International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-3, Issue-3)

Research Article | **March 2014**

Android Security –Big Challenge

Mis.Prajakta S. Deshbhratar[1], Prof. Mayur S. Burange[2]
P.R.Pote Collage of Engineering ,Amravati
SGBA University ,Amravati, India

Abstract-

Smartphone are now very popular. Aside from calling and texting, people use them for connecting with their digital life–email, social networking, instant messaging, photo sharing and more. With that Smartphone store valuable personal information such as login credentials, photos, emails and contact information. The confidentiality of that data is of paramount importance to the user because it might be abused for impersonation, blackmailing or else. Smartphone are very attractive for attackers as well: First, attackers are interested in the precious private information. Second, Smartphone are constantly connected, which makes them useful as bots in bonnets. Third, Smartphone can send premium SMS or SMS that subscribe the victim to costly services, and thus directly generate money for the attacker. It is up to the Smartphone operating system (OS) to ensure the security of the data on the device. In the last two years Android became the most popular mobile OS on the market. With over 1.5 million device activations per day Android is expected to cross the one billion active device barriers in 2013. Its worldwide market share has reached 70 percent of all Smartphone. In this paper we systematize the knowledge about the Android security mechanisms and explain about how attack can be avoided when handling a android mobile operating system.

1. INTRODUCTION

An Android is an open source operating system, key mobile applications having API libraries for executing android applications. Android smart phones offers advanced computing ability and connectivity as compares to other mobile phones operating systems. Android is operating system which designed hardware so that communication between hardware and software with user interface can easily be done. Google has released tool i.e. Google apps that implement under some security policies. There are so many facilities like password protection also implement in android smart phones. Android is Linux based operating system. The architecture of android operating system is designed in such manner so that communication at application level and end user will be quite easy. Android applications are written in Java, a programming language. But Android has its own virtual machine i.e. DVM (Dalvik Virtual Machine), which is used for executing the android applications.

Android applications also run on non-mobile household devices like oven, AC, refrigerators, washing machines etc. Designing of android application is easy as compared to other applications of iphones. Android operating system is also used for business purposes. At business level, risk will be high while transferring data from one end to another end. As Android provides remote access to official sensitive data, which can lead to data hack if smart phones are hacked into. For this security purpose, Google has designed their operating system to execute applications in specialized sandboxes, which minimize the capability of malware attacks to the applications in smart phones.

1.1 Android Encryption:

Disk encryption on Android operating system is based on dm-crypt, which is kernel feature that works at block device layer. It works with flash devices which present themselves to the kernel as a block device. The file system to use on these devices is ext4. It is an independent either encryption is used or not. Linux kernel is doing the encryption work. The actual encryption used for file system for first released was 128 AES with CBC. The master key is encrypted with 128 AES to the openssl library. By adding new module to void and its components, it will invoke encryption features. There are commands like checkpw, restart, changepw and crypt complete used for encryption on android operating system.

1.2 How to enable encryption on a device?

Android is designed having multi layer security which provides flexibility for this platform. When attackers attempt attack on device, android platform help to reduce the portability of the attack. There are key components of android security which are described as follows:

a) Design review: when a security model is designed then it will be reviewed by the developers so that risk level will be very less while using the model. When risks come, immediately teams for controlling risk factors will start work on that to reduce the risks level

b) Code review and penetrating testing: the goal of this code review is that in which it will be checked that how the system will become strong?

c)Open source and community review: android uses open source technologies that have significant external review such as Linux kernel.

d) Incident response: android team enables the rapid mitigation of vulnerabilities to ensure that potential risks to all android users are minimized.

2. LITERATURE REVIEW

In [1],The sheer number of mobile applications at a time when the technology in mobile security is still in its infancy presents complex, multifaceted, and unprecedented security challenges to enterprises while putting individual privacy at high risk. As attackers adapt to static security controls, we need to lever-age the collective intelligence of the community, private networks, IPSs, and other sources of data to identify and arrest malicious applications at network speed. Mobile security has certainly been one of the most pressing issues in enterprises—the September/October 2013 issue of this magazine will focus on this topic extensively. On the horizon, we see a world in which attackers set their sights on mobile devices in a relentless push to compromise networks. It has long been the calculus of the information security industry to wait for attackers to act and then to react, but with an agile and collaborative approach to mobile security, we might be able to flip that equation.

In[2], Hackers recently created a potential security prob-lem by stealing digital certificates for several major websites, including Skype, Google's Gmail, Microsoft's Hotmail, and Yahoo Mail. The attackers obtained nine SSL certificates generated by security vendor Comodo. On 15 March, Comodo reportedly realized the theft had taken place, revoked the certificates, and contacted browser makers Google, Microsoft, and Mozilla. Google patched its Chrome browser on 17 March; Mozilla and Mi-crosoft updated Firefox and Internet Explorer on 22 and 23 March, respectively. Comodo said evidence indicates Iran's government launched the attack, perhaps to set up fake websites from which authorities could identify and monitor antigovernment activists

In[3], Linux provides several access control mechanisms. The basic element of these mechanisms is users (that is, entities). Users (represented by an integer number or user id) own objects (a process, file, or directory). Users are further assigned to groups. In the Linux file permissions mechanism, each file is associated with an owner user and group IDs and three tuples of read, write, and execute (rwx) permis-sions. The kernel enforces the first tuple on the owner, the second on users belonging to the group, and the third on the remaining users. Files in Android (both application and system files) are subject to the Linux permission mechanism.

In[4], MOSES is the first solution to provide policy-based security containers implemented completely via soft-ware. By acting at the system level we prevent ap-plications to be able to bypass our isolation. How-ever, at the present moment MOSES has also some limitations. At first, fine-grained policies and allowed applications are specified using the UID of an applica-tion. Meanwhile, in Android it is possible that some applications share the same UID. Thus, if we apply MOSES rules and restrictions to one application they automatically will be extended to the other ones with same UID. Furthermore, some fine-grained policies in MOSES are built on top of Taintdroid functionality. Thus, MOSES inherits the limitations of Taintdroid explained in Section 3. It should be also mentioned that the applications that have root access to the system can bypass MOSES protection. Thus, MOSES is ineffective in combating with the malware that obtains root access, e.g., rootkits.

In[5], In many ways, Android pro-vides more comprehensive security than other mobile phone platforms. However, learning how to effectively use its building blocks isn't easy. We're only beginning to see different types of applications, and as Android matures, we'll learn how faulty application policy affects the phone's security. We believe that tools such as Kirin and those like it will help mold An-droid into the secure operating system needed for next-generation computing platforms.

3. ANDROID SECURITY CHALLENGES

3.1 Android Platform Security

Android runs on a wide range of devices and Android's security architecture relies on security features that are embedded in the hardware. The security of the platform depends on a secure boot process. Secure Boot The boot process of an Android device is a five-step process. First the CPU starts executing from its reset vector to which the initial boot-loader (IBL) code from the ROM is wired. Then the IBL loads the bootloader from the boot medium into the RAM and performs a signature check to en-sure that only authenticated code gets executed. The bootloader loads the Linux kernel and also performs a signature check. The Linux kernel initial-izes all the hardware and finally spawns the first user process called init. Init reads a configuration file and boots the rest of the Android user land.

Rooting In general, mobile devices are subject to strict scrutiny of the mobile operators. That is it employs secure boot to ensure that only code is being booted, that has received the official blessing in the form of a certification from the operators. This is being done to ensure that the mobile OS's security measures are implemented and the device does not become a harm to the cellular network. Rooting involves a modification to the system partition. Modifications to the system partition require root permissions, which are not available by default. There are two ways of obtaining root permissions: Either the customer boots a custom system that gives him a root shell, or he exploits a vulnerability to obtain root permissions at runtime. Rooting, voluntarily or involuntarily has repercussions on device security. Unsigned kernels can contain malware that runs with full permissions and is undetectable by anti-virus software (rootkits). Further, rooted devices do not receive over the air updates. If an application has received root permissions, it can essentially do as it pleases with the device and its data, including copying, modifying and deleting private information and even bricking the device by overwriting the boot-loader.

3.2 Android System Security

The flash storage of an Android device is usually divided into multiple partitions. The system partition contains the Android base system such as libraries, the application runtime and the application framework. This partition is mounted read-only to prevent modification of it. This also allows a user to boot their device into a safe mode which is free of third party software. Since Android 3.0 it is possible to encrypt the data partition with 128bit AES. To enable file system encryption the user has to set a device password which is used to unlock the master key. Data Security By default an application's files are private. They are owned by that application's distinct UID. Of course an application can create world readable/writable files which gives access to everybody. Applications from the same author can run with the same UID and thereby get access to shared files. Files created on the SD card are world readable and writable. Since Android 4.0 the framework provides a Keychain API which offers applications the possibility to safely store certificates and user credentials. The key store is saved at /data/misc/key store and each key is stored in its own file. A key is encrypted using 128-bit AES in CBC mode. Each key file contains an info header, the initial vector (IV) used for the encryption, an MD5 hash of the encrypted key and the encrypted data itself. Keys are encrypted using a master key which itself is encrypted using AES.

3.3 Android Application Security

In Android application security is based on isolation and permission control. In the picture you can see, that there are processes that run with root priv-ileges. Zygote is the prototype process that gets forked into a new process whenever a (Java) application is launched. Each application runs in its own process with its own user and group ID which makes it a sandbox. So, by default applications cannot talk to each other because they don't share any resources. This isolation is provided by the Linux kernel which in turn is based on the decades-old UNIX security model of processes and file-system permissions. It is worth noting that the Dalvik VM itself is not a security boundary as it does not implement any security checks. In addition to tradi-tional Linux mechanisms for inter-process communication Android provides the Binder [8] framework. Binder is an Android-specific IPC mechanism and remote method invocation system. Binder consists of a kernel-level driver and a user space server. With Binder a process can call a routine in another process and pass the arguments between them. Binder has a very basic security model. It enables the identification of communication partners by delivering the PID and UID. Android Permissions On Android services and APIs that have the potential to adversely impact the user experience or data on the device are protected with a mandatory access control framework called Permissions. An application declares the permissions it needs in its AndroidManifest.xml such as to access the contacts or send and receive SMS. At application install time those permissions are presented to the user who decides to grant all of them or deny the installation altogether. Permissions that are marked as normal such as wake-up on boot are hidden because they are not considered dangerous. The user however can expand the whole list of permissions if he wants to.

3.4 Android Security Enhancements

With Android 4.2 and the following minor releases Google introduced new security features in Android. We will present a small selection of these enhancements in the following paragraphs. The user now can choose to verify side-loaded applications prior to installation. This is also know as the on-device Bouncer. It scans for common malware and alerts the user if the application is considered harmful. So far the detection rates don't measure up with other commercial malware scanners [5]. With Android 4.2.2 Google introduced secure USB debugging. That means only authenticated host de-vices are allowed to connect via USB to the mobile device. To identify a host, adb generates an RSA key pair. The RSA key's fingerprint is displayed on the mobile device and the user can select to allow debugging for a single session or grant automatic access for all future sessions. This measure is only effective if the user has a screen lock protection enabled. Prior to Android 4.2 the optional exported attribute of a Content Provider defaulted to true which hurts the principle of least privilege. This lead to developers involun-tarily making data accessible to other apps. With Android 4.2 the default behaviour is now "not exported". SELinux on Android The SEAndroid project [15] is enabling the use of SELinux in Android. The separation guarantees limit the damage that can be done by flawed or malicious applications. SELinux allows OS services to run without root privileges. Albeit SELinux on Android is possible it is hard to configure and it slows down the device. Samsung Knox has been announced to actually roll-out SEAndroid on commercial devices.

3.5 Android Security Problems

F-Secure Response Labs 96% of mobile malware that was de-tected in 2012 targets the Android OS [11]. In this chapter we want to shed light on the security weaknesses of Android that enabled such a vibrant market of malware. In short, Android has four major security problems: First, security updates are delayed or never deployed to the user's device. Second, OEMs weaken the security architecture of standard Android with their custom modifications. And third, the Android permission model is defective. Finally, the Google Play market poses a very low barrier to malware. We will now detail each of these problems. Android Update Problem There are four parts of the system that can contain vulnerabilities: the base system containing the kernel and open source libraries, the stock Android runtime including basic services and the Dalvik runtime, the Skin supplied by the OEM and the branding. The Android base system and runtime are published with full source by the AOSP. This code is the basis of all Android based smart phones. Any vulnerability found therein can potentially be used to subvert countless Android devices. In other terms, a vulnerability has a high impact. In practice, updates are very slow to reach the devices, with major updates taking more than 10 months [3]. Many vendors do not patch their devices at all, as the implementation of a patch seems too costly [4].

According to Google Inc.'s own numbers, the most recent version of Android is deployed to only 1.2% of devices [2]. To remedy this problem, Google announced an industry partnership with many OEM pledging to update their devices for 18 months. This partnership is called the Android Update Alliance. However, there has been no mentioning of the alliance since 2012, and updates are still missing [3]. Bringing the updates to the devices is more involved however. Once the update reaches the OEMs, they incorporate it into their internal code repositories. For major updates, this includes porting their Skin forward. A faulty firmware update has very bad consequences for the OEM's reputation. Therefore the updated firmware is subject to the OEM's quality control. In summary, incorporating an update into a device firmware is therefore very costly to the OEM both temporal and financial. Cellular operators certify devices for correct behaviour. This is done to ensure that the device does not misbehave and therefore does not put the network and its users at risk. Updated firmwares need to be re-certified before they can be deployed. Depending on the operator this can take a substantial amount of time. For example re-certification at T-Mobile takes three to six months [12], other carriers opt out of the process and do not ship any updates at all.

4. CONCLUSION

In this paper we investigated the security of the Android mobile OS. We described the Android security measures, and its problems. We derived a set of lessons learned that will help future mobile OS designers to avoid pitfalls. With existing online services. However, as the importance of the data and services our cell phones support increases, so too do the opportunities for vulnerability. It's essential that this next generation of platforms provides a comprehensive and usable security infra-structure.

REFERENCES

1. Qing Li and Greg Clark |Blue Coat Systems "MobileSecurity:ALookAhead"January/February 2013 Copublished by the IEEE Computer and Reliability Societies 1540-7993/13/$31.00 © 2013 IEEE
2. David Ladd, daveladd "NewsBriefs Mobile security"copublished by the ieee computer and reliability societies 1540-7993/11/$26.00 © 2011 IEEE MAY/JUNE 2011
3. AsAf shAbtAi, YuvAl fledel, AndYuvAl elovici "Securing Android-Powered Mobile Devices Using SELinux" Copublishe B The Ieee Computer An Reliability Societies 1540-7993/10 /$26.00May/June2010
4. Yury Zhauniarovich, Giovanni Russello "MOSES: Supporting and Enforcing Security Profiles on Smartphones" 1545-5971 (c) 2013 IEEE.
5. WilliamEnck, machigar Ongtang, andPatrick mcdaniEl "understanding android security" ieee seCurity & PrivaCy January/February 2009