ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 11, November 2017

Method & Implementation of Android Application Base STK Attacks

[1] Sushma Singh, [2] Ranjit Singh Chauhan
[1] Research Scholar Dept of ECE, Seth Jai ParkashMukund Lal Institute of Engineering &Technology, Yamuna Nagar.
[2] Asst. Professor Dept of ECE, Seth Jai ParkashMukund Lal Institute of Engineering &Technology, Yamuna Nagar.

***Abstract:-* Smart-phones are becoming so popular these days because of their extra features and their extraordinary capabilities compared to regular phones. They are very similar to PCs and laptops; therefore they are vulnerable to security problems and even worse with respect to memory and space limitations. Since smart phones are connected to internet so chances of security attacks are more to them along with GSM network Attacks. Lots of attractive games and applications are developed which intends to install suspicious software which can control our mobile phones by STK attacks. These applications can stole the secure information from our phone by STK attacks - send automatic SMS or make automatic calls .Smart-phones Vs simple mobile phones uses different technologies, so they are more exposed to different attacks. In addition, they are interoperating devices which work between the Internet and telecom networks, so they can bring Internet security threats to the telecom networks and cause serious damages. In this thesis STK-SMS attacks against Smartphone's are presented and some countermeasures against the attack are introduced. In addition some solutions internet-side defence and telecom-side defence and co-ordination mechanism in order to make these threats ineffective are discussed.**

***Keywords*: STK, SIM, OTA, ANDROID, JAVA, SMS**

I. INTRODUCTION

Google's Android mobile phone software platform may be the next big opportunity for application software developers. Google announced the Open Handset Alliance and the Android platform in November of 2007, releasing the first beta version of the Android Software Development Kit (SDK) at the same time. Within a matter of a few months, over 1 million people had downloaded versions of the SDK from Google's website. In the United States, T-Mobile announced the G1 Android mobile phone in October of 2008, and estimates are that several hundred thousand G1s were sold before the end of that year. There are already several competing mobile phone software stacks in the market, so why is there such interest in Android?

Android has the potential for removing the barriers to success in the development and sale of a new generation of mobile phone application software. Just as the the standardized PC and Macintosh platforms created markets for desktop and server software, Android, by providing a standard mobile phone application environment, will create a market for mobile applications—and the opportunity for applications developers to profit from those applications.

SIM Application Toolkit (commonly referred to as STK) is a standard of the GSM system which enables the SIM to initiate actions which can be used for various value-added services. The SIM Application Toolkit consists of a set of commands programmed into the SIM card which define how the SIM should interact directly with the outside world and initiates commands independently of the handset and the network. This enables the SIM to build up an interactive exchange between a network application and the end user and access or control access to the network. The SIM also gives commands to the handset, such as display menu and ask for user input.

Android - It is no longer just a mobile phone. Nowadays Android applications are running anywhere and everywhere. Home Appliances, watches, TVs, car applications and with the Internet of Things kicking in quickly, Android applications will probably become even more prevalent in our lives.

Smart-phone is the trend of unified communications which integrate telecom and Internet services onto a single device because it has combined the portability of cell-phones with the computing and networking power of PCs. Today's smart phone operating systems allow users to install third-party

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 11, November 2017

applications directly from on-line application markets. In order to perform their functions, applications typically need specific permissions such as network access or access to user's personal data. How-ever, given the large number of independent developers, every application cannot be trusted to behave according to their declared purpose Certain types of malicious behaviours can be detected by inspection and testing while others cannot, malicious applications therefore find their way into the application markets to limit the potential impact of malicious applications, mobile phone operating systems (e.g., Android OS, Symbian OS, MeeGo OS, and Windows)

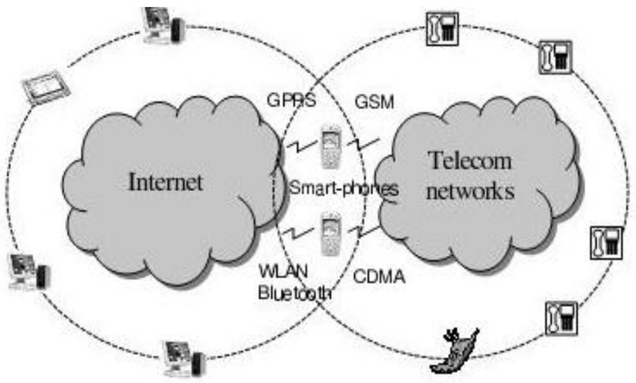


Figure 1: Architecture of Network

As illustrated in Figure, smart-phones, as endpoints of both networks, have connected the Internet and telecom networks together. Although the detailed design and functionality vary among these OS vendors, all share the following features:

- Access to cellular network with various cellular standards such as GSM /CDMA and UMTS.
- Access to the Internet with various network interfaces such as infrared, Bluetooth, GPRS/CDMA1X, and 802.11; and use standard TCP/IP protocol stack to connect to the Internet.
- Multi-tasking for running multiple applications simultaneously.
- Data synchronization with desktop PCs.
- Open APIs for application development.

II. ANDROID EXECUTION ENVIRONMENT

Applications in Android are a bit different from what you may be used to in the desktop and server environments. The differences are driven by a few key concepts unique to the mobile phone environment and unique to Google's intentions for Android.

Limited Resources

Mobile phones today are very powerful handheld computers, but they are still limited. The fundamental limitation of a mobile device is battery capacity. Every clock tick of the processor, every refresh of memory, every backlit pixel on the user's screen takes energy from the battery. Battery size is limited, and users don't like frequent battery charging. As a result, the computing resources are limited-clock rates are in the hundreds of MHz, memory is at best a few gigabytes, data storage is at best a few tens of gigabytes.

Mobile Mashups

In the desktop Internet world, mashups make it very easy to create new applications by reusing the data and user interface elements provided by existing applications. Google Maps is a great example: you can easily create a web-based application that incorporates maps, satellite imagery, and traffic updates using just a few lines of JavaScript on your own web page. Android extends that concept to the mobile phone. In other mobile environments, applications are separate, and with the exception of browser-based applications, you are expected to code your applications separately from the other applications that are running on the handset.

Interchangeable Applications

In other mobile software environments, applications are coded to access data from specific data providers. If you need to send an email from a Windows Mobile application, for example, you code explicit references to Pocket Outlook's email interface, and send the email that way.

The Android Software Development Kit supports Windows (XP and Vista), Linux, and Mac OS X (10.4.8 or later, Intel platform only) as host development environments. Installation of the SDK is substantially the same for any of the operating systems, and most of this description applies equally to all of them.

1. Install JDK

The Android SDK requires JDK version 5 or version 6. If you already have one of those installed, skip to the next step. In particular, Mac OS X comes with the JDK version 5 already installed, and many Linux distributions include a JDK.

2. Install Eclipse

The Android SDK requires Eclipse version 3.3 or later. If you do not have that version of Eclipse installed yet, you will need to go to http://www .eclipse.org/downloadsto get it, and you might as well get version 3.4 (also known as Ganymede), since that package includes the required plug-ins mentioned in the next step. You want the version of the Eclipse IDE labeled "Eclipse IDE for Java Developers," and obviously you want the version for your operating system.

3. Check for required plug-ins
You can skip this step if you just downloaded a current version of Eclipse as we recommended. If you are using a preinstalled version of Eclipse, you need to make sure you have the Java Development Tool (JDT) and Web Standard Tools (WST) plug-ins. You can easily check to see whether they are installed by starting Eclipse and selecting menu options "Windows →Preferences...". The list of preferences should include one for "Java" and one for either "XML" or "Web and XML."

3. Install Android SDK
This is where you should start if you already have the right versions of Eclipse and the JDK loaded. The Android SDK is distributed through Google's Android site, http://developer.android.com/sdk/1.1_r1/index.html. You will need to read, review, and accept the terms of the license to proceed. When you get to the list of downloads, you will see a table of distributions. Select the one for your operating system (XP and Vista use the same distribution).

III. IMPACT OF COMPROMISE

An attacker who has fully compromised a device which remains in use (whether a smart-phone or a PC) can effectively impersonate the user of that device.
This includes access to all data and network resources available to the user. This is because a sophisticated attacker can elevate privileges to that of the device's operating system, and carry out any activity from the device that the user would (and without the user knowing).
This includes making use of any credentials stored directly on the device, or those which are accessible from it. Storing credentials on hardware tokens provides a mitigation, as the attacker is then required to connect to the compromised device in order to make use of these credentials. This requires an attacker to expend more effort and engage in more -visible network activities. Any credentials stored directly on the device's main storage, however, can be collected by an attacker during the initial compromise and then used to impersonate the user and access resources from another location at the attacker's leisure.
As malicious email or web pages can be used by an adversary to make a successful initial intrusion into either a smart-phone or desktop, little stands i n the way of an attacker making further use of such techniques to compromise other systems (and gather privileged credentials) once inside an enclave. This can be enabled by using contacts listed in the address book of the user's device. For outdated smart-phones which are most vulnerable to this kind of attack, it is notable that applying the limited configuration guidance available for browsers, email clients, or PDF readers is a very weak mitigation when compared to updating to newer software. Although modern smart-phones are more resistant to fully remote compromise when compared to outdated android systems, their array of hardware features provides an attacker with much greater capabilities for information gathering and remote communications. This includes a microphone for listening to conversations, GPS for location tracking, cameras for visual surveillance, and cellular or WiFi radio for non- enterprise controlled or monitored network communications.

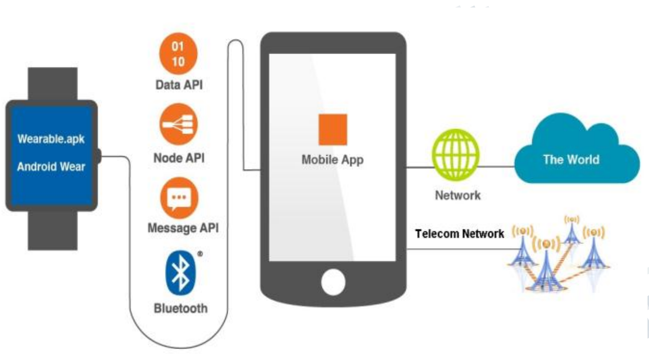


Figure 2: Communication with Mobile App

IV. DEFENSE

Proper security mechanism can be used while development of android systems and mobile applications so that security can't be breached and smart-phones can't be compromised.
This can be done in below ways
A) Android Platform Security Essentials
Android…. It is no longer just a mobile phone.
Nowadays Android applications are running anywhere and everywhere. Home Appliances, watches, TVs, car applications and with the Internet of Things kicking in quickly, Android applications will probably become even more prevalent in our lives.

Figure 3: Security Essentials

Android is based on a customized Linux OS version. The main differentiation from the classic PC Linux is that the

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 11, November 2017

Android OS was adapted to define every Application on the device as a separate User or entity.
Each Application runs on its own Virtual environment within the OS called a "Dalvik Machine (DVM)"*. Application code written in Java is modified to Java Byte Code and then converted to DEX (Dalvik byte code). The DVM will generate, on the fly, machine specific instructions to the ARM CPU (or other CPU in use). All Android applications are packaged as an APK (Android Application Package). The APK is a type of archived file which contains everything the android device needs in order to execute the application downloaded via the Google Play store or an alternate source.
*Dalvik is being shifted aside (Android L). Newer Android OS versions are using ART (Android Runtime) however the general idea stays the same.
As Android runs on Linux it is just natural that it will be using the Linux security mechanisms which have earned good reputation over the years.
Security wise, logically and physically separating each App allows processes and files to be protected from tampering and modification. Each App runs in its own Virtual Environment and does not (or rather should not) look into other app data or processes unless specifically allowed. This method of separation provides good basic protection even when developers are careless.
And additional security measure taken by Android is to ensure that APKs are signed. This will not allow the simple user to identify malicious behaviour however will allow App integrity validation. Most applications are installed from the formal Google Play Application store. Google do take a few basic measures to validate app integrity and security (See: Bouncer service) however there have been and will be some malicious developers who upload a malicious app to the store. These are usually detected by Google after a few or more than a few infections have been reported. At that point Google will remove the App from the Store however by that time the App has found its way onto devices and other alternative stores. Note: Android phones do have a setting that disables downloads from sources other than google play however this can easily be disabled.
So it seems that Google have taken significant steps to ensure the User's security.
Where is the problem? What are we worried about?

B) Mobile Application Threats Compared to Web Application Threats

Mobile Apps threat model is different and cannot be compared to other Apps such as web apps. Although the server functionality might be very similar between Web and Android Apps, when analyzing the threat model for Android Applications there is an extra major piece to be considered. Client side security concerns. The attack surface on Android Mobile devices or any mobile device includes a combination of server and client side security concerns. Therefore there are more problems to worry about compared to an equivalent web app. When performing Penetration tests on regular applications (Non Mobile), testers will issue request to the server side in an attempt to manipulate requests. This does not suffice for Mobile applications. There are additional considerations to keep in mind on the client side such as reverse engineering which is a key requirement when testing Mobile Apps. The Pen tester's skill set requirements are different. Tools to analyze Android application security are becoming more available however they are not yet as mature as the ones available for web apps.

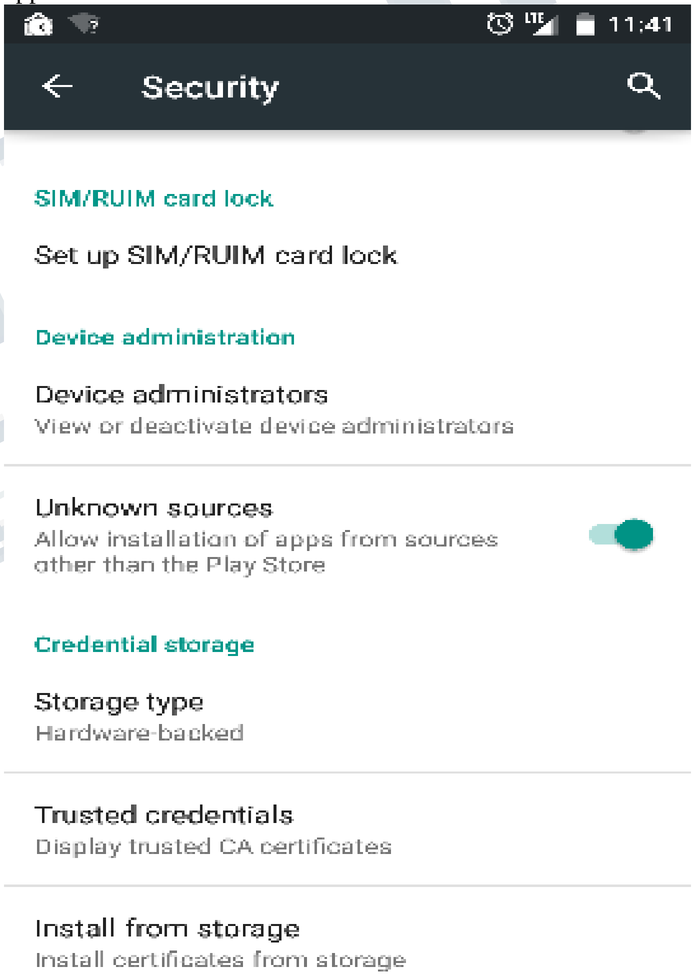

11:41
Security
SIM/RUIM card lock
Set up SIM/RUIM card lock
Device administration
Device administrators
View or deactivate device administrators
Unknown sources
Allow installation of apps from sources other than the Play Store
Credential storage
Storage type
Hardware-backed
Trusted credentials
Display trusted CA certificates
Install from storage
Install certificates from storage

Figure 4: Security in Mobile App

C) Android Mobile Application common vulnerabilities to look out for

This section discusses the most common vulnerabilities found on Android Mobile Applications however there are many more to be covered when developing or analyzing an application.

1. Bad file permission

Developers can mistakenly poke holes in the OS built in security known as the sandbox. Such holes might allow file unwanted access.

File creation – Make sure that you don't allow other apps to read your files by ensuring the permissions are correctly set

File Storage – If you are planning on using the device's SD card to store data you have to make sure the data is properly encrypted. The OS sandbox does not cover the SD card and therefore all applications can access the data it stores.

2. Sensitive client data storage (PII) – Addresses, emails, CC numbers and other IDs are used frequently for registration, authentication and for other legitimate purposes. Mobile Devices tend to get lost, forgotten or even stolen. PII data always has to be protected to avoid exposing personal data which can lead to privacy invasion or even identity theft. Do not include PII clear data within Log files, XML files, App manifest, SQL DB or even cloud.

3. Usage of bad cryptography – Developers understand that data needs to be protected however not always is the encryption methodology used strong enough. Don't create your own cryptography. Create separate keys per users, store keys securely and use the most up to date encryption techniques. Bad encryption is like having no encryption at all and provides a false sense of security!

4. Bad transport level security – TLS and SSL usage is a must to secure all communication channels. SSL provides encryption to the transport layer to prevent communication sniffing and when using SSL we are authenticating the server side with the certificate to ensure we are talking with the correct server and not a fake/malicious one (Malware Command and Control servers). Developers sometimes disable certificate validation code by mistake. This means that server authentication is not present which will allow MITM (Man in the middle) attacks. MITM attacks perform interception of data and can then decrypt and re-encrypt the data allowing the criminals to read your most secret or sensitive communication. A good approach to prevent that would be SSL Pinning. This means that app will only accept a single certificate CA which is pre-defined.

Weak server side controls –Regular web server attacks such as SQL injection, CSRF etc. are no different when it comes to mobile applications. Validate you are not exposing any vulnerabilities when building your code.

Authentication – Many mobile applications use hardware device identifiers to authenticate users. These can be IMEI, MAC addresses or similar. While these identifiers are not modifiable on the hardware layer, they can quite easily be controlled or tampered with on the software layer. Use strong authentication techniques when required. Some examples are 2-Factor Authentication and out of band authentication.

Insecure IPC (Inter Process Communication) – Android Mobile communication is possible by using internal communication mechanism and sending intents to each other. There are two types of intents. Explicit and implicit intents. Explicit intents are usually more secure as they specifically chose which application can trigger data collection a service on a different App. Implicit intents allow any App to trigger data collection services on the attacked Application. Using implicit intents is risky. Intents can be sent by an invalid caller i.e. a malicious app, which can be abused to cause your application to perform sensitive actions without your control, delegate permissions which it is not permitted to use such as collecting GPS coordinates, or trigger a DoS (Denial of Service) attack by calling exposed components which cause CPU intensive operations. These are just some basic examples however the range of possible abuse for implicit intents is much wider.

Client side injection – Malicious Apps can inject queries or code to other applications on the device and modify values of the attacked App.

Hard Coded secrets – Storing user IDs and sensitive information hardcoded

Secure Log exposure – Application store information about the app in Logcat or general files stored on the device or server side. This files can contains session IDs and other sensitive information. The Logs and apps should be protected. Do not store sensitive information on your logs.

V. DISCUSSIONS

There are some apps that never want to talk to the internet, but the app requests that permission (e.g. games that have some leader board or other social garbage we could care less about). There is no reason for us to take on the security risk of that permission if we don't want that feature. How about letting users deny some permissions, but still install the app. This is a very welcome move. I would add that I'd like to see one addition to installing Android software that could further prevent people from adding apps that could harm them or their phone.

1) When an app is installed, the permissions are checked as usual.

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 11, November 2017

2) Permissions that might cost the user money or want wider access than normal, are flagged in red.

3) The user cannot click the "OK" button until a certain amount of time has passed, say 10 seconds.

These three steps would work to prevent users to blinding clicking "OK" without first reading the permissions. Since they are unable to install the app until the counter has finished, it is increasingly likely they will read what is displayed on the screen and act accordingly.
Adding confirmation before any activity tends to send data outside from smart phones.
An example with snapshot is a s below:-
It is an Android application, once is installed in phone it replaces the default Message handling application, then it will run in background continuously and tracking every message which are coming from network. For every SMS it will ask a confirmation POPUP either you want to send message or cancel it. It will act as a barrier for SMS which were automatically send by mobile. It will also help to stop sending message if by mistaken SMS button is pressed by us. Overall it is good software to control our SMS permissions. Installation of this software will be helpful to stop STK events to be fired without prior to your knowledge (Only feasible for android & Smart Phones). Warning Screen before installing some applications tending to Smart phone compromise.

Applications

It's a very small software but very helpful to control sms.No messages can be Sent without our confirmation. We can also make same soft ware's for exchanging media, Bluetooth sharing,nfc, calls, mms etc so that all these can also be controlled.

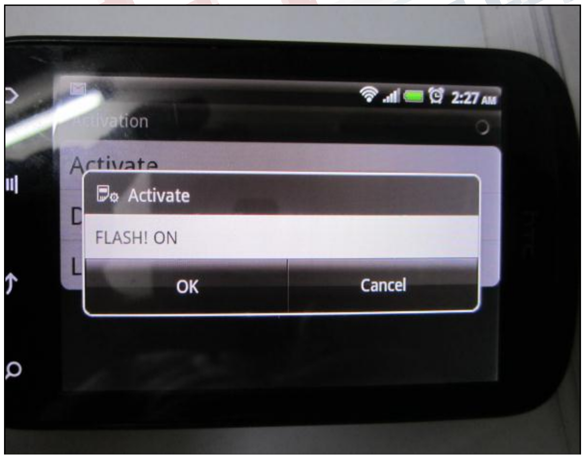


Figure 5: Confirmation Action For SMS

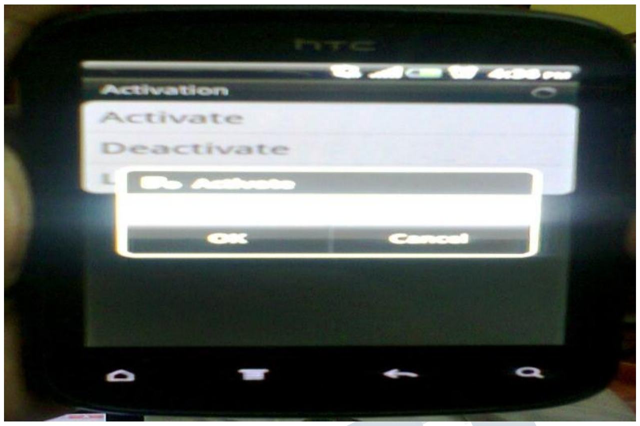
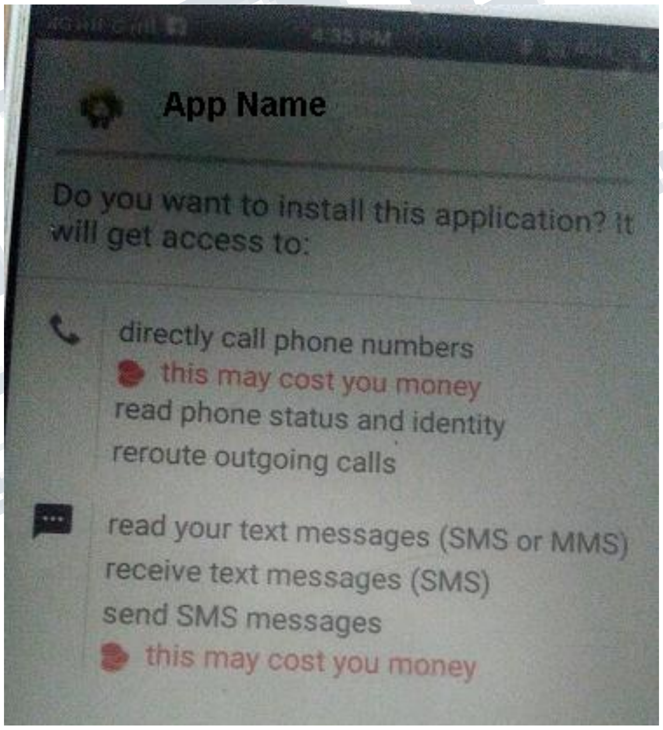


Figure 6: Confirmation Action For Call


App Name

Do you want to install this application? It will get access to:

directly call phone numbers
this may cost you money
read phone status and identity
reroute outgoing calls

read your text messages (SMS or MMS)
receive text messages (SMS)
send SMS messages
this may cost you money

Figure 7: Installing the App

We can update the permissions before installing such apps by selecting the options as below

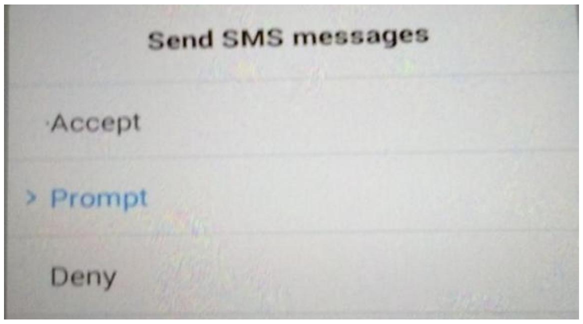

Send SMS messages
Accept
Prompt
Deny

Figure 8: Selection Steps in App

Confirmation box in case any app tries to send message in background

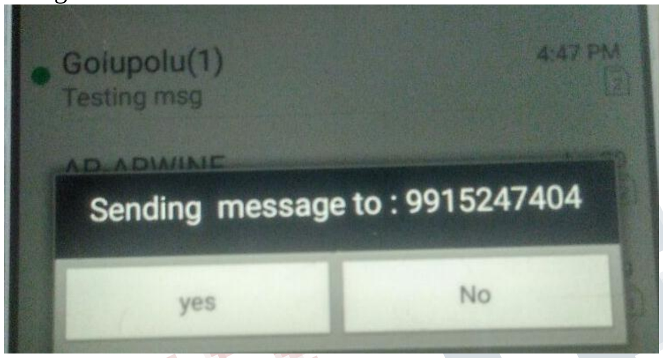

Goiupolu(1)
Testing msg
4:47 PM
Sending message to : 9915247404
yes
No

Figure 9: Sending Message Screen

VI. CONCLUSION

The new generation of smartphones is more resistant to some types of cyber -attacks that have proven extremely damaging , such as spear phishing and user -installed malicious software. At the same time, their use involves acceptance of other risks such as attacks via the cellular network, and a greater likelihood of data loss due to lost or stolen devices. Overall, vast numbers of obsolete desktops are likely to continue to be attackers' front door to networks, although smart-phones do permit highly motivated adversaries to carry out highly -targeted attacks against senior leaders. NSA continues to partner with industry to develop technological enhancements that prevent and detect such attacks. Hence in this position paper, we wish to alert the community on the imminent dangers of potential smart -phone attacks against telecom infrastructure, the damages caused by which could range from privacy violation and identity theft to emergency centre outage resulting in national crises. We have outlined defense strategies, many of which demand much further research.

REFERENCES

[1] SMS Forum, Short Message Peer-to-Peer Protocol Specification version 5.0,http://www.SMSforum.net.

[2] The Monthly Statistics in Communication Industry, http://www.mii.gov.cn/art/2006/12/22 /art_27_27537.htm.

[3] T. Walter, et al., Secure mobile business applications framework, architecture andimplementation, Information Security Technical Report, 9, Issue 4, December2004, pp. 6–21.

[4] S.P. Shieh, F.S. Ho, Y.L. Huang, An efficient authentication protocol for mobilenetworks, Journal of Information Science and Engineering 15 (1999) 505520.

[5] M. Hassinen, SafeSMS—end-to-end encryption for SMS messages, Proceedings ofthe 8th International Conference on Telecommunications, 2, June 15–17, 2005,pp. 359–365.

[6] M. Hassinen, K. Hypponen, Strong mobile authentication, 2nd InternationalSymposium on Wireless Communication Systems, Sept. 5–7 2005, pp. 96–100

[7] Almeida, T. A., Gómez Hidalgo, J. M., &Yamakami, A. (2011). InProceedings of the11th ACM Symposium on document engineering DOCENG'11 (pp. 259-262).Mountain View, CA, USA: ACM.

[8] Beaufort, R., Roekhaut, S., Cougnon, L. A., &Fairon, C. (2010). A hybrid rule/modelbased finite-state framework for normalizing SMS messages. InProceedings ofthe 48th annual meeting of the association for computational linguistics ACL '10(pp. 770–779). Stroudsburg, PA, USA: Association for Computational Linguistics

[9] https://www.defcon.org/-Bogdan-Alecu-Attacking-SIM-Toolkit-with-SMS

[10]. www.google.com/smartphones attacks and defenses.

[11].en.wikipedia.org/wiki/Mobile_security
.research.microsoft.com/enus/um/people/helenw/papers/smartphone
[13].http://www.nsa.gov/ia/_files/factsheets/mobilerisks.pdf