Security in Mobile: A Survey

Rashmi V R
P.G Student, Computer Science
Thejus Engineering College
Thrissur, Kerala

Sneha Johnson
Asst Professor,CSE
Thejus Engineering College
Thrissur, Kerala

ABSTRACT
Security is a moving target. Never mind if the pursuit of security can even be fully realized on mobile. Society at large has accepted the tenuous nature of security on personal computers as a virtually unavoidable consequence or side effect of the modern age. But compared to the PC world, mobile is still a relatively new and uncharted battlefield. Nowadays, mobile devices are an important part of our everyday lives since they enable us to access[4] a large variety of ubiquitous services. In recent years, the availability of these ubiquitous and mobile services has significantly increased due to the different form of connectivity provided by mobile devices, such as GSM, GPRS, Bluetooth and Wi-Fi. In the same trend, the number and typologies of vulnerabilities exploiting these services and communication channels have increased as well. Therefore, smartphones may now represent an ideal target for malware writers. As the number of vulnerabilities and, hence, of attacks increase, there has been a corresponding rise of security solutions proposed by researchers.

General Terms
Botnet, Smart Phone,Mobile malware

Keywords
Botnet, smart phone, mobile malware

1. INTRODUCTION
Nowadays, mobile devices are an important part of our everyday lives since they enable us to access a large variety of ubiquitous services.The modern mobile platforms, like Android, iOS and Symbian, increasingly resemble traditional operating systems for PCs. Therefore, the challenges in enforcing smart-phone security are becoming similar to those present in PC platforms.The latest research has reported that on average people own three internet-connected smart devices such as smartphones and tablets.Botnets of mobile devices (also known as mobile botnets) are a group of compromised smart devices that are remotely controlled by botmasters via command-and-control (C&C)[1] channels.Smart devices are now widely used by billions of users due to their enhanced computing ability, practicality and efficient Internet access.Moreover, smart devices typically contain a large amount of sensitive personal and corporate data and are often used in online payments and other sensitive transactions. The ide spread use of open-source smart device platforms such as Android and third-party applications made available to the public also provides more opportunities and attractions for malware creators.By installing malicious content, smart-phones can be infected with worms, trojan horses or other virus families, which can compromise user's security and privacy or even gain complete control over the device. Such malicious content can easily spread due to advances in mobile network technologies which provide smart-phones with capability of constant Internet connection over 3G or Wi-Fi networks. Additionally, the improvements in smart-phone features introduce new types of security concerns. By compromising mobile OS, malicious applications can access voice-recording devices, cameras, intercept SMS messages or gain location information. Such security breaches severely compromise user's privacy. Various approaches and solutions have been proposed to address these challenges, ranging from device based intrusion detection systems to execution isolation through application sandboxing and bare metal hypervisors to ontology based firewalls to behavior based detection to cloud-based protection via VPN technology.

2. MOBILE TECHNOLOGIES
2.1 Wireless Mobile Technologies
The most important wireless technologies targeted at mobile communications are GSM, GPRS, EDGE and UMTS[4]

2.2 Network Technologies
Bluetooth, and Wireless LAN IEEE 802.11

Bluetooth is a standard that enables devices to exchange data over a small area through short wavelength radio transmissions. IEEE 802.11 is a family of standards for WLAN that includes several protocols for communicating at different frequencies (2.4, 3.6 and 5 GHz).

3. MOBILE MALWARE
Malware is any kind of hostile, intrusive, or annoying software or program code (e.g. Trojan, rootkit, backdoor) designed to use a device without the owner's consent. Malware is often distributed as [2]a spam within a malicious attachment or a link in an infected websites. Malware can be grouped in the following main categories, according to its features.

3.1 Virus
A virus is a piece of code that can replicate itself

3.2 Worm
Worm is a self-replicating malicious application designed to spread autonomously to uninfected systems. This type of malware has been ported to mobile platforms since the introduction of Cabir. A more recent example of a worm type malware for mobile devices is Ikee.B which is used to steal financially sensitive data from jailbroken iPhones.

3.3 Trojan Horse
By deploying malicious mobile applications the attacker could gain control over the device. Such applications usually perform some useful functionality[2] while running malicious activities in the background. This way the Trojan can be used to gather private information or to install other malicious applications like worms or botnets. In addition, Trojans can be used to commit phishing activities.

3.4 Botnet
Botnet is a set of compromised devices which can be controlled and coordinated remotely. This attack strategy is

used to utilize the computing power of compromised devices in order to commit various activities ranging from sending spam mail to committing DOS attacks. An example of a botnet designed specifically for mobile devices is Waledac. Waledac uses SMS and MMS[2] messages to exchange the data between nodes therefore enabling the botnet to remain active even if the nodes are not connected to the Internet. With PC-based botnets, cybercriminals often use zombies within botnets to launch DDoS attacks. Even though there have been no major mobile DDoS incidents, with current trends we can expect to see this in the near future. Botnets are maintained by malicious actors commonly referred to as ―bot-masters‖ that can access and manage the botnet remotely or via bot proxy servers. The bots are then programmed and instructed by the bot-master to perform a variety of cyber-attacks, including attacks involving the further distribution and installation of malware on other information systems.

3.5 Rootkit

Rootkit is a malicious application which gained rights to run in a privileged mode. Such[2] malicious applications usually mask their presence from the user by modifying standard operating system functionalities. Although no current rootkit type threats for mobile devices exist, recent research efforts indicate the potential of this attack strategy and classify it as an emerging threat to mobile security.

4. THREAT MODEL FOR MOBILE PLATFORMS

4.1 Attack Goals

There are three basic motives for breaching mobile devices security. The first two goals described are covert, while the latter is harmful. Covert approach to executing an attack is to perform malicious operations[2] while avoiding users detection. The goal of such attacks is to disrupt the operation of the device as little as possible while performing activities useful to the attacker. On the other hand, harmful attacks are aimed at disrupting the normal operation of a mobile device.

4.1.1 Collect Private Data

Since the mobile devices are in effect becoming storage units for personal data, they are an attractive target for breaching users privacy. The attackers target both the confidentiality and integrity of stored information. A successfully executed attack can empower the attacker [2]with ability to read SMS and MMS messages, e-mail messages, call logs and contact details.

4.1.2 Utilize Computing Resource

Call The increase in computing resources is setting the contemporary mobile [2]devices into focus for malicious attacks with aim to covertly exploit the raw computing power in combination with operating frequencies in excess of 1GHz, and physical memory well over 512MB.

4.1.3 Harmful Malcious Action

Harmful malicious actions are aimed at generating device users discomfort rather on performing useful operations for the attacker[2].

4.2 Attack Vectors

Mobile platforms provide multiple attack vectors for delivery of malicious content. The attack vectors are classified into four categories: mobile network services, Internet access, Bluetooth, and access to[2] USB and other peripheral devices.

4.2.1 Mobile Network Services

Cellular services like SMS, MMS and voice calls can be used as attack vectors for mobile devices.

4.2.2 Internet Access

Mobile devices can access the Internet using Wi-Fi networks or 3G/4G services provided by mobile network operators. Although such high[2] speed Internet connections ensure comfortable browsing, they also expose the mobile devices to the same threats as PCs.

4.2.3 Bluetooth

Bluetooth attacks are a method used for device-to-device malware spreading. Once the two devices are in range, the compromised device[2] pairs with its target by using default Bluetooth passwords.

4.2.4 USB and other Peripherals

Apart from the mentioned attack vectors, mobile devices could be compromised by using other connections, like widely spread USB. The USB connection in commonly used to synchronize the [2]mobile device with a personal computer.

5. LITERATURE REVIEW

5.1 Android Security Model

5.1.1 Operating System Background

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android is an application execution platform for mobile devices comprised out of an operating system, core libraries, development framework and basic applications. Android OS has four layers: Linux kernel, libraries (+Android runtime), application solution and applications layers.So, basically Android runtime is a kind of glue between the Linux kernel and the applications.

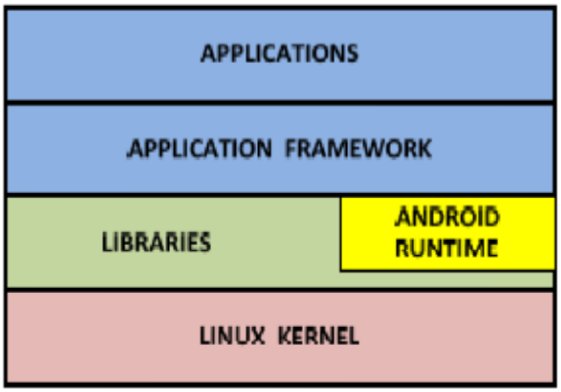

Fig:01 Android Layer[1]

5.1.2 Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own been written so that a device can run multiple efficiently. The Dalvik VM executes files in the Dalvik Executable[1] (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik VM on the Linux kernel for underlying functionality such as threading and low level memory management.

5.1.3 Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.It helps to manage security, memory management, process management, network stack and other important issues. Therefore, the user should bring Linux in his mobile device as the main operating system and install all the drivers required in order to run it. Developers have full access to the same framework APIs used by the core applications. The main security features common to Android involve process and file system isolation; application or code signing; ROM, firmware, and factory restore; and kill switches. However, the main security issue with Android OS is it relies heavily to the end-user to decide whether an application is safe or not. Android operating system is built on top of a Linux kernel. The Linux kernel is responsible for executing core system services such as: memory access, process management, access to physical devices through drivers, network management and security. Atop the Linux kernel is the Dalvik virtual machine along with basic system libraries. The Dalvik VM is a register based execution engine used to run Android applications. In order to access lower level system services, the Android provides an API through afore mentioned C/C++ system libraries.

5.1.4 Security Model

The model is based on application isolation in a sandbox environment. This means that each application executes in its own environment and is unable to influence or modify execution of any other application. Application sandboxing is performed at the Linux kernel level. In order to achieve isolation, Android utilizes standard Linux access control mechanisms. Each Android application package (.apk) is on installation assigned a unique Linux user ID. This approach allows the Android to enforce standard Linux file access rights. Since each file is associated with its owne's user ID, applications cannot access files that belong to other applications without being granted appropriate permissions. Each file can be assigned read, write and execute access permission. Since the root user owns system files, applications are not able to act maliciously by accessing or modifying critical system components. On the other hand, to achieve memory isolation, each application is running in its own process, i.e. each application has its own memory space assigned. Additional security is achieved by utilizing memory management unit (MMU),[2] a hardware component used to translate between virtual and physical address spaces. This way an application cannot compromise system security by running native code in privileged mode. Furthermore, exchange of data and functionalities between applications enhances the capabilities of the development framework. The shared user ID and permissions are two mechanisms, introduced by the Android, which can be used to lift the restrictions enforced by the isolation model. The mechanisms must provide sufficient flexibility to the application developers but also preserve the overall system security. In order to address the security issues, the Android platform implements a permission based security model, As presented in Figure 04, two applications can share data and application components, i.e. activities, content providers, services and broadcast receivers

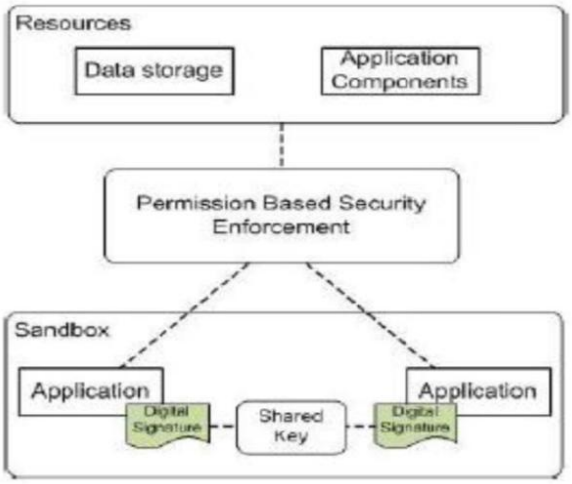

Fig:02 Android Security model[2]

5.1.5 Advantages Offered by Android

The Android Platform offers a variety of advantages not currently available in other mobile operating systems. Google opened the Android market, allowing 20 application developers to publish applications without any restrictions. Additionally, being an open platform encourages device and service provider independency. Consumers are not tied to a specific device or cellular service company to use Google Android. Android provides fully-developed features to exploit cloud-computing resources and supports a relational database on the handset [HK09]. It supports 2D and 3D graphics as well as various media file formats, allowing developers to create media common applications. The Dalvik VM significantly enhanced the power management system of the Android Platform. This custom VM takes generated Java class files and combines them in to its own native executable format. Since it reuses duplicate information across various class files, space requirements are half what the JVM .jar file requires Google also fine-tunes the garbage collection, omits the just-in-time (JIT) compiler, and uses registers instead of the stack for generation of assembly code. These enhancements significantly reduce the power requirements of the system, making the Android Platform suitable for mobile device use. Finally, Android application developers can develop applications for any platform and applications can run in parallel when loaded on the device. This allows processes running in the background to send alerts and notifications to the user.

5.1.6 Android Protection System Perfomance

The Android Protection System (APS) is a signed code modification of the Android OS 1.5 running on a smartphone device. White-list creation and hash digest placement are described in the security mechanism implementation. The evaluation technique is examined and results for functional protection and performance overhead are reported and analyzed.

5.1.7 Android Protection System Model

Proper identification of Android application code is essential for successful APS implementation. Android application code is delivered in packages called .jar or .apk files similar to .zip archives. Android applications are typically written in the Java programming language. The Dalvik Virtual Machine (DVM) operates strictly on Dalvik bytecode, so all Java bytecode is converted and stored in a file called Classes.dex, which is packaged inside the applicationspecific .apk file. The

DVM must extract the Classes.dex file from the .apk to install and run the application. Default applications come pre-installed on the Android device and all default applications are considered approved content.

5.2 IOS Security Model

Unlike the Android security architecture, iOS security model provides different philosophy for achieving mobile device security and user‟s protection. The iOS application platform empowers developers to create new applications and to contribute to the application store. However, each application submitted by a third party developer is sent to the revision process. During the revision process the application code is analyzed by professional developers who make sure that the application is safe before it is released the application store. However, such an application, when installed, gets all the permissions on a mobile device. Application might access local camera, 3G/4G, Wi-Fi or GPS module without user's knowledge.

5.2.1 System Architecture

The tight integration of hardware and software on iOS devices allows for the validation of activities across all layers of the device. From initial boot-up to iOS software installation and through to third-party apps, each step is analyzed and vetted to ensure that each activity is trusted and uses resources properly. Once the system is running, this integrated security architecture depends on the integrity and trustworthiness of XNU, the iOS kernel. XNU enforces security features at runtime and is essential to being able to trust higher-level functions and apps.

5.2.2 Secure Boot Chain

Each step of the boot-up process contains components that are cryptographically signed by Apple to ensure integrity, and proceeds only after verifying the chain of trust. This includes the bootloaders, kernel, kernel extensions, and baseband firmware.When an iOS device is turned on, its application processor immediately executes code from read-only memory known as the Boot ROM. This immutable code is laid down during chip fabrication, and is implicitly trusted. The Boot ROM code contains the Apple Root CA public key, which is used to verify that the Low-Level Bootloader (LLB) is signed by Apple before allowing it to load. This is the first step in the chain of trust where each step ensures that the next is signed by Apple. When the LLB finishes its tasks, it verifies and runs the next-stage bootloader, iBoot, which in turn verifies and runs the iOS kernel. This secure boot chain ensures that the lowest levels of software are not tampered with, and allows iOS to run only on validated Apple devices. If one step of this boot process is unable to load or verify the next, boot-up is stopped and the device displays the Connect to iTunes‖ screen. This is called recovery mode. If the Boot ROM is not even able to load or verify LLB, it enters DFU (Device Firmware Upgrade) mode. In both cases, the device must be connected to iTunes via USB and restored to factory default settings.

5.2.3 Security Model

While Android lets each user handle its own security on their own responsibility, the iOS platform makes developers to write safe code using iOS secure APIs and prevents malicious applications from getting into the app store.The iOS security APIs are located in the Core Services layer of the operating system and are based on services in the Core OS (kernel) layer of the operating system. Application that needs to execute a network task, may use secure networking functions through the CFNetwork API, which is also located in the Core Services layer. The iOS security implementation includes a daemon called the Security Server that implements several security protocols, such as access to keychain items and root certificate trust management. The Security Server has no public API. Instead,applications use the Keychain Services API and the Certificate, Key, and Trust services API, which in turn communicate with the Security Server.

- Keychain Services API is used to store passwords, keys, certificates, and other secret data. Its implementation therefore requires both cryptographic functions (to encrypt and decrypt secrets) and data storage functions (to store the secrets and related data in files). To achieve these aims, Keychain Services uses the Common Crypto dynamic library.
- CFNetwork is a high-level API that can be used by applications to create and maintain secure data streams and to add authentication information to a message. CFNetwork calls underlying security services to set up a secure connection.
- The Certificate, Key, and Trust Services API include functions to create, manage, and read certificates, add certificates to a keychain, create encryption keys, encrypt and decrypt data, sign data and verify signatures and manage trust policies. To carry out all these services, the API calls the Common Crypto dynamic library and other Core OS–level services.
- Randomization Services provides cryptographically secure pseudorandom numbers. Pseudorandom numbers are generated by a computer algorithm but the algorithm is not discernible from the sequence. To generate these numbers, Randomization Services calls a random- number generator in the Core OS layer. In case that the developers use the presented API properly and do not integrate malicious activities into the application, the application will be accepted into the App store.

6. COMPARISON BETWEEN ANDROID AND IOS

6.1 Development Environments

6.2 Language

- Android: Java
- IPhone: Objective-C

6.3 IDE

Android: Android development leverages the excellent JDT tools.

Everything Java is indexed, the IDE has a rich model of the source code, and refactoring is seamless; JDT’s incremental compiler provides immediate feedback with errors and warnings as you type.

IPhone: Xcode IDE, Instruments, iPhone simulator, frameworks and samples, compilers, Shark analysis tool, and etc.

6.4 Programming Model

Android: With Android’s support for multiple processes and component reuse, the platform itself provides support for Intents and Activities provide a way of declaring user

preferences in XML; XML format is extensible allowing custom UI components to be integrated.

IPhone: MVC design pattern, provide a way of declaring user preferences in XML; iPhone developers that wish to customize preferences will have to implement a UI from scratch.

6.5 UI Builder

Android: Android UI builder can't display UIs how they'll actually appear.

IPhone: iPhone app developers are given a good UI builder; It's flexible and can model some sophisticated UIs.

6.6 Ease to Port Third Party Applications

Android: Basically speaking, Android shares more in common with other Java platforms than with desktops with Desktop Linux. Rather than running desktop Linux PC software like Nokia's N900 running Maemo Linux, Android supplies a modified Java Virtual Machine similar in many respects to the BlackBerry OS and Symbian phones designed to run Java ME apps. Google has modified Android's Java byte code interpreter to avoid having to pay Sun licensing fees for the official JVM in Android. This enables Google to offer Android for free, and without any interference from Sun. It also effectively makes Android a Java platform, not a Linux platform. One fundamental characteristic of Android that is both strength and weakness is its insular nature. Android's unique userspace stack offers no compatibility glide path for porting applications to and from conventional Linux environments, but it does offer a significantly higher degree of cohesion across devices, which means less fragmentation and a more predictable target for third party software developers.

IPhone: Apple has taken an entirely different approach to delivering its mobile software platform. Rather than building a byte code interpreter based upon a specific, customized implementation of Java ME, Apple introduced the iPhone running a scaled down version of its desktop Mac OS X Cocoa development environment. This leverages the installed brain trust of the company's Mac developers rather than the installed base of Java ME coders in the existing smart phone market. It's still possible to port Java code to the iPhone, but it requires more translation work as Apple only supports Objective-C/C as an iPhone development language in its own tools. Rather than allowing iPhone developers to easily port over desktop Mac apps to the iPhone, the great overlap between iPhone and Mac development tools appears to have been more of strategy to draw developer attention to the Mac. Apple already sells about twice as many iPhones as it does Macs, and the iPhone certainly casts a larger mindshare net than the Mac platform does itself.

6.7 Reliability and Security

Android: Android is a multi-process system, in which each application (and parts of the system) runs in its own process. Most security between applications and the system is enforced at the process level through standard Linux facilities, such as user and group IDs that are assigned to applications. Additional finer-grained security features are provided through a "permission" mechanism that enforces restrictions on the specific operations that a particular process can perform, and per-URI permissions for granting ad-hoc access to specific pieces of data. As an open platform, Android allows users to load software from any developer onto a device. As with a home PC, the user must be aware of who is providing the software they are downloading and must decide whether they want to grant the application the capabilities it requests. This decision can be informed by the user's judgment of the software developer's trustworthiness, and where the software came from.

IPhone**:** IPhone has no security software and Apple doesn't let people load third party programs on the device, which could reduce the risk of infection from malicious software. When the iPhone is connected to the Web, dangerous possibilities emerge.The iPhone Auto-Lock disables the device's screen after a preset time period of non-use, but the Passcode Lock feature takes that a step further. Whenever the device's display locks, whether due to Auto-Lock or because you've hit the iPhone Sleep button–found on the top right of the device Passcode Lock requires a four-digit code to be entered before the device can be employed again. The iPhone OS security APIs are located in the Core Services layer of the operating system and are based on services in the Core OS (kernel) layer of the operating system. Applications on the iPhone call the security services APIs directly rather than going through the Cocoa Touch or Media layers. Networking applications can also access secure networking functions through the CFNetwork API, which is also located in the Core Services layer.

6.8 Some Important Points Clearing the Differences

Android:

1. SMS delivery report - for the IPhone you need a third party apparently

2. Notifications without INTERNET - one of the biggest drawback of the IPhone is that you cannot have notifications without Internet the notifications are stored on the Apple servers

3 Can install applications from any site - IPhone applications can only be installed from the Apple store (unless the phone is jail broken)

4 Multiple physical menu buttons - used for navigation and quick shortcuts, allows greater screen size (no more software menus)

5 Physical menu button allows recent 6 tasks (like ALT+TAB in Windows) absolutely useful

6 Can install on the Home screen - widgets, shortcuts, folders

7 Physical keyboard - on some models

8 Can install different/homebrewed firmware

9 Background apps/ multitasking

10 Dev. SDK is free and cross platform. IPhone is for $100+ and only works on MAC.

11 Programming is done in Java; bridges exist from J2ME, C#, etc. IPhone uses Objective C

12 Programming - can run interpreters. IPhone only allows running Objective C byte code

13 Easy access to the SD card (both from computer and from the phone). Can copy MP3s, read eBooks, etc.

14 Cheaper than the IPhone

15 Easy removable/replaceable battery.

IPhone:

1. Screen brightness/clarity

2. Bigger software keyboard - because of the wider screen

3. Great 3D apps and hardware

4. Easy data synchronization

5. Proximity sensor - saves battery and "locks" the screen

6. Zoom using two fingers - pictures, browser, etc - though some Android phones also support multi touch.

7. CONCLUSION

Recent advancements in mobile technology have brought the mobile devices into focus of malicious attacks. The trends show a severe increase in mobile malware as many threats, designed for PC operating systems, migrate to mobile platforms.Striking increase in malware and notable advancements in malware-related attacks, particularly on the Android platform as the user base has grown exponentially. Today's users utilize their mobile smart devices for everything from accessing emails to sensitive transactions such as online banking and payments. As users become more dependent on their mobile devices as digital wallets, this creates a very lucrative target for cybercriminals. Mobile smart device users can expect to see a significant malware increase on finance related applications, such as mobile Internet banking. Detecting and preventing malware in mobile device need comprehensive and multi-level approaches. There is a analysis attacker's goals, attack vectors and attack strategies. Furthermore, discuss about the security models implemented by two widely spread mobile platforms: the Google Android and Apple iOS. The two platforms have distinctly different approaches in dealing with security issues. The Android security model relies on user‟s judgment to install applications from reliable sources or to evaluate whether the application requests reasonable permissions for its intended operation.

8. REFERENCES

[1] Abdullahi Arabo_ and Bernardi Pranggono.Mobile Malware and Smart Device Security: **.**Trends, Challenges and Solutions, 19th International Conference on Control Systems and Computer Science, _The Oxford Internet Institute (OII), Oxford University, Oxford, OX1 3JS,

[2] G. Delac, M. Silic and J. Krolo.**2011**,.Emerging Security Threats for Mobile Platforms", International Conference on Privacy Communication System & Mobile Platforms ,Faculty of Electrical Engineering and Computing, University of Zagreb, CroatiaK.

[3] Atul M. Tonge1, Suraj S. Kasture2 , Surbhi R. **May. -** Jun. 2013.Chaudhari,Cyber security: challenges for society. IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 12, Issue 2 (), PP 67-75 www.iosrjournals.org

[4] Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra," A Survey on Security for Mobile Devices", ieee communications surveys & tutorials, vol. 15, no. 1, first quarter 2013

[5] Mohsen Sharifi, Somayeh Kafaie, and Omid Kashefi, Member, IEEE," A Survey and Taxonomy of Cyber Foraging of Mobile Devices", Ieee Communications Surveys & Tutorials, Vol. 14, No. 4, Fourth Quarter 2012.

[6] Henne, B. ; Distrib. Comput. & Security Group, Leibniz Univ. Hannover, Hannover, Germany ; Szongott, C. ; Smith, M. " Coupled multi-agent simulations for mobile security & privacy research " Date of Conference: 18-20 June 2012

[7] Hong-Il Ju ,"Security architecture for smart devices:, Mobile Security Res. Team, Electron. & Telecommun. Res. Inst., Daejeon, South Korea ; Jeong-Nyeo Kim 15-17 Oct. 2012

[8] Wanqing You, Xiamen, China ; Longteng Xu ; Jingyu Rao, "A comparison of TCP and SSL for mobile security " , Dept. of Software Eng., Univ. of Xiamen, 18-19 May 2013

[9] Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra, "Survey on Security for Mobile Devices**",** IEEE communications surveys & tutorials, vol. 15, no. 1, first quarter