware multiagent based framework for monitoring behaviour of Android devices based on JADE 4.2. They allow collecting multiple data features to compute feature vectors using Action agents that are launched based on context on Android devices. These features are gathered to detect if Android devices have been infected by malware. This detection is performed by Analysis agents of server side that run pre-trained machine learning classifiers. They conducted experiments to measure the CPU and network usage patterns exhibited by our system based on measuring battery contexts. They also performed experiments to launch Action agents only when certain network or GPS conditions are met. The Profile agents on the Android devices send the data collected to the Profile Service Agent on the server for analysis. [21]

Teufl P., et al. (2014) concluded that the special architecture of the Android platform facilitates the implementation of SMS catcher and sniffers even on non-rooted smartphones. Recent examples of malware that attempt to compromise the security of e-banking applications emphasize the need for appropriate methods to detect SMS catchers and sniffers. In their paper they have proposed and analyzed possible detection methods that can be directly executed on mobile end-user devices. These detection methods have also been arranged in a workflow. This workflow puts rock-solid methods, such as application whitelists and manifest checks at the beginning of the detection chain. These methods are intended to filter out as many benign applications as possible. Remaining applications that potentially contain malicious code are then analysed with more complex static code-analysis techniques. [32]

X. Li, et al. (2014) As smart mobile devices integrate more resources, the physical boundary among resources disappears. Therefore critical resources are exposed to

resource-abusing apps. Existing solutions fail to balance security and usability. In this paper, they proposed RVL which can systematically protect resources through light-weight resource virtualization. RVL defines profiles containing a set of virtual resources to confine Android apps. Profiles with virtual resources alleviate users' burden in weighing permissions requested by untrusted Android apps, and thus improve usability; isolation between profiles ensures stronger protection. They showed the compatibility, effectiveness and performance through evaluation on real-world apps. [47]

Zhang Y. et al. (2014) This paper presents VetDroid, the first approach to perform accurate permission use analysis to vet undesirable behaviours. To construct permission use behaviours, this paper proposes a systematic framework that completely identifies explicit and implicit permission use points with accurate permission information. VetDroid is shown to be able to clearly reconstruct malicious behaviours of real-world apps to ease malware analysis. It can also assist in finding information leaks, analysing fine-grained causes of information leaks, and detecting subtle vulnerabilities in regular apps. In all, VetDroid provides a better vehicle for analysing and examining Android apps, which brings benefits to malware analysis/detection, vulnerability analysis, and other related fields. [49]

Enck W. et al.(2015) gave a comprehensive review over the security aspects in android devices. They divided their work into three categories, firstly they considered the exploration of the issues which were uncovered till the present date version of android and also the malware advisories. Secondly, they considered the general coding failures which occur and then create the vulnerabilities of security breach. Thirdly, they found out the existing flaws in the security architecture of the android operating system and then related it to how these flaws could be misused.[43]

Jia P. et al. (2015) forward PriGuard model for protecting privacy information dynamically based on the research of RPC and function redirecting technology on Android platform, and focuses on how to establish the control strategy and how to control application's behavior dynamically. The research carried out has the following contributions: (1) Propose a scheme to control the application's behavior dynamically on Android; (2) Put forward a method establishing the behavior control strategy by

feature selection algorithm. (3) Implement the transformation from the passive detection to active defense for users` privacy information leaks of Android platform. Furthermore, the technical ideas PriGuard used also can be applied to malware detection and other aspects.[27]

Khandelwal A. et al. (2015) Android operating system is widely being used in mobile phones.in this paper we presented an overview of android architecture .We then looked at three areas namely privilege separation, permissions and code signing we can use these features to increase security .We then looked briefly at security risks, and some of the ways in which android facilitates this.[1]

Faruki P. et al. (2015) concluded that android is a core delivery platform providing ubiquitous services for connected smartphone paradigm, thus monetary gains have prompted malware authors to employ various attack vectors to target Android. Due to large increase in unique malware app signature(s) and limited capabilities within Android environment, signature based methods are not sufficient against unseen, cryptographic and transformed code. Researchers have proposed various behavioural approaches to guard the centralized app markets as malware authors are targeting easy to reach user online distribution mechanism. In this survey, we discussed Android security architecture and its issues, malware penetration and stealth techniques. They discussed static and dynamic approach for malware analysis and detection. Both approaches can be used separately, but each one has its own limitations. Static analysis can be thwarted by employing encryption and/or transformation techniques discussed. Dynamic analysis can be evaded by several anti-emulation techniques. They also covered prominent malware analysis and detection approaches and summarized in a table according to their goal, methodology and deployment. Summary shows there is not a single solution that addresses every issue. To tackle wide variety of new malware, a comprehensive evaluation framework incorporating robust staticand dynamic methods can be proposed on Android platform.[30]

Pak W. et al. (2015) Concluded that currently, it is impossible to achieve 100 % detection rate for malicious application with a static analysis method, whereas possible with our proposed approach if the attacker performs cyber-crimes with leaked private data. Furthermore, the damage can be minimized owing to the data

even though it is abused for cyber-crimes. In these days, the demarcation between trusted and malicious application has become increasingly blurred. Therefore, that it is hoped that this approach will help users to protect their private data more effectively.[46]

Li L. et al. (2015) proposed the same thing as the apps which requires more application permissions than they actually require and they also have an algorithm which analyses the applications and finds out if the access permissions asked by them doesn't falls in the grey area i.e. their application's permissions should not exceed their actual requirements.[19]

Hence many attempts have been made and published in order to prevent the theft of data from external sources be it in the computer networks or be it the excavation of memory of android devices. Till date no effective technique have been built or implemented which is safe, secure and efficient enough to suit the needs of the modern day mobile phone users having android open source operating system. Now some application is required at a native level for the android operating system in order to satisfy both the needs i.e. security as well as efficiency. In terms of security we need to have a robust application that should not miss any chances of attack through any vulnerability and by efficiency it means the application should be generous to the limited resources of the mobile device i.e. it should have minimal memory and CPU usage.

CHAPTER 3

PROBLEM FORMULATION

3.1 PROBLEM STATEMENT

This is an age where every individual is dependent on IT for his/her basic requirements. All the works of an individual i.e. from grocery shopping to important bank transactions which includes millions of rupees are accomplished with the help of the internet. Now when people are dependent so much on something they forget that what comes along with it and here the bad is considered. Everyone who uses the internet is so much dependent on it that it doesn't even realise that what is being mortgaged from him for this service. Whenever one installs a new software or subscribe to a new service online then there is always a service agreement which has to be read and then decided if one wants to accept it or reject it, but a very few of them actually read the service document and then accept it. Lives are moving so fast that people don't even think about their personal as well as professional lives being at stake by these online services. Now similarly when one installs an application on an android device than it is notified that a particular application will require permission to access some features of the respective device and one quickly grants them the permission to access all those features. This happens because people are so shallow thinkers that for their small needs they give away a large part of the privacy. This document refers to problems or vulnerabilities in the mobile devices running an android open source operating system which may cause the information leakage. Information theft in android devices can be in many ways but here SMS theft is considered. As android is made up of Java APIs working on Linux kernel, the open source nature of the operating system has proved to be a boon and a bane at the same time. Android devices are prone to many attacks because of the open source nature of them. Privacy over android devices is a cumbersome task because of these vulnerabilities. If the Apple's iOS and Blackberry's OSX is looked upon in terms of security then Android has a lot to improve, but again if evolution is the consideration then these operating systems are far behind android. At cheaper prices an android user gets access to many a functionality which can only be dreamt of by an iOS user. Now to leak messages from an android device one requires a basic knowledge of some of

the objects and classes which are related to the inter application communication of the android native applications which are installed from day zero.

3.1.1 SMS theft attack in android mobile phone devices

Here in this approach several in built APIs of the android operating system have been used to leak the messages (SMSs) through a parallel medium which are received by an android device. Now these inbox messages are very much useful and can contain crucial data which can be misused. For instance if a device receives an OTP i.e. a one-time password for registrations or verification for a service or from a bank then if the messages on that device could be read by some malicious user, then every single detail of the person using that device could be leaked. Now here are some classes and objects in the android and java API that could be used to leak all the messages very easily to an external file. [28]

3.1.2 Android specific java classes and objects used

Intents- They is the class of the android.content package. The code can send them to the Android system defining the components targeted. For example, via the startActivity() method defines that the intent should be used to start an activity. "intent" can contain data via a "Bundle".

Context- It's an abstract class whose implementation is provided by the Android system. Context allows access to application-specific resources and classes, as well as calls for application-level operations such as launching activities, broadcasting and receiving intents, etc.

Telephony Manager- Provides access to information about the telephony services on the device. Applications can use the methods in this class to determine telephony services and states, as well as to access some types of subscriber information. Applications can also register a listener to receive notification of telephony state changes.

BroadcastReceiver class- They simply respond to broadcast messages from other applications or from the system itself. These messages are sometimes called events or intents. For example, applications can also initiate broadcasts to let other applications

know that some data has been downloaded to the device and is available for them to use, so this class will take the control and will initiate appropriate action. [17]

3.2 ALGORITHM FOR SMS THEFT ATTACK IN ANDROID DEVICES

An algorithm has been devised to leak the SMSs in an android mobile phone device using the native Java classes.

3.2.1 Flowchart for the SMS theft attack

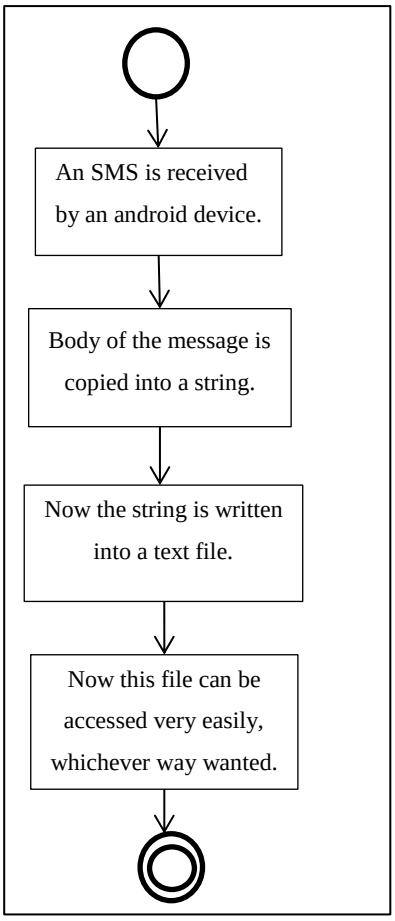


Figure 7 Flowchart for SMS theft algorithm

Figure 7 shows the working of the SMS theft algorithm in the form of a flow chart. Firstly a message is received by an android mobile phone device in the form of an SMS. Now using the basic android telephony attributes, the body of the message can be extracted very easily and thus can be accessed and copied into a string. String is a data type class in the native java API which helps in storing a cluster of characters in a particular sequence, what is fed in. Once a word a sentence or a character is fed to a string then it can be further extracted in any file that supports the standard asci

character recognition and also helpful in the display of the same. The file here we use has to be stored in an external file directory so that it could be easily accessible. Now, when the messages are stored in a file with *.txt format in the external file directory then it could be accessed or transported whichever way it suits or required.

3.2.2 Algorithm to leak the SMSs

Now, the algorithm that has been deployed in leaking the messages is quite simple and thus gives an idea about the level of security deployed by the android operating system. Moreover whenever an application is installed then the requested permissions are displayed on the device's screen so that the user should also know the capabilities of the application and realise about the vulnerability of breaching into his\her privacy, now it's the choice of the user to accept it or not and generally he\she does.

Step I – At first the application gains the access to the SMS receiver of the android phone so that it could read all the messages which are received by the device.

To acheive this, the following needs to be done:
1. An extra permission to receive the incoming SMS messages.
2. The SMS receiver is declared in the AndroidManifest.xml file.
3. Then an SMS receiver class is written to handle received messages
4. Decide if we have to show up the message in the default messaging client.

Adding receiver to Android Manifest

Add the sms receiver class to your manifest. If you want your app to handle the SMS message before any other app, then set the priority to the maximum: 999

Setting the maximum priority also comes with added responsibility. You can actually mess up other SMS messages that you dont want to handle from reaching the messaging app.

Here is a code snippet of the manifest.xml file.

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />

    <receiver android:name=".MySMSReceiver">
       <intent-filter android:priority="999">
```

```
            <action
android:name="android.provider.Telephony.SMS_RECEIVED" />
          </intent-filter>
     </receiver>
```

Step II – Now, after gaining access to these messages a data type is required to store the accessed messages. String is considered as the most appropriate data type for temporary storing the messages so that they could be used further.

Step III – Once the message is retrieved in a string type then it becomes very easy to transport it anywhere. This message can be stored into a text file i.e. a file with *.txt extension and then that file can be transferred from one device to another. The file transfer may take place by establishing a socket connection over TCP or UDP.

3.2.3 Illustration of SMS theft attack

To demonstrate the theft of messages (SMSs) in an android device, an application has been coded which reads the SMSs received on it and stores it in a text file in the external memory directory. Now, when some information is placed in the form of a file in the external memory location then it can be easily forwarded to any other device through various means like TCP file transfer SMS forwarding etc. Here, a simple application has been coded in the Android API. Now, as it is known that Android is an open source operating system and all its APIs are nothing but Java APIs so this makes the task rather easy for anyone.[15]

Following are the steps in which the SMS theft attack has been performed.

Step I – First a text message is received on the android mobile device. Here for testing purpose a text message is sent and received on a same device as shown in Figure 8. First “Test message 1” is sent from the device at 2231 hrs and then received on the same device. Then “Test message 2” is sent from the device at the same time and again it is received on the same device at 2231 hrs. Similarly third test message i.e. “Test message 3” is sent from the device to the same device. Here in this screen shot it was not possible to show the same because of the limitation of the screen size of the device which has been used to do the same.

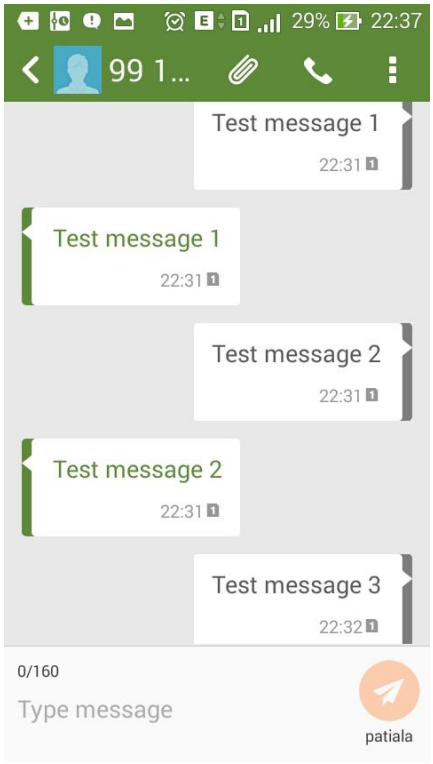
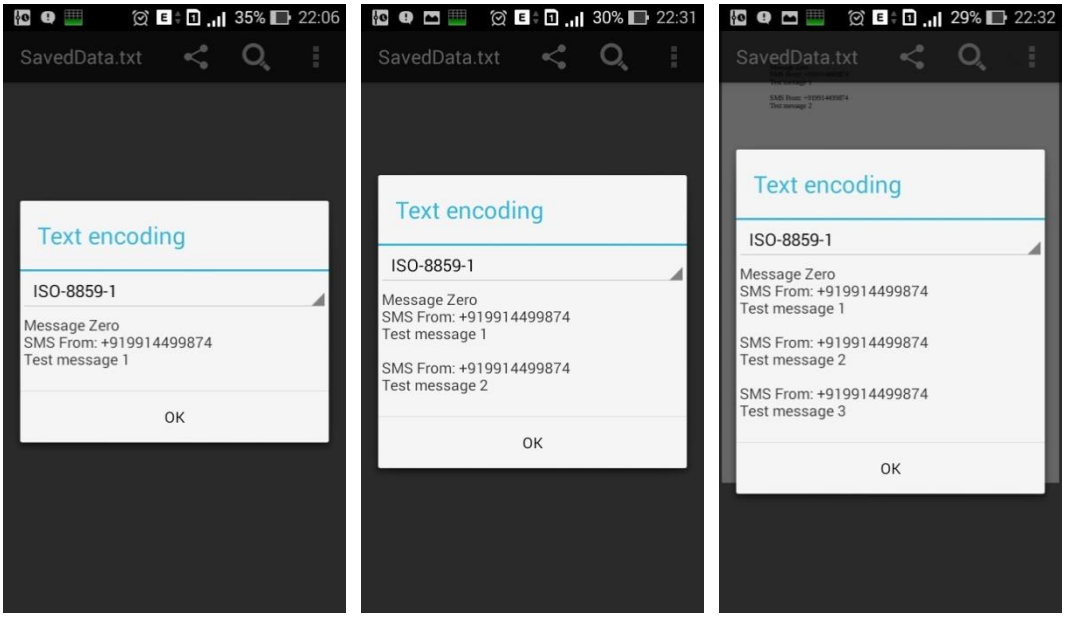


Figure 8 SMS received on an android mobile phone device

Step II – Now the application reads the messages and saves them to a text file. Figure 9 shows how messages are received and then dumped into a file. The text encoding mentioned here is ISO-8869-1 which is the traditional encoding technique for the SMSs. If the figure 9(1) is looked upon the message, ”Test message 1” is received by the device at 2206 hrs and it has been saved into the file “SavedData.txt” thereafter. The string “Message Zero” shows the starting point of the messages i.e. from there the leaked messages starts and then further they are appended. Now in figure 9(2) at 2231 hrs another message i.e. “Test message 2” is received and again read by the application and gets appended into the file again. The appending of messages is clearly shown here and the pop up is also showing the encoding type of the text which is being transferred into the file. The pop up appears as soon as the file is opened through the file manager and this file is non-editable as it is controlled by an application that is running consistently in the background itself. After that in figure 9(3) at 2232 hrs another message i.e. “Test message 3” is received by the android device and again it is appended into the text file as shown in the third section of the image. This pop up vanishes once we push the ok button on the screen and then the actual text file appears again in a non-editable format.



(1) (2) (3)

Figure 9 SMS being transferred to an external file

Step III – Now, when the text file is ready it looks like in figure 10 and thus can also be forwarded anywhere. This file is non-editable and only opens up in the view mode. This is because the file is opened under the application and the file write permissions are only restricted with one application at a time.

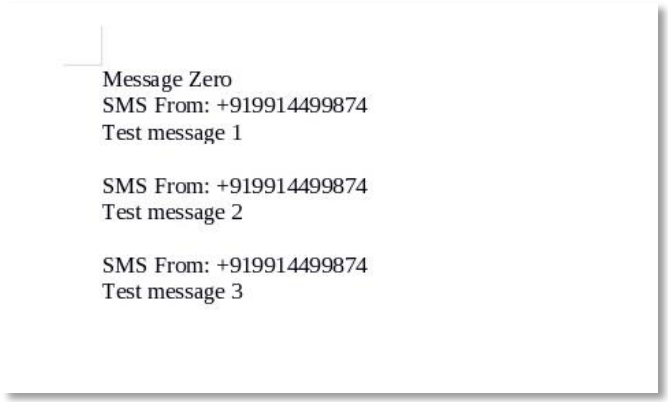


Figure 10 External file that stores the SMSs

As it has been observed in this section how simple it is to leak SMSs from an android mobile phone device, thus a solution is required to curb the vulnerabilities related to the SMS API in the android. One such solution will be discussed in the next chapter.

CHAPTER 4

PROBLEM SOLUTION

4.1 GOOGLE'S SECURITY ENHANCEMENTS IN ANDROID V4.4.

The problem of SMS theft attack is not new but not many real solutions have come up to revolutionize the SMS privacy. After the launch of Android KitKat version 4.4 there have been a major change though. On the previous versions of android i.e. till Jelly Beans the SMS receiver functionality was not given a native or default status i.e. any application could be set to receive and send the SMSs, but from version 4.4 onwards there has been an improvement that a default application has to be there for receiving and sending SMSs. There is a broadcast receiver class in the android API that is used to broadcast the messages of a particular intent to the apps which are intended to receive those. Now, when the messages are received by those apps they could abort the broadcast of the SMS to the native app and could act as SMS eaters. From version 4.4 onwards after the concept of the compulsory default app to receive and send SMSs, this exploit was handled by Google. [26]

Below is a code segment that shows how the SMSs were received and could get eaten away without even notifying the device user.

The messages will only be restricted to the particular application to which this code corresponds to but only for the versions 4.3 or below.

```
if (true) {
    abortBroadcast();
   }
```

There are many direct or indirect methods which are available through which we can detect or stop the leakage of messages (SMS) in android devices.

4.2 PROPOSED SOLUTION

If the SMSs received on a phone are stored into an open data base then anyone can access them, but if the SMSs are stored in the internal memory of the application i.e. the reserved memory, then they cannot be accessed by any other application until the architecture of the application is known.[36]

4.3 SECURED SMS RECEIVER

Secured SMS receiver is an application which access the SMS API in the existing android system but then saves the messages in its own internal memory which is further not accessible by anyone or any other application. This application does not acts as the default SMS application for sending and receiving messages but only a sample application for receiving the messages and saving them in its own application memory. By this application it has been illustrated that if the messages are stored in the internal directory then the message handling application can be much safer than the existing ones.

4.3.1 Flowchart for the algorithm

The flow of the process in the proposed algorithm i.e. the secured SMS receiver is shown in figure 11. Here the method is almost the same as that in the theft attack but with slight differences.

1. Here first an SMS is received on the android device.

2. Now, again the body of the SMS is copied into a string data type of the java native API. Now when the data is transferred into a string then it can be transferred to any file.

3. Thus the data is transferred into a text file which is in the internal memory of the application.

4. Now this data is saved in the system's memory allocated to the application.

This file cannot be accessed by any other application or by any file explorer. To read this file through an external source root permissions of the operating system needs to be accessed which is not possible in a non-rooted device. [49] Whenever a new android device is bought from the market place these root accesses are not provided to the customer by the handset manufacturers as they have their own proprietary version of OS installed on their devices. This is done to provide security to the end user as android is very vulnerable to attacks if rooted.

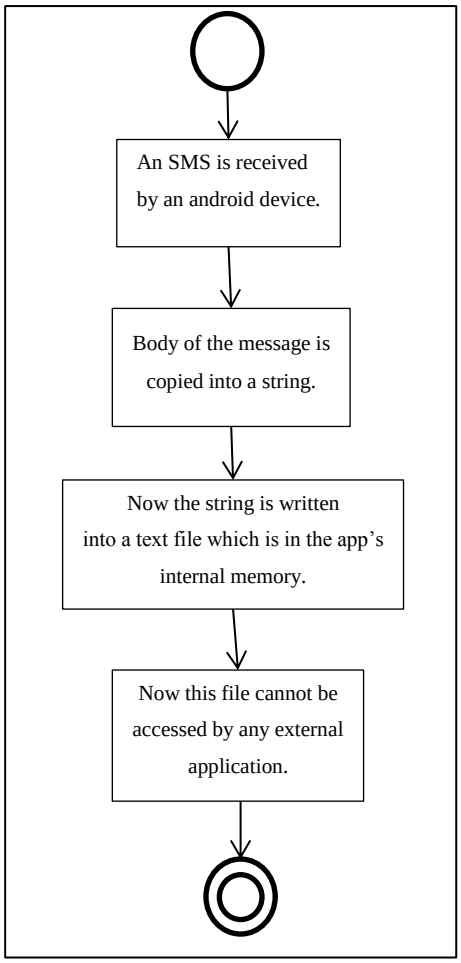


Figure 11 Flowchart for secured SMS server

4.3.2 Algorithm for secured SMS receiver

Step I – Set the priority on the intent to receive sms in manifest.xml file to 999 as:

```
<intent-filter android:priority="999">
```

Step II – When an SMS is received on the device then it is caught by the intent,

```
if(intent.getAction().equals(Telephony.Sms.Intents.SMS_RECEIVED_ACTION))
```

Step III – Now an Object array is created which could hold the sms data from the sms bundle.

```
Object[] sms = (Object[]) intentExtras.get(SMS_BUNDLE);
```

Step IV – When the SMS_BUNDLE is typecast as an Object instance then it is converted into a string.

Step V – When this string is available it is written into a file along with the sender's name into the internal memory of the device. This file is not reachable to any other application but to the application which is accessing it.

```
String path=getApplicationContext().getFilesDir().
getAbsolutePath();
```

CHAPTER 5

RESULTS AND COMPARISONS

5.1 INDIRECT METHODS TO DETECT SMS THEFT

These are some methods which are used to keep a check on, the applications in an android device, which requires permissions. Some of them also tell if the application permissions required falls in the grey area of their core functionality or if they are genuinely required by the app. [47]

For observing the memory requirements at runtime for various applications the "OS monitor application" has been used. This application is available for free on the google play store. There are many other things this application shows i.e. the start timing of the process, the name of the particular process in the application which is running, thread id of the thread which is being executed, process id which is used to identify the process with the scheduler and status of the process if it is in the sleep mode or active, user of the process, and the runtime memory which is required by the app. [14]

5.1.1 Checking for application permissions in system's app settings.

In every android device there is an app monitor in the system options where one can check all the applications along with their permissions as shown in figure12. By this way it can be made sure if any application using SMS API actually require it for its core functioning or not. If an application requires permission to access the SMS functionality and it may sound fishy as that functionality really doesn't really contribute to the core functioning, then that app could be uninstalled. This utility tells which application can access the SMS data, but again do not prevent any kind of leakage or theft of SMS.[6]

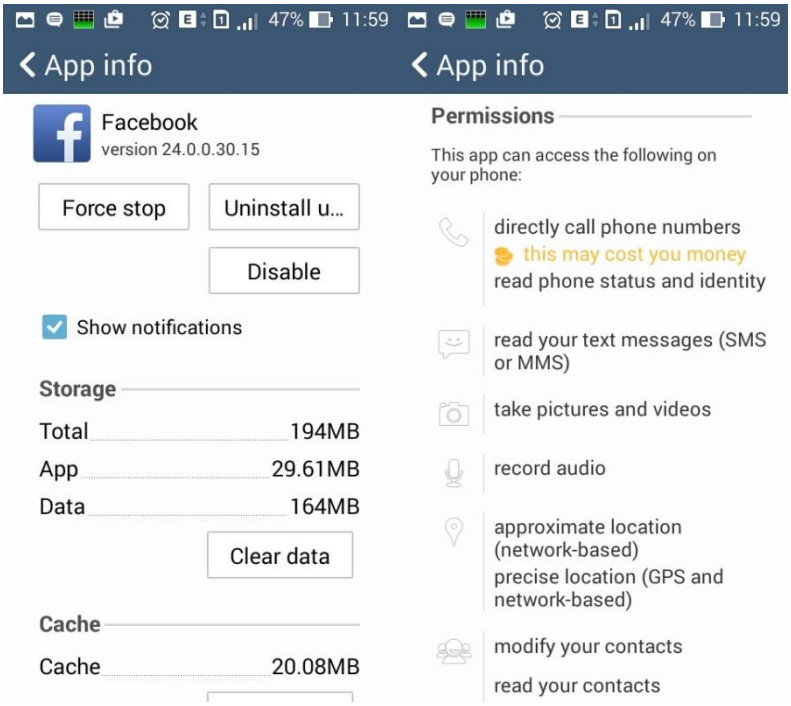


Figure 12 Android app info and permissions

Memory requirements:

The android system settings monitor the whole device regularly and thus have some memory requirements which have been captured using the OS Monitor software available on the app world. The android system server requires 100.7 MB of RAM space as shown in figure 13 under the Android System tag in the memory section.

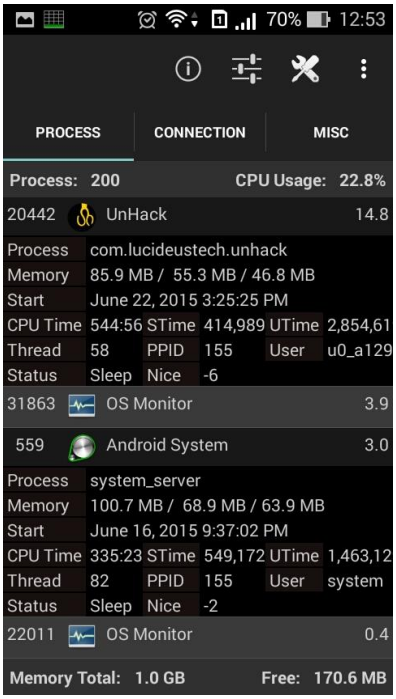


Figure 13 OS Monitor Snapshot showing requirements of system settings

5.1.2 Unhack for android devices.

Unhack is an application that was uploaded by Lucideus Corp. in the beginning of this year i.e. 26th February 2015. This app allows the device user to look upon all the application's permissions in a very organized manner. Here one can look all the apps installed on the device grouped into permissions that they have attained. Say the apps which require the access of SMS are grouped into a cluster then the apps which require permissions to access call logs are grouped into different cluster. Obviously the permissions overlap with each other in various devices and so do the groups. This application let the user identify if a particular app is fishy or safe or totally unsafe based on the comparison between the app's permissions and the permissions required for the core functionality of that particular app. This application tells which application can access the SMS data, but again do not prevent any kind of SMS theft. The snapshots of the same has been shown in the figure 14 below.[10]

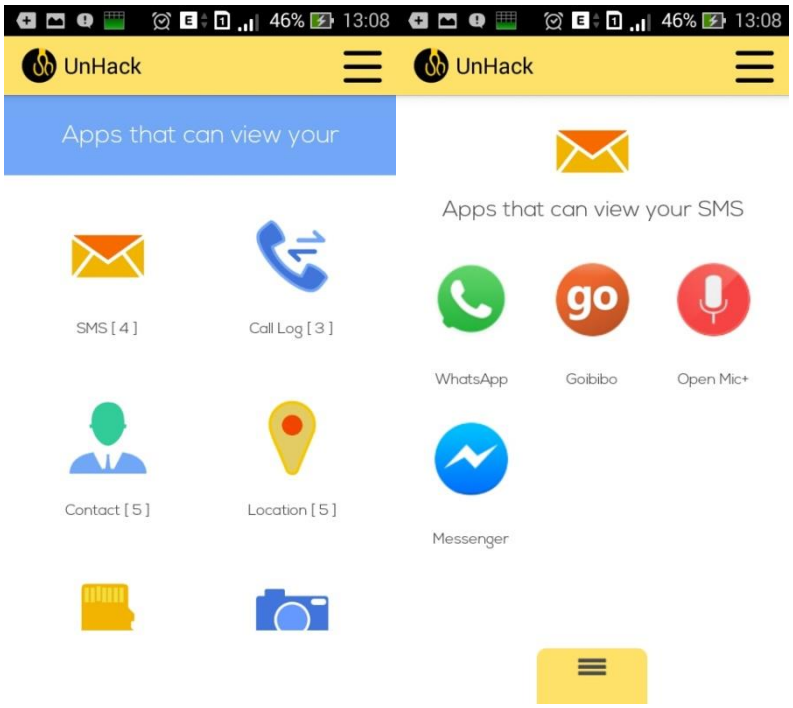
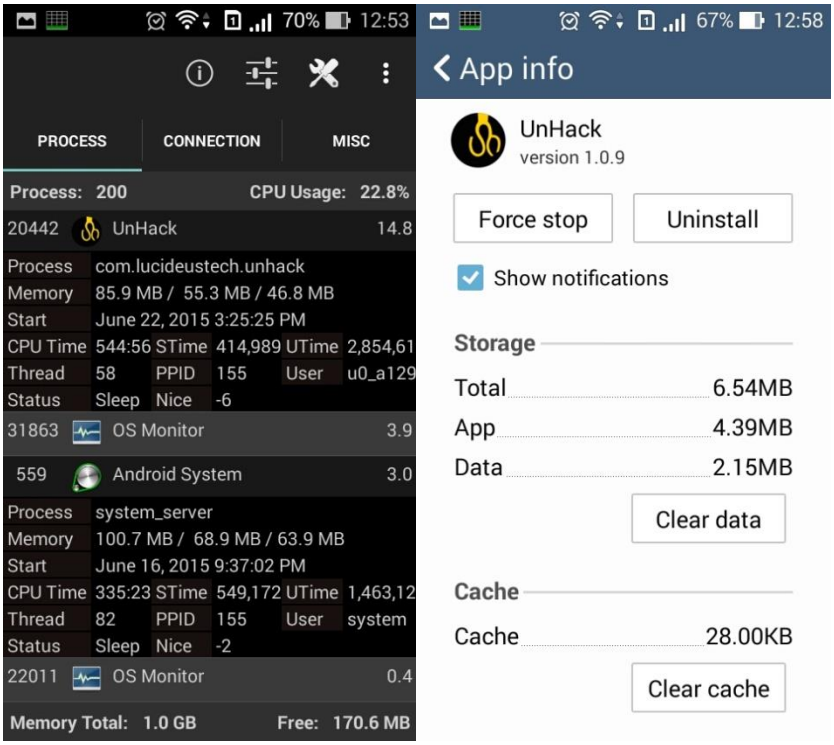


Figure 14 Unhack app snapshots

Memory requirements:

"Unhack" monitors the whole device regularly and thus, have some memory requirements, which have been captured using the OS Monitor software available on the app world. The Unhack app requires 85.9 MB of RAM space as shown in figure 15(1) under the Unhack tag in the memory section. The storage memory required by

the app is 6.54 MB comprising of 4.39 MB for the app and 2.15 MB for the data. The same has been shown in figure 15(2).



(1) (2)

Figure 15 Snapshots showing memory and RAM requirments of Unhack

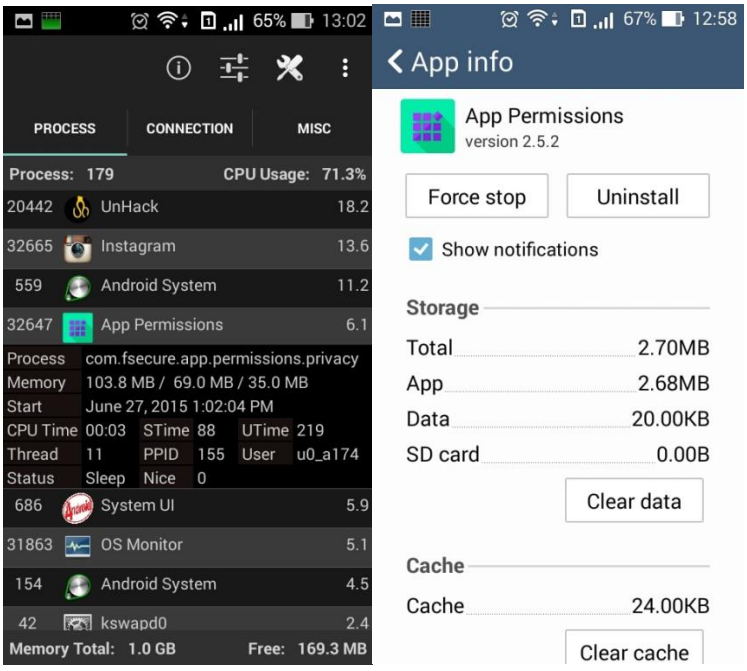
5.1.3 F-Secure app permissions.

Similar to UnHack there's another app i.e. F-Secure App permissions, which allows the device user to look upon all the application's permissions in a very organized manner. Here, one can look all the apps installed on the device grouped into permissions that they have attained. Say the apps which required to access the SMS are grouped into a cluster then the apps which require permissions to access call logs are grouped into different cluster. Obviously the permissions overlap with each other in various devices and so do the groups. This application let the user identify, if a particular app is fishy or safe or totally unsafe based on the comparison between the app's permissions and the permissions required for the core functionality of that particular app. This application tells which other application can access the SMS data, but again do not prevent any kind of SMS theft. Snapshots of this app have been shown in figure 16. It shows how applications have been classified neatly on the basis of system permissions required by various apps.[13]

Figure 16 F-Secure app snapshots

Memory requirements:

"F-Secure app permissions" monitors the whole device regularly and thus, have some memory requirements which have been captured using the OS Monitor software available on the app world. The "F-Secure App permissions" app requires 103.8 MB of RAM space as shown in figure 17(1) under the "App-Permissions" tag in the memory section. The storage memory required by the app is 2.70 MB comprising of 2.68 MB for the app and 20 KB for the data. As shown in the figure 17(2).



(1) (2)

Figure 17 Snapshots showing F-Secure's memory and RAM requirements

5.2 DIRECT METHODS TO PREVENT OR DETECT SMS THEFT

There are some methods or apps which are built to directly access the permissions and edit them in a rooted android device or help in the direct prevention of SMS theft attack if happening, by aborting the application. These methods are not very much successful yet, still they began to revolutionize the SMS security concern and may be able to accomplish it soon.

5.2.1 Permission manager for android devices

Permission manager is an application that takes the route access to an android device and then helps in editing the permissions required by various applications in the device. The only drawback with this is, that it requires the root access to the android device and do not works properly on the proprietary android OS. The snapshots of this app have been shown in figure 18 below. [12]

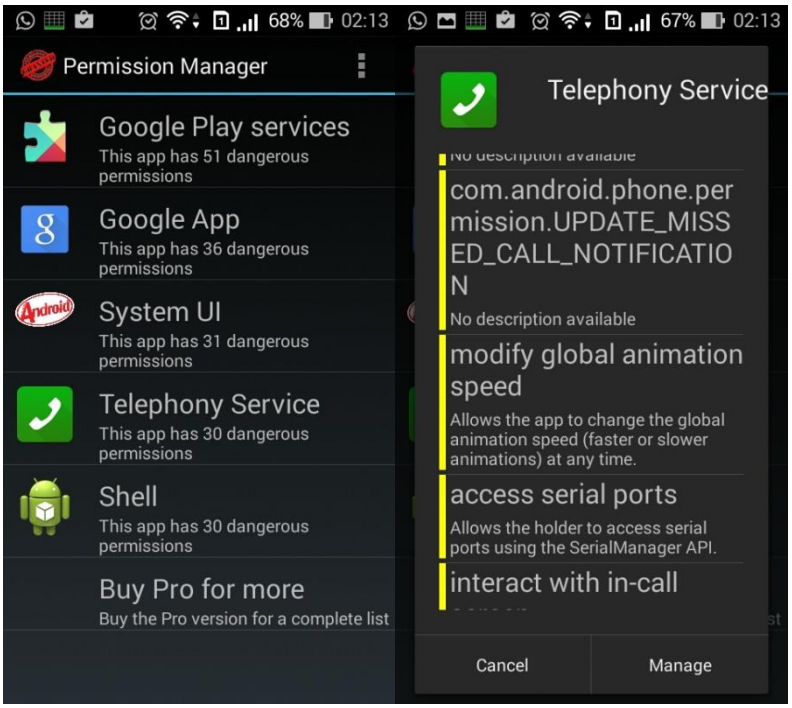


Figure 18 Permission manager app snapshots

5.2.2 Advanced native SMS app by Dongwoo Kim

Dongwoo Kim gave a solution that requires changes in the native SMS apps. It provides a search based classification of the text messages received on an android device based on the keywords present in the messages. There is a set of regular keywords which are considered to be residing in important or transactional messages.

Every message that is received on the device is first scanned word by word and then is decided whether to keep it in normal or secure message. This method works on the simple principal, "the SMS which is received on an android device is stored in a database which can be retrieved very easily". Hence, this application first classifies the messages on the basis of keywords and then keeps the important messages in a secure database format which is encrypted by a strong algorithm. The rest of the messages which are not as important as the transaction or verification messages can reside in the normal database.

The flaw with this app is that it can only run on a rooted android device as it requires changes in the android API at root level. Moreover, when every message is scanned when received it makes the application further more vulnerable if hacked and surely it is time consuming. Snapshot of the application with their classification (secure and normal) has been shown below (figure 19). [7]

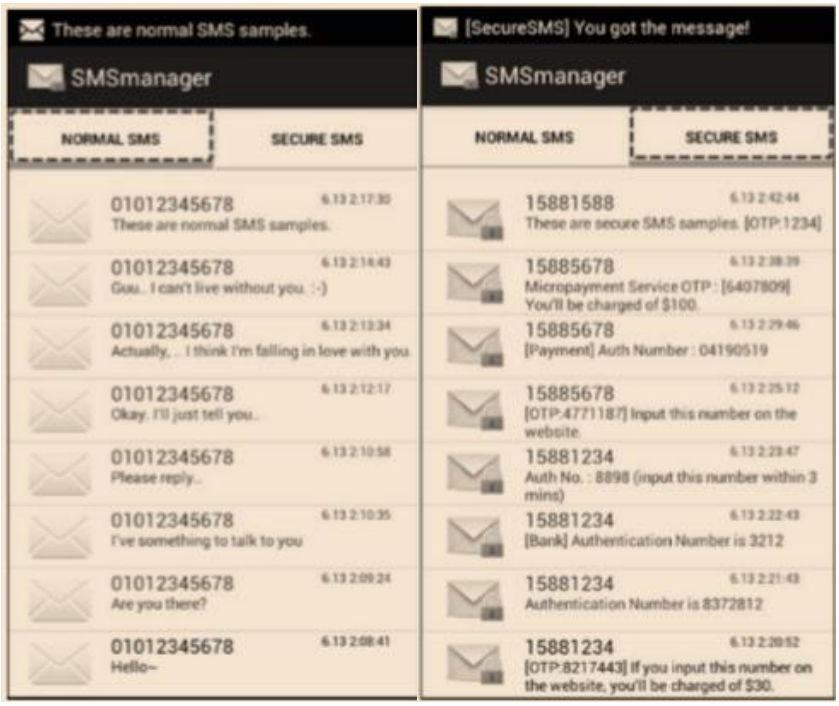


Figure 19 Advanced native SMS app snapshots. [7]

5.3 PROPOSED METHOD OF SECURING SMSs FROM THEFTS

Many techniques have been surveyed in this document, which are available and a self-made approach is also provided to curb the problem of SMS leak in Android devices. Comparisons of all those applications which are available under the indirect solutions

section as well as the direct solutions section have been made to the proposed solution i.e. a secured SMS receiver.

Figure 20 depicts a simple SMS inbox as the application has been made to depict a simple secured SMS receiver functionality. Though, many more things can be incorporated into it once it could be implemented as a native or default application to send and receive SMSs through an android device. The messages which are displayed on the screen are being extracted from an internal file which has already been fetched. Further, an attack failure has also been demonstrated i.e. if an external app tries to access the internal file of this application then it gets aborted and is stopped by the operating system itself.

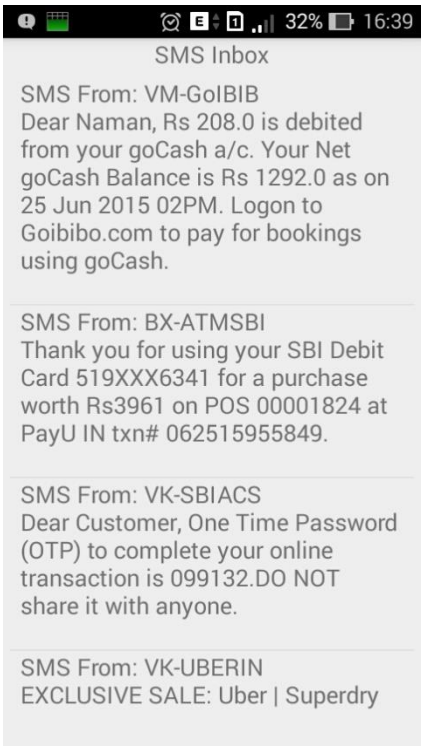


Figure 20 Secured SMS Receiver snapshot

Here all the messages are displayed in a list view in an android activity and all the data corresponding to these messages is rested in an internal file directory. These messages data cannot be read from any other applications. Now if any other application tries to access a message from this app it will not be able to retrieve the same. To illustrate this, an application has been developed namely “My First App” which tries to retrieve or leak SMSs from the internal memory of this application. Now, when a message is received, this application tries to retrieve the messages from

the internal data of the app which is not possible because of the android system properties. [48] So, when it tries to access the database anyway, a message pops up on the device and the “My First App” is aborted by the system itself.

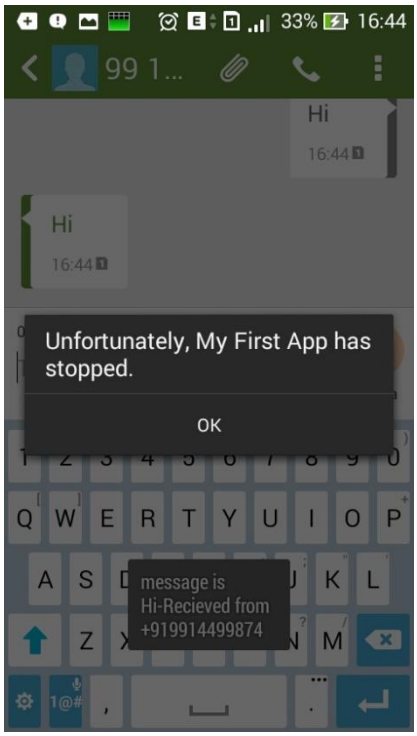


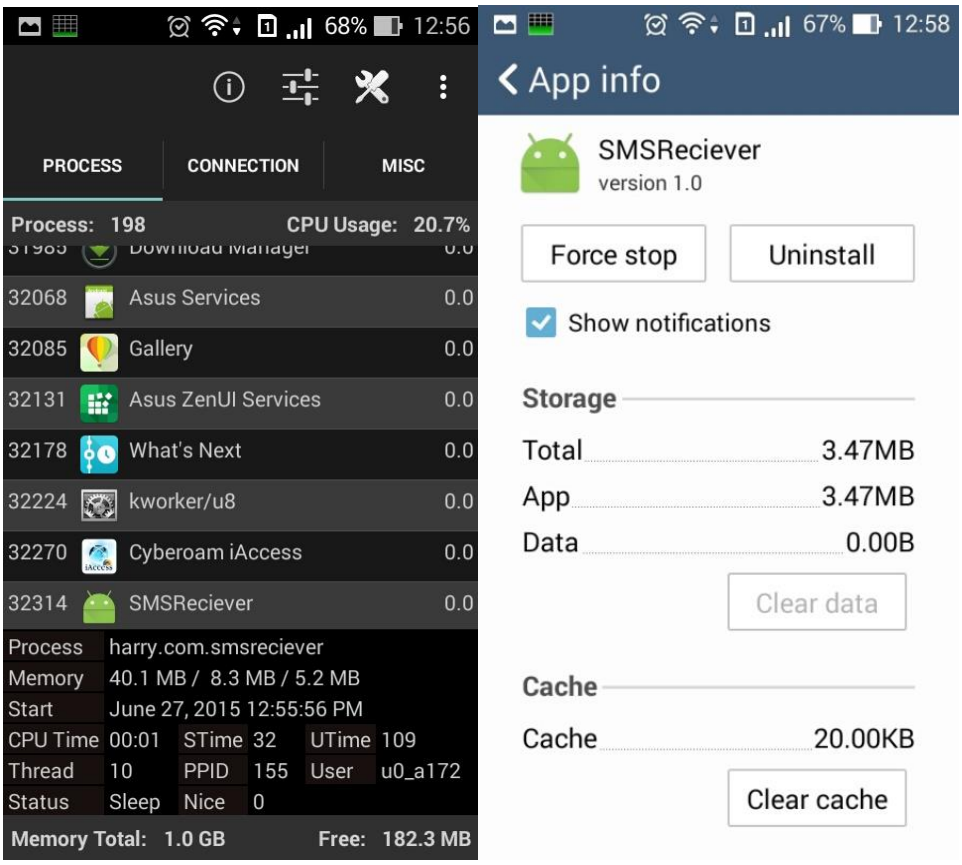
Figure 21 Application aborted while trying to access internal file

Same has been depicted in the figure 21 above. It happened because the app i.e. “My First App” was trying to access the root permissions which reside only with the system. If any application tries to access the root file system then it is forcefully closed i.e. aborted. Moreover the overall security of the android open source operating system is built only on one strong pillar i.e. stopping the end user to fully use the device’s or operating system full capabilities. This means that when an android phone is bought from the market place by an end user then its root access is not provided to the user and if any application tries to access the root which is by the way not authorized to then that particular application gets aborted.

Memory Requirements:

Secured SMS receiver runs in the background so that if an SMS is received it could retrieve it and save it in the internal database and thus have some memory requirements which have been captured using the OS Monitor software available on the app world. The secured SMS receiver app requires 40.1 MB of RAM space as shown in figure 22(1) under the “SMS receiver” tag in the memory section. The

storage memory required by the app is 3.47 MB. The same has been shown in the figure 22(2).



(1) (2)

Figure 22 Snapshots showing memory and RAM requirements of Secure SMS Receiver

5.4 COMPARISON OF SECURED SMS RECEIVER WITH THE INDIRECT METHODS

This data has been calculated from the actual usage of these applications and the memory and CPU usage statistics have been calculated by using an application called OS Monitor which is available over the Google Play Store. Moreover the features of some applications are also available in the application details available over the play store itself. The indirect services do not play a sufficient role though but still they could be of substantial use. The comparison of these solutions has been provided in a tabular format in the Table 2 below. It can be observed that secured SMS receiver is the only appropriate solution for the SMS theft problem in the device having the open source android operating system. Secured SMS receiver is the only application which prevents the leakage of SMSs to happen through the device whereas the other

applications are just there to observe the system permissions granted to the applications installed on the device.

Table 2 Comparison of Secure SMS Receiver with Indirect methods

Features / Solutions	Prevention	Targets SMS Only	Memory Req.	RAM Requirements
System Settings[32]	No[1]	No[1]	System dependent[1]	100.7 MB[1]
Unhack[10]	No[2]	No[2]	6.54 MB[2]	85.9 MB[2]
F-Secure[13]	No[3]	No[3]	2.70 MB[3]	103.8 MB[3]
Secure SMS Receiver	Yes[4]	Yes[4]	3.47 MB[4]	40.1 MB[4]

1. For the validation of above facts please refer to Section 5.1.1 of this chapter.
2. For the validation of above facts please refer to Section 5.1.2 of this chapter.
3. For the validation of above facts please refer to Section 5.1.3 of this chapter.
4. For the validation of above facts please refer to Section 5.3 of this chapter.

No application targets the SMS thefts, but the proposed system. Now, the memory requirement is one of the most important factors which one considers while installing an app on an android device as the device has a limited memory and resources at its disposal. If the above table is looked upon and a comparison between the memory requirements is made then clearly secured SMS receiver requires the least RAM of all of them. The RAM requirement of the secure SMS receiver is 40.1 MB as observed at a particular instance. This number is even less then the half of the RAM requirement of the other apps in the competition. Now if the storage is concerned the secured SMS receiver app is a little behind the F-Secure applications permissions app. But the major aspect of the application which targets the sensitive vulnerability like this should lie in the effectiveness and resource utilization at the same time. A little more storage memory i.e. a very feeble difference hardly matters. Storage is way cheaper than processing time as well as security. There are many other applications in the

market that helps in keeping a check on the application permissions in android devices but none of them helps in the prevention and that is the major drawbacks of these monitoring applications.

5.5 COMPARISON OF SECURED SMS RECEIVER WITH THE DIRECT METHODS

Table 3 shows the comparison between the direct solutions and the newly proposed one.

Table 3 Comparison of Secured SMS Receiver with direct methods

Features / Solutions	Prevention	Proprietary OS Support	Useful for non-rooted devices	Private secure database
Permission Manager[12]	Yes[1]	No[1]	No[1]	No[1]
Adv. Native App by Dongwoo Kim[7]	Yes[2]	No[2]	No[2]	Yes[2]
Secure SMS Receiver	Yes[3]	Yes[3]	Yes[3]	Yes[3]

1. For the validation of above facts please refer to Section 5.2.1 of this chapter.
2. For the validation of above facts please refer to Section 5.2.2 of this chapter.
3. For the validation of above facts please refer to Section 5.3 of this chapter.

Now here, the comparisons have been made among the applications which are built to fight against the SMS theft attacks. Advanced Native SMS app by Dongwoo Kim and the proposed secured SMS receiver are built to safeguard the SMSs which are received on an android mobile phone device by keeping them in a secret database. The app by Kim first classifies the messages on the basis of the keywords in them and then keeps them in two different sections i.e. normal and important messages, whereas the secured SMS receiver keeps the data in a secured database in the internal application memory which further cannot be accessed by any other application unless it has the root access to it. The disadvantage with the advanced native SMS app is that

it runs only on a rooted device for now, until some proprietary embeds it into its own android OS with custom access permissions. Now here, the secured SMS receiver has an edge over the advanced native SMS app. The secured SMS receiver runs perfectly fine over the proprietary as well as the rooted version of the android operating system. As the secured SMS receiver is in its initial stage some more improvements are required that may require some changes in the operating system at kernel level, but at this level it works perfectly fine. On the other hand if Permissions Manager app is concerned it helps in editing the application's permissions which the apps acquire from the operating system, i.e. if an application requires SMSs control then that control can be seized from that application. This application works fine but again it requires the access to the root as the control of the communication between the apps and the operating system has to be controlled which is only the task of the system administrator.

CHAPTER 6

CONCLUSION AND FUTURE WORK

CONCLUSION

SMS theft is a cyber-crime and can lead to serious consequences. Although, people now-a-days don't use SMS for general communication yet all the important transactional and verification messages are communicated through SMS only. Many people have worked to eradicate it, but the open source nature of android operating system has always have been a challenge. All the applications or solutions that could be devised are majorly working for rooted android devices. Rooting may help us in fighting the situation of information theft but causes a whole lot of problem i.e. it increases the vulnerabilities in an android device as if the root access is granted then anyone can secretly install an application on the android device which may or may not be visible as an icon and can take over the control on the android device remotely. This research has tried to safeguard the SMSs from being stolen by saving them in the internal database i.e. in the system's memory of the android OS.

FUTURE WORK

The futuristic motive of making this application is to make it a part of android's native API i.e. it should become the default application to receive and send SMSs. There is an abort function in the intent class of the android API. It restricts further receipt of message by any other application, when received by the default or native application. The abort function was revoked from the android API after the launch of v4.4 because of security reasons. After SMSs are received by the application proposed in this research if the further broadcast is aborted, then it may help in securing the messages from external applications which tries to leak them from the device.

References

[1] A. Khandelwal and A. K. Mohapatra, ‘An Insight into the Security Issues and Their Solutions for Android Phones’ in *2nd International Conference on Computing for Sustainable Global Development*, 2015, pp. 106-109.

[2] A. P. Felt et al., ‘Android Permissions Demystified’ in *CCS*, Chicago, Illinois, USA, 2011, pp. 627-637.

[3] A. Wahid et al.‘Anti-Theft Cloud Apps for Android Operating System', *Sixth International Conference on Computational Intelligence and Communication Networks*, pp. 765 - 769, 2014.

[4] Android Developers. [Online]. Available: http://developer.android.com/training/index.html [Accessed: May 5, 2015].

[5] Android Sending SMS. [Online]. Available: http://www.tutorialspoint.com//android/android_sending_sms.html [Accessed: April 12, 2015].

[6] B. Sanz, et al. 'On the Automatic Categorisation of Android Applications', *The 9th annual IEEE Consumer Communications and Networking Conferrence*, pp.149 – 153, 2012.

[7] C. Mulliner et al., ‘SMS-based One-Time Passwords: Attacks and Defense’ in *Technische Universitat,* Berlin, 2014.

[8] D. Bruschi et al., ‘S-ARP: a Secure Address Resolution Protocol’, in Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC), 2003.

[9] D. Kim and J. Ryou, ‘SecureSMS: Prevention of SMS Interception on Android Platform’ in *IMCOM (ICUIMC)*, Cambodia, 9-11 January, 2014.

[10] E. Erturk, ‘A Case Study in Open Source Software Security and Privacy: Android Adware’, in *World Congress on Internet Security*, 2012, pp. 189-191.

[11] E. Sugiono et al., ‘Android Security Assessment Based on Reported Vulnerability’ in

International Conference on Data and Software Engineering, 2014, pp. 1-6.

[12] Google Play Store. [Online]. Available: https://play.google.com/store/apps/details?id=com.lucideustech.unhack&hl=en. [Accessed: May 15, 2015].

[13] Google Play Store. [Online]. Available: https://play.google.com/store/apps/details?id=com.fsecure.app.permissions.privacy&hl=en. [Accessed: May 15, 2015].

[14] Google Play Store. [Online]. Available: https://play.google.com/store/apps/details?id=com.appaholics.applauncher&hl=en. [Accessed: May 15, 2015].

[15] J. Kulandai. [Online]. Available: http://javapapers.com/ [Accessed: May 5, 2015].

[16] K. Allix et al., 'A Forensic Analysis of Android Malware How is Malware Written and How it Could be Detected? ', in IEEE 38th Annual International Computers, Software and Applications Conference, 2014, pp. 384-393.

[17] K. Hamandi, et al. 'Android SMS Malware: Vulnerability and Mitigation', 27th Internationa Conference on Advanced Information Networking and Applications Workshops, pp. 1004 - 1009, 2013.

[18] L. Jeter and S. Mishra, 'Identifying and Quantifying the Android Device Users' Security Risk Exposure' in International Conference on Computing, Networking and Communications, 2013, pp. 11-17.

[19] L. Li et al., 'IccTA: Detecting Inter-Component Privacy Leaks in Android Apps' in IEEE 38th Annual International Computers, Software andApplications Conference, 2014, pp. 384-393

[20] M. Ahmad, et al. 'Comparison Between Android and iOS Operating System in terms of Security', 8th International Conference on Information Technology in Asia (CITA), pp. 196 – 208. 2013.

[21] M. Alam, et al. 'Context-aware multi-agent based framework for securing Android',

Multimedia Computing and Systems (ICMCS), 2014 International Conference, pp. 961 – 966, 2014.

[22] M. Ataullah and N. Chauhan, ‘ES-ARP: an Efficient and Secure Address Resolution Protocol’, in IEEE Students Conference on Electrical, Electronics and Computer Science, 2012 IEEE.

[23] M. G. Gouda and C. Huang, ‘A secure address resolution protocol’ in International Journal of Computer and Telecommunications Networking, Computer Networks, Elsevier, 2003, pp. 57-71.

[24] M. S. Ahmad et al., ‘Comparison Between Android and iOS Operating System in terms of Security’ in 8th International Conference on Information Technology in Asia, 2013.

[25] M. Tripunitara and P. Dutta, ‘A middleware approach to asynchronous and backward compatible detection and prevention of ARP cache poisoning’, in Proceedings of the 15th Annual Computer Security Applications Conference, December 1999.

[26] N. Peiravian, et al. ‘Machine Learning for Android Malware Detection Using Permission and API Calls’, IEEE 25th International Conference on Tools with Artificial Intelligence, pp. 300 – 305, 2013.

[27] P. Jia et al., ‘A Framework for Privacy Information Protection on Android’, in International Workshop on Sensor, Peer-to-peer and Social Networks, ICNC Workshop, 2015, pp. 1127-1131.

[28] P. Chan, 'Static Detection of Android Malware by using Permissions and API calls', Proceedings of the 2014 International Conference on Machine Learning and Cybernetics, Lanzhou, pp. 82 - 87, 2014.

[29] P. D. Meshram and R.C. Thool, ‘A Survey Paper on Vulnerabilities in Android OS and Security of Android Devices’, in IEEE Global Conference on Wireless Computing and Networking, 2014, pp. 174-178.

[30] P. Faruki et al., ‘Android Security: A Survey of Issues, Malware Penetration, and Defenses’ in IEEE Communication Surveys & Tutorials, 2015, pp. 998 -1022.

[31] P. Limmaneewichid and W. Lilakiatsakun, ‘P-ARP: A novel enhanced authentication scheme for securing ARP’, in International Conference on Telecommunication Technology and Applications, Proceedings of CSIT, 2011.

[32] P. Teufl, et al. 'Android – On-Device Detection of SMS Catchers and Sniffers',Privacy and Security in Mobile Systems (PRISMS), 2014 International Conference, pp. 1 – 8, 2014.

[33] R. J. Vargas et al., ‘Security Controls for Android’ in Fourth International Conference on Computational Aspects of Social Networks, 2012, pp. 212-216.

[34] R. M. Savola and M. Kylanpaa, ‘Security Objectives, Controls and Metrics Development for an Android Smartphone Application’ in Information Security for South Africa, 2014, pp. 1-8.

[35] R. M. Savola et al., ‘Toward Risk-driven Security Measurement for Android Smartphone Platforms’ in Information Security for South Africa, 2013, pp. 1-8.

[36] S. Cho, et al. 'Benchmarking Java Application Using JNI and Native C Application on Android',12th International Conference on Control, Automation and Systems, pp. 285 – 288, 2012.

[37] S. G. Haider et al., ‘Preventing ARP Spoofing Attacks through Gratuitous Decision Packet’ in 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science, 2012.

[38] S. G. Hong, ‘Application Development Research Based on Android Platform’ in 7th International Conference on Intelligent Computation Technology and Automation, 2014, pp. 579-582.

[39] S. Yubo et al., ‘Using Short Message Service (SMS) to Deploy Android Exploits’ in International Conference on Cyberspace Technology, 2014, pp. 1-5.

[40] S.G. Bhirud, ‘Light weight approach for IP-ARP spoofing detection and prevention’, in Second Asian Himalayas International Conference, 2011.

[41] T. E. Wei et al., ‘Android Privacy’ in Proceedings of the 2012 International Conference on

Machine Learning and Cybernetics, Xian, 15-17 July, 2012, pp. 1830-1837.

[42] The New Boston. [Online]. Available: https://www.thenewboston.com/videos.php?cat=278. [Accessed: May 7, 2015].

[43] W. Enck et al., 'A Study of Android Application Security' in IEEE Communication Surveys & Tutorials, 2015, Vol. 17, No. 2 pp. 998-1022.

[44] W. Hu, et al. 'MIGDroid: Detecting APP-Repackaging Android Malware via Method Invocation Graph', Computer Communication and Networks (ICCCN), 2014 23rd International Conference, pp. 1 - 7, 2014.

[45] W. Lootah et al., 'TARP: Ticket-based address resolution protocol', in Proceedings of the 21st Annual Computer Security Applications Conference, 2007, pp. 4322–4337.

[46] W. Pak et al., 'Detecting and tracing leaked private phone number data in Android smartphones' in International Conference on Information Networking, Cambodia, 2015, pp. 503-508.

[47] X. Li et.al. 'A Light-weight Software Environment for Confining Android Malware', Eighth International Conference on Software Security and Reliability - Companion, pp. 158-167, 2014.

[48] X. Wu, et al. 'Hack Android Application and Defence', 3rd International Conference on Computer Science and Network Technology, pp. 676 – 680, 2013.

[49] Y. Zhang, et al. 'Permission Use Analysis for Vetting Undesirable Behaviors in Android Apps', IEEE Transactions on Information Forensics and Security, pp. 1828 – 1842, 2014.

Publications

N. Sharma and G.S. Chhabra ‘Centralized ARP Server to Detect ARP/MITM Attacks’, *International Arab Journal for Information Technology*.

Status :- Under review.

G.S. Chhabra and N. Sharma ‘Secured SMS Receiver System for Android Devices’, *International Journal of Network Science.*

Status :- Under review.

YouTube Video Link

https://youtu.be/1NHJFe1QlNk