A Security Overview in Google's Open Source Android Phone

Sunitha Medayil Vijayamma
School of Computer and Information Science
Edith Cowan University, Western Australia, Australia
smedayil@ecu.edu.au

Abstract

As the increase in the number of mobile phones in the market, people tempt to use many technological innovations. Recently, Google has developed an Open Source Android phone running in Linux platform. The security architecture of Android phone was well constructed with 'sandbox' approach and there were no impacts to attack the phone. But recent news released that the phone has multiple vulnerabilities and security holes by which the attackers could easily access the browser information stored on the phone because of its open nature. The purpose of this paper is to study the security aspects of Android phone in two different ways. Firstly the relevant features of the security framework in the Android phone rather than in other phone by analysing security in three approaches to find out how can be the phone vulnerable for an attack. And secondly about the existing vulnerabilities and security holes in the Android's open source platform that exploit by an attacker to get the phone. Overall, 'A Security Overview in Google's Open Source Android Phone' mentions the security aspects as well as issues prioritized in the Android open source platform.

Keywords

Android, Security model, analysis, vulnerabilities, malware

INTRODUCTION

A world without Mobile devices is not imaginable because of its mobility and easy access to information. In addition, it has become a part of our daily lives for communication and other activities that the Personal Computer does. Considering the historical development of Mobile phones is a good idea to know the importance of this among many other inventions. The first mobile network was introduced by USA in 1946 (Speckmann, 2008) for military purposes and there after in 1973, the first cellular phone was invented by Motorola. Since then there was a flow of mobile devices in the commercial market. However, the cell phones in the past decades have not included many technological features that the currently available phone covers.

But in 2007, Apple invented an impressive phone named 'iPhone' with touch screen interface and web browser facility. There onwards a number of phones have come and all are still available in the market. But Google's newly invented G1-Android phone has a most challenging interface that any other phones have. Actually Google has started working with the Mobile Phone platform development from July 2005 and the dream project was finished only in 2007. Finally, after the second half of 2008, Google announced the newly developed T-Mobile's G1 Android phone and the phone have become most popular because of its unlimited functionality. The following figure illustrates the historical development of cell phones (This does not contain all phone manufacturer's information).

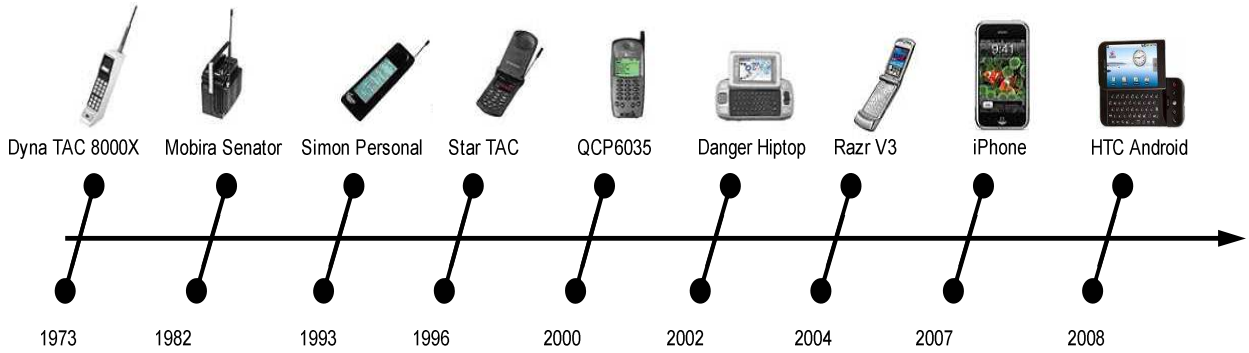

Figure 1: Historical development of cell phones (Speckmann, 2008)

This paper discusses Google Android Phone in detail. Section 1 provides the framework of Android phone and its features. Section 2 entitles the security model of the phone that depicts how the Android phone manages the security in its processing environment. When discussing the Security features, the issues are also inevitable, Section 3 will specify the security holes and vulnerabilities in the Phone and a Scenario has included that how an

attacker exploit a major vulnerability in Phone to hack the information. The report concluded with the implementation of technical measures to avoid the security issues.

ANDROID- AN OVERVIEW

"Android makes it easier for consumers to get and use new content and application on their handset"

- Andy Rubin (Director, Google Mobile Platform)

G1-Android is the first commercially available phone that features Google's Android Platform SDK. This is a partnership project with Taiwan Based HTC Corp and it is supported by the United States Service provider T Mobile. Thus the phone is available in the market as T-Mobile HTC Android. According to the Android Publication (2009), Android is the first truly open source platform for mobile devices with a fully integrated software stack that consists of an operating system, middleware, user friendly interface and applications, and also allows the users to develop additional software and change or replace functionality without limitations. In order to achieve the unlimited functionality, Android uses Linux Operating system and it is clear that individual's can experience same internet activities equal to what the people can experience on a desktop PC.

Speckmann (2008) pointed out that Android Framework is a very complex architecture so that it operates in four layers. As part of this paper work, we will be concentrated only on the security side of the framework.

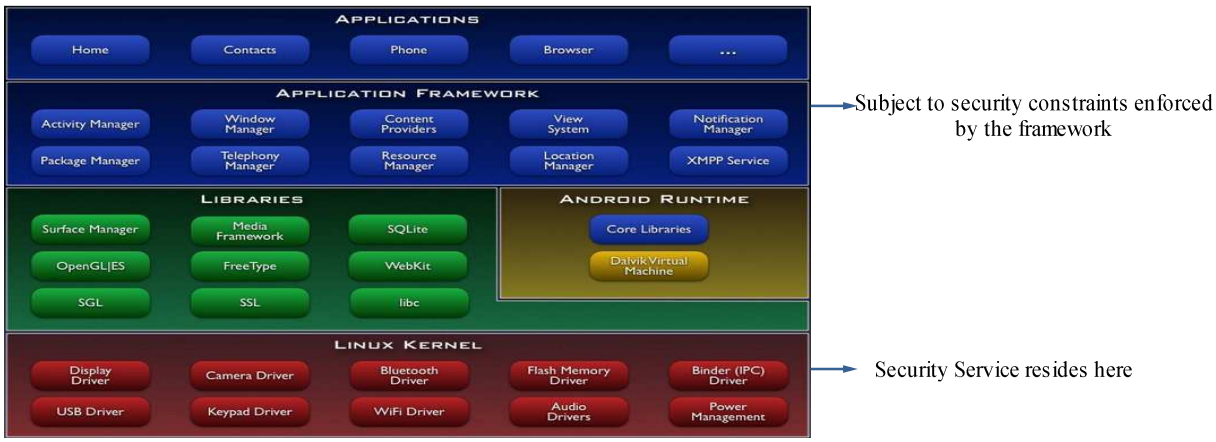

Figure 2: Architecture of Android (Speckmann, 2008)

Android Architecture

Android's basic applications such as browser, email client, SMS, calendar and many other applications resides in the top most Application layer. A significant advantage of Android Phones is its new integrated Browser based on Open Source Web Kit engine that allows users to access web pages the same way as through the Personal Computer. The Application framework in the next layer supported by a number of open source libraries, also, Android security constraints are enforced in this layer. Android has included a set of core libraries in the Android Runtime sub layer and every Application runs on Android has its own processes with its own instance of the Dalvik Virtual Machine.

Android relies on Linux Version 2.6 in the Linux Kernel layer (Android, 2009) for security services, memory management, process management, network stack and drivers. This helps to manage security, memory management, process management and installation of new drivers into the mobile platform. All application runs on the Android phone is subjected to the security constraints enforced by the Application framework. All these features in the Android phone increase the stability and reliability of the Processing.

HOW ANDROID SECURITY MODEL WORKS

The Open Source Android phone has a significantly different model rather than other commercially available Mobile Phones in the market. Typically, the security in Android resides on the Linux Kernel and uses a method such as Sandbox approach.

Security Overview in different OS Mobile

To analyse the security feature of Android, we compared the traditional security methods used in different Mobile Operating system platforms such as Symbian and Windows mobile (Speckmann, 2008). We can see from the following table that how an Android security mechanism differs from other phone OS.

Phone Model	Security Mechanism
Google Android	Linux Facilities (User and Group ID) Permission Level Security
Windows Mobile	Security Policies Roles and Certificates
Symbian OS	Certificate Management and Cryptography

Table 1: Security features

Symbian Operating system uses a Certificate Management and Cryptography techniques to defend against malicious and unwanted programs. Windows Mobile has its own security model with s combination of security policies, roles and certificates to access the data on the device. That is; all access to the data is controlled by a built in security policy. In contrast to these, Android uses a different security model in which the application runs on the Android platform is controlled by providing Linux User Identifiers (UID) and special permissions (Spectrum, 2008).

For analysing the security features of Android phone we used three different approaches in accordance with the security mechanism used in the phone.

Phase1: Security Analysis with Android

In this approach, we used the security infrastructure of Android phone and traditional Windows mobile platform. The major difference of the security in Android is its 'Secure Sandbox' approach. This is considerably a different security model than any other phones in the market. The open nature of Android phone and well constructed sandbox security provides a quite different way of accessing the data on the phone. Unlike Windows platform, Android's Applications runs on its own instances of the Dalvik Virtual machine in which each instance represents a Linux kernel process. And the instances running on each application is completely isolated from the other application and memory. Each application of Android has a unique User Identifier (UID) and file permissions to access the database and file on the phone. In this way, programs and applications runs on the Android platform cannot disrupt other processes in other application (Enck, & McDaniel, 2008). In the Windows mobile security method, all applications are running in a process with same user identifiers and the security is through the built in security policy resides on the phone. Thus, there is no additional security provided to access the file and databases on the phone.

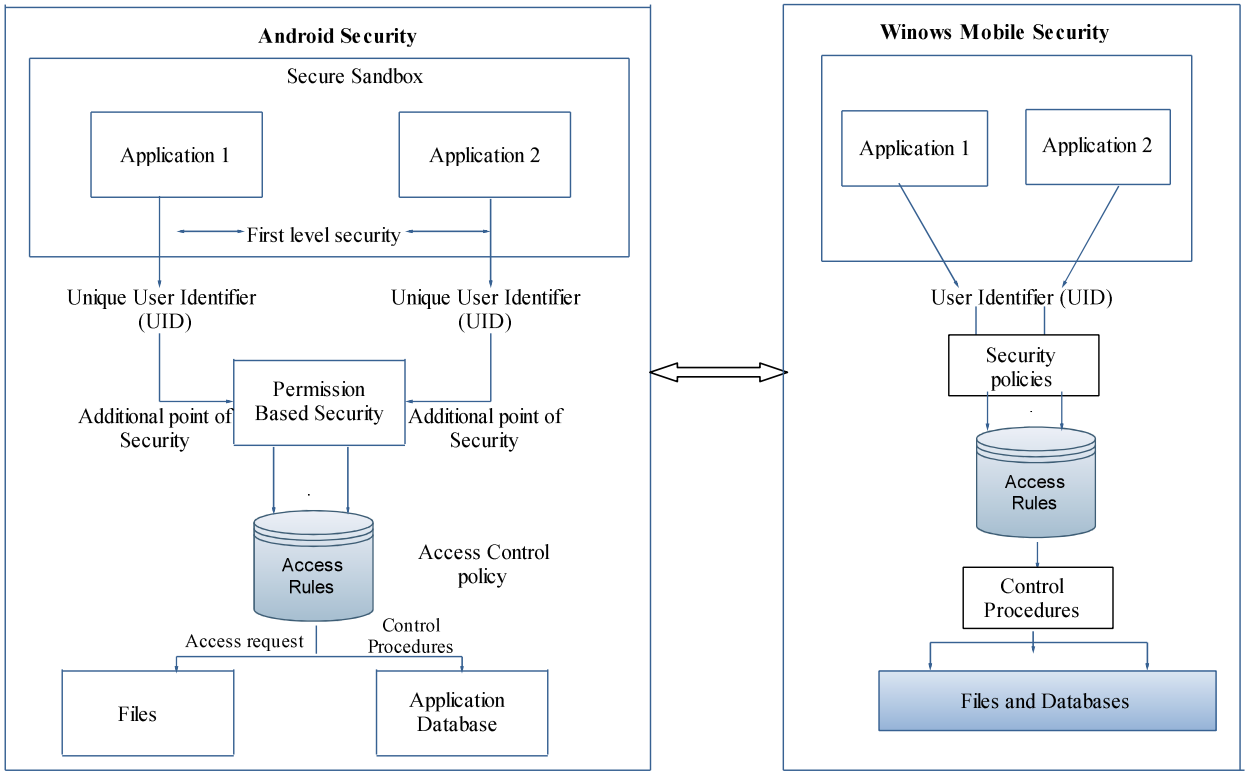

Figure 3: Phase 1 Security Analysis

When we run three processes such as dial the phone, take the pictures and Use GPS in three applications in the top most layer of Android, it runs with three unique user identifiers. The additional point of security is provided in the permission based level; there it provides a proper access privilege by using an access control policy mechanism (Spectrum, 2008). This enables to access the data only by certain users. The following example denotes the processes from different Applications by providing rights to access the information in a method of general permission. This permission handled by automatically allowing and disallowing based on the certificates from the Android application developer. So when a user use GPS application and dial the phone simultaneously, it will not disrupt each other and the security is maximum in this point of level.

ACCESS_GPS (Provide Rights to access information in a method of general permission)

ACCESS_CONTACTS (Provide Rights to access information in a method of general permission)

ACCESS_EMAIL (Provide Rights to access information in a method of general permission)

READ_CONTACTS (Providing rights to interact with user's list of contacts)

WRITE_CONTACTS (Providing rights to interact with user's list of contacts)

By default, the Android phone enforces security with the Access control settings in the lowest layer, only the right user ID with permissions can be allowed to access the files and databases on the particular application. However, the other application's interface is invisible to the user running on a particular application. These settings can be involved in the Google code libraries settings, unfortunately any developer can write and modify the settings because of the openness of the phone platform.

The above security approach in Android phone provides considerably a good security than any other phones. Because of its isolation from other applications running on the Sandbox, there is no possibility that an attacker can steal information running on one application from another running application. Thus the security model used in Android is better than other operating systems.

Phase 2: Security Analysis

In this security approach, we analyse how secure the Android phone when permission is granted between two different applications. According to Android Publication (2009), a particular permission can be enforced at number of places during the program's operation. These are enforced to:

- At the time of a call into the system, to prevent an application from executing certain functions
- When starting an activity, to prevent application from launching activities of other application
- Both sending and receiving broadcasts, to control who can receive our broadcast or who can send a broadcast to us
- When accessing and operating on a content provider
- Binding or starting a service

Suppose, we are running one application process and wants to access other application we need to share the User Identifiers between the applications. In this instance, instead of different User Identifier we are sharing the same User Identifiers for different applications. This can be achieved by using a Content Provider. Content providers provide an additional level of security by giving permissions applied between applications, which restrict the access to the data by a certain user. Enck, & McDaniel (2008) noted that when a user shared the User Identifier with other application that provide a weak link of security. Because of the open nature of the Android phone and no centralised control for the applications running on the phone, there are several points of security vulnerability. The applications running on Android uses a self signing certificate from the Application developer, in the event of sharing the information between applications, user Identifiers in both applications are signed by the same authority. This causes impact to the permission based security in different ways:

- First, the Application can give a certain type of permission to visible all database and files to other application by a general permission
- When two application share the same User Identifier, they declare the same User ID for both applications that are signed by same authority (same developer)

In this method, user can share the different applications with same User Identifier:

- ACCESS_CONTACTS, ACCESS_CELLID, ACCESS_GPS (Providing rights to access information in a method of general permission)
- READ_CONTACTS and WRITE_CONTACTS (Providing rights to interact with user's list of contacts)

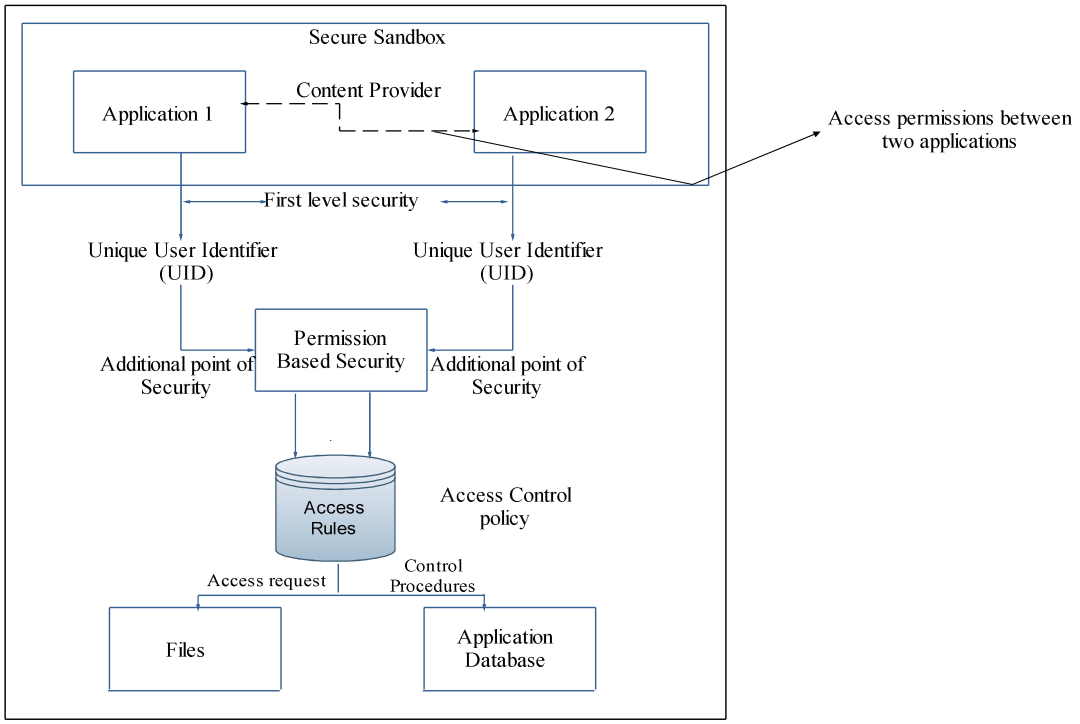

Figure 4: Phase 2 Security Analysis

This security approach is not so secure when handling sensitive data. According to Spectrum (2008), Android uses less memory for security and permissions, and only way to avoid this by allocating memory to the sensitive information, could still be used by a potential hacker to break into secure Content Provider service. So this method cannot provide a good security when sharing information between applications.

Phase 3: Security Analysis

There are many security discussions still undergoing on the Android security discussion forum that no one is clear whether the security certificates used in Android phone is secured or not. For this purpose, we applied the security principle from Public Key Infrastructure to the Android phone's permission based security. There are many security implications to note that Certificates used in Android phone. According to Ellison & Schneier (2000), there are several risks when we are using Certificates in a system. They pointed out that the public keys used for certificate verification are vulnerable, because they are public keys and there are no secrets to protect. However, if an attacker compromised a system and adds his own public key, he can issue his own certificates, that will looks exactly as a legitimate certificate. These self signed certificates does not provide any more security instead, it will be a potential hole to the attacker to access the contents. The following process depicts how a malicious code writer compromises a phone and targets the sensitive information from the victim's phone.

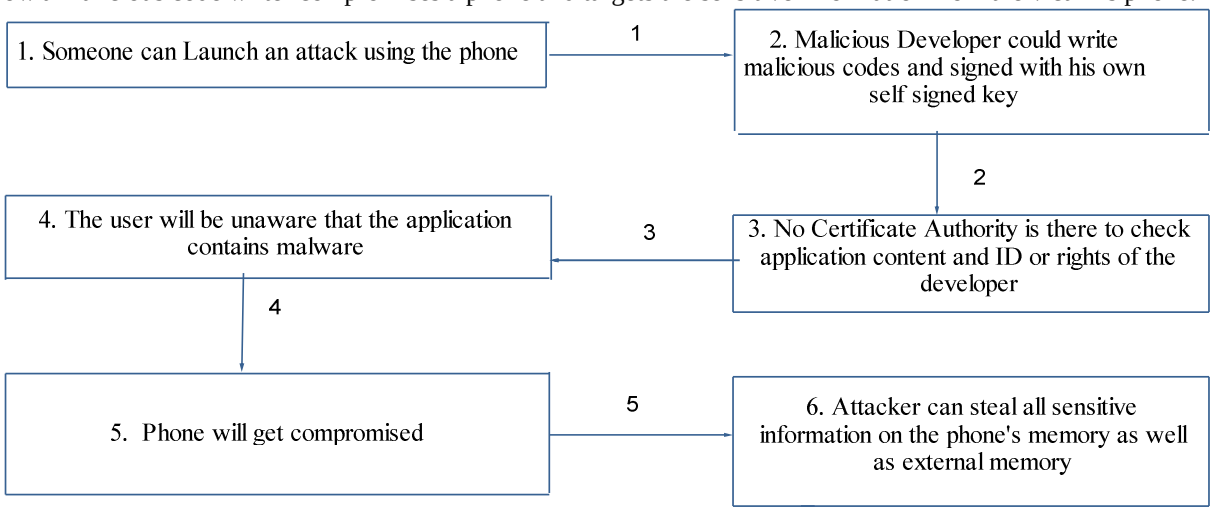

Figure 5: Phase 3 Security analysis

Because of the open nature of Android and self signed certificate mechanism for applications, any users can write the application and signed with the same key they have. There is no central control for certificates and this causes many risks in the mobile phone platform by means on an uncomplicated interface and permissions to update the applications on the phone. When the applications are sharing the same user identifiers with the same key allows the attackers to compromise the phone and use information for theft.

In this way, malware writers can easily install the source code into a mobile with self signing certificate because of the open nature of Android phone. From the three phases of security analysis, we can realise that granting permissions to a particular application will provide an additional security. Also, each applications running on its own UID will minimize the malicious threats from a source code developer. However, in the second phase, when sharing a UID for different applications can have potential impact of entering unprivileged software to a phone, when a hacker get access to one application (Enck & McDaniel, 2008). The most controversial use of self signed certificate verification for a new application allows the malicious code writers to access a user's phone, and the user is not at all aware of the fact that his phone gets compromised. So the permission based security is not a good security strategy used in Android.

Major issues in Android

The resultant security achieved from the Android phone is controversial in some aspects. It provides certain level of security than other phone but it lacks security in several ways.

- Android core libraries are open and any user can write their own applications that pose security threat from malware, Trojans hidden in animated images.
- Android' s self signed certificate for verifying applications can also be a target for hacker to add their own private key for applications and no centralised control for applications and certificates
- Permission based security adversely impact a user's privacy by reading or writing user's private data such as emails, contacts, read or write another application files, and performing network access (Enck & McDaniel, 2008), use GPS to track a person's location without his knowledge.
- The open source Android platform does not run on encrypted file system and has a vulnerable login. Android also has a built in unencrypted browser and the information sends via network is not secure (Mocano, 2008).

VULNERABILITY EXPLOITATION IN ANDROID

The vulnerabilities exposed with the current Android deployment are dangerous when sharing information on the network. There are several factors affects the security issues in Android. The openness nature and its incompatibility with other software products available in the market, makes a minimum level of defending against malware and Trojans. Rothman (2008) noted that Android is not so secured than Apple's iPhone, because iPhone has implemented VPN and many security goodies to secure the phone. The openness nature of the phone is truly operated like Windows Vista Operating system, in which the system asks to grant permission for each and every capability that an application desires. The working environment of Android is just like a Desktop Computer, so that the Google developers must had to implement the security mechanism as like the traditional computer to secure the phone from malicious developers. In this point, Google does not have any excuse for vulnerability exposal found in the phone.

The incompatibility and the number of open source Anti-virus software tools in the market show a more dangerous situation to avoid the risks. From the security software available in the Insecure.org 'Top 100 Network security tools' (Insecure.org, 2006), some of the open source applications will work with the current Android deployment and the major part are not compatible.

Open Source Software	Compatibility	Functions	Approximate Storage Space needed
Antivirus- Clam	Partially works	Email scanning	28 MB
Firewall- Netfilter	Partially works with current Linux kernel in Android	Which reside in the Linux kernel and filter packets	-
Rootkit Detector- Chkrootkit	Partially compatible	Scan root kits and Trojans	588KB
Intrusion Detection- Snort	Not works	Traffic analysis and packet logging on IP network	-
Open SSH	Partially works	Make sure the password is encrypted while using internet	10688 KB

Table 2: Security tools

According to Spectrum (2008), current version of Android deployment lacks security in several ways, because Android Open source provides less memory for security. For this reason, some of the available open source Antivirus and other security goodies will not completely work with the current version of Android.

Malicious attacks in Android

The open nature of Android phone is one of the major defects that an attacker exploits to attack. The open core libraries in the third layer are a potential target that a malicious code writer to install the malicious software such as viruses and the user is unaware that it contains malicious contents. When they run the application, the phone get compromised and make a facility for attacker to exploit it. Once it compromised, attackers can access the saved credentials from the user's phone without their knowledge. Charles Miller, a security researcher from Independent Security Evaluators, presented in a conference in Washington D.C that Android's Browser is vulnerable that a hacker can get the control of the phone remotely (Mills, 2009). This is because of the Open source core libraries used in Android, and which is complex that lead to bugs quickly. There are several reasons that the Google phone gets attacked, first, the direct user strategy that any user can install applications to mobile core libraries and then the unencrypted browser used in the phone's platform. Attackers can install malicious code in to the Google open source core libraries and redirect the web pages to the malicious sites from whatever we type in the address bar. When a user tries to visit a particular web site, it is going to another web site without any notification; it is potential to the phone to get compromised and ready for an attacker's exploitation. This is because of the open source code vulnerability. In the second security issue, the malicious web pages that a user accidentally visits from the unencrypted browser. The real fact about android's malicious vulnerability is the user's unawareness that they accidentally giving permissions to a malicious application to enter the device. However, Android is more secure than other operating systems that it uses a sandbox approach that stops malicious code injected into browser from accessing and taking to other parts of mobile OS.

From this point, we can sure about any type of malicious attacks are possible when a user access the network from unprotected browser. If a hacker sets a malicious web site and tricks the user to visits it, then the hacker can run the malicious code and take over the browser to capture the keystrokes entered by the user when surfing to other sites. Also, the attacker could have access to all the information stored on the browser's memory such as cookies, password, and the data on the web applications. So in comparison with computer based attacks, attacker can easily attack the phone user by so called 'Phishing attack'.

How Malware based Phishing attack accomplished in an Android Phone

Phishing is an online identity theft in which online identity information is obtained from an individual to a hacker (Ciampa, 2007). These threats are mainly from malware based phishing and DNS server attacks. Malware based phishing refers generally to any type of phishing that involves running malicious software on a user's phone. Whenever the user is online, the phone is vulnerable to a hacker. Because of the permission based security used in Android phone, accidentally a user runs the malicious web sites from an attacker (Enck, & McDaniel, 2008). The following diagram depicts how an attacker tracks the information from a victim's phone to his computer.

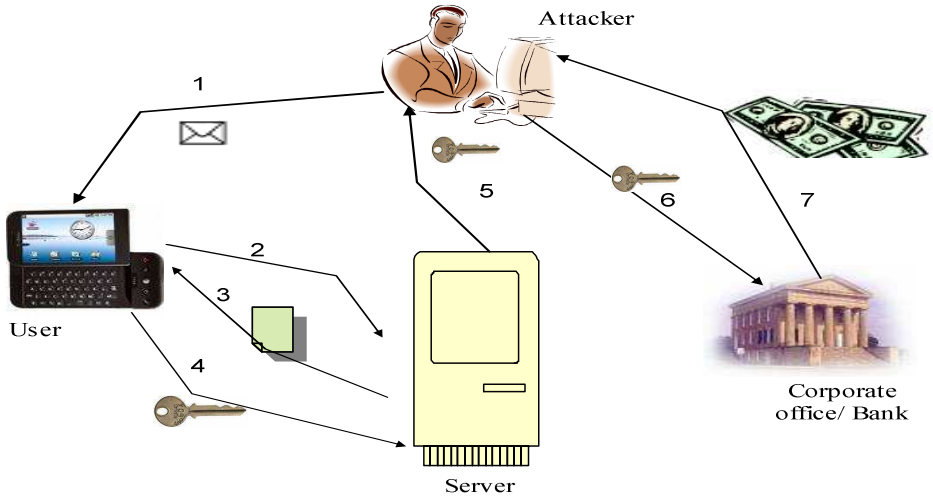

Figure 6: Phishing Attack

Step 1: Attacker sends malicious information to a phone user

Step 2: User makes him to be vulnerable to an information compromise

Step 3: User is prompted for valuable information through a malicious web site

Step 4: User information stored on phisher's server

Step 5: Phisher track username, password and other valuable information from phisher web server to his computer

Step 6: phisher use this for illegal use

Step 7: Phisher engages to receive money

According to Mills (2009), there are several kinds of attacks can be possible because of the open core vulnerabilities in Android. Android core libraries are open and any user can write their own applications that pose security threat from malware, Trojans hidden in animated images (PNG, .GIF, .BMP formats). Malware writers use the open source platform in Android platform and installs software to the core libraries and which may be invisible to others. This may be a virus and whenever the user installed into their phone, it gets compromised. The most advanced form of this attack is the Denial of Service attacks.

Distributed Denial of Service attacks (DDoS) in Android

Once Android phones get compromised from the viruses, the attackers can easily exploit this vulnerability by Denial of Service attacks. As long as the user is unaware of the risk, the phone is more dangerous for a most sinister form of attacks such as Distributed Denial of Service attacks and Botnets. For this, attacker uses special software to scan for vulnerability in the phone. Then they installed malicious code into the phone's memory and control the phone's operation remotely. The Android's open source code and search engines makes attacker's job easily to attack. There are several DDoS attacks that the attackers used in the digital environment in which Android phone is vulnerable for the following attacks.

DDoS(Malware and Trojan) – In this attack, attacker installs specially designed botnets into the user's phone for sending files across IRC channels, download Trojans and Malware to access the data from the phone's memory and record keystrokes from the user and track username, password, and other credentials for illegal purposes.

DDoS (Redirect DNS) – A more advanced form of phishing that an attacker modifies the contents of a web site on the host device. This will accomplish by Trojans and corrupts the DNS server by replacing the IP address.

There are several DNS vulnerabilities found in the Android phone after its release, that the hackers poison the DNS. From the Google publication (2008), noted that several companies experience that the attackers poisoned their DNS and sent people to the wrong pages. Also, some of the Android users experience that they try to visit yahoo home page and it redirected to an Antivirus download page for whatever reason. This is a trick used by the attackers to make a phone to get compromise from the attack.

Scenario

Of the DDoS attacks specified, DNS cache poisoning is very worst that looks like a legitimate site for the end user, actually its not. The attack scenario in this case involved in four stages.

Stage 1: User connects to www.ADBBank.com site -DNS Working

Step 1: Android user tries to connect www.ADBBank.com

Step 2: Caching server connects to the Authoritative server and return the IP Address of the legitimate ADBBank.com at 172. 16.251.11

Step 3: The address is cached and returned to the user and the user connects to ADBBank

Stage 2: Attacker looks for a door

Stage 1: Attacker uses special tools to expire the domain entry from the user on the caching server

Stage 2: Attacker inserts fake entry address in to the DNS Server

Stage 3: Attacker use the actual response from the user and match the fake address and send to the DNS Server

Stage 3: Attacker starts attack

Step 1: Attacker uses his own IP address instead of the legitimate server and controls the operation, that is www.ADBBank.com is at 172.16.251.15.

Step 2: DNS server responds to user queries with fake address

Stage 4: Attacker re-routed the connection

Attacker re-routed the connection to the server he controls; the phone user is unaware of the risk that he is sending the sensitive credentials to attacker's server.

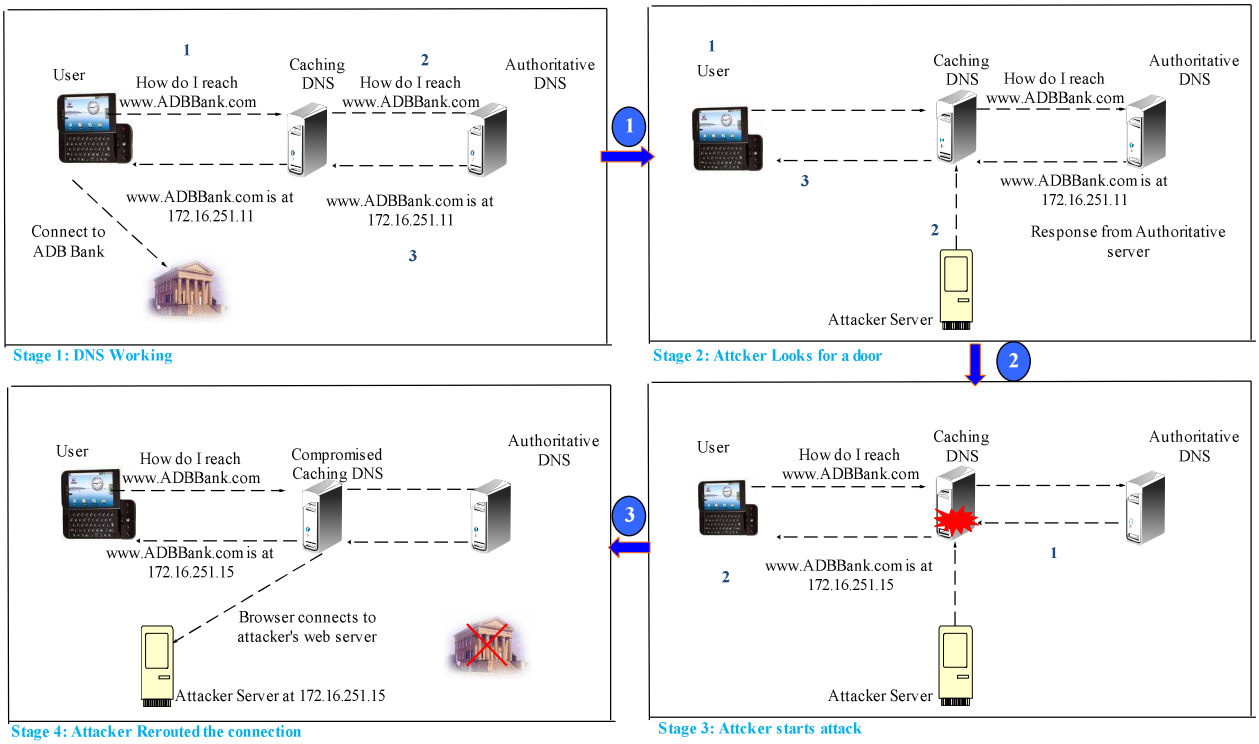

Figure 7: DNS Cache Poisoning

Google's Open Source Android phone is vulnerable in many ways as described above, even if Google will release a fix for the vulnerability, at that time, most of the users might have compromised and hacker got enough information to access their useful information. These vulnerability is not only limited to browser application but a malicious developer can insert a malware into any of its application because of the openness.

Security discussion

To enhance the security in Android phone, Google developers have to enforce the security in each layer. From the security analyses, we can realise that Android phone is secure in its well constructed Sandbox; however, it can be more secure, when the developer team implement security services in each layer of its framework. It can be accomplished by applying the following modification in their current system. This may reduce the multiple vulnerabilities and security issues so far (Corelab, 2008). A secure implementation should consider the VPN security enabled in the Apple iPhone (Rothman, 2008). Thus the following security mechanism will provide an additional security to Android with Sandbox approach.

- A VPN communication between phone and internet (authentication credentials such as username and password provided high level security)
- A secure Webkit browser
- Malware and Virus protection
- Encrypted email

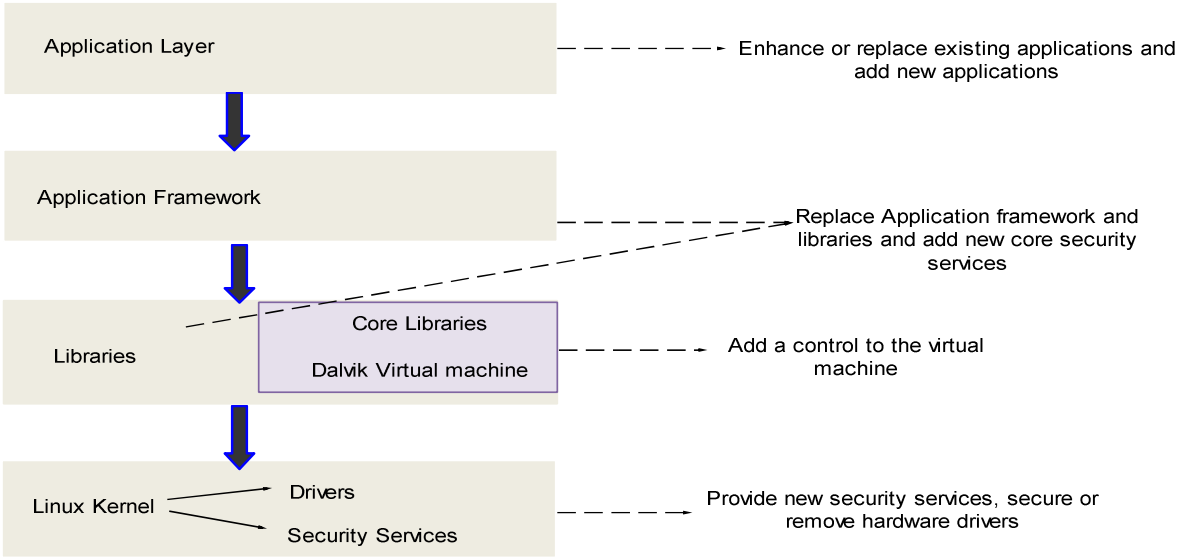

Figure 8: Android Security enhancement

But the debate will not end with its open user strategy and self signed security certificates for newly developed applications. The reality will hit it how much secure the Android Phone is?

CONCLUSION AND FUTURE WORK

The openness regarding the source code could be a problem concerning security in Android. From the security analysis in three phases, in some point of view, Android is less vulnerable than Apple iPhone, because once compromised, the attackers can easily capture all application on the iPhone easily. In contrast to this, Android uses sandbox approach so that when the phone gets compromised, attackers can access only the process involved in that particular application. However, Android lacks security in several ways; the permission based security in the security analysis Phase 2 is not so secure if two applications will share the User Identifier (UID). In addition, the phone user can accidentally gives permission for a malicious code to install into their mobile platform. The open nature and self signed certificates are the major vulnerability posed in the phase 3 Security analysis, any users can write the application and signed with the same key they have. There is no central control for certificates and this causes many risks in the mobile phone platform by means on an uncomplicated interface and permissions to update the applications on the phone.

The incompatibility issue with the available open source security software in the market are another reason that user cannot install this into the phone. This limits the security against malware entering into the phone. The current Android platform uses older version of open source programs and that are not up to date makes vulnerability in the core libraries. The unencrypted browser and source code vulnerabilities make the phone to a potential target for malware and Trojans. Furthermore, Android lacks security without a VPN implementation in their current platform, but Apple iPhone does. Thus, iPhone and Android is not secured than each other, only except the Android's well constructed Sandbox ensures more security than iPhone.

Android has an unpredictable future because of its openness nature of source code writing and vulnerabilities found in its core libraries so far. The only solution is that Android security team has to implement the architecture in some way to ensure the security and centralised control for writing and installing source code into mobile platform, Google has to take necessary precautions to harden security features for the current version, this not limited to certain features, also enhancing the existing applications and add new applications used in the first layer, harden the framework and add new security services, add a control for virtual machine, and provide more security services on the Linux kernel service layer. Using Security suites to protect the phone is only the next stage of security implementation.

REFERENCE

Android. (2009). Android Developer's Guide (Publication. Retrieved May 8, 2009: http://developer.android.com/guide/basics/what-is-android.html

Bournique, D. (2008). Google's Disruptive Android Strategy. Retrieved April 22, 2009, from http://wapreview.com/blog/?p=1105

Ciampa, M. (Ed.). (2007). *Security Awareness: Applying Practical Security in your world* (2 ed.). USA: Thomson Course Technology.

Corelab. (2008). Multiple vulnerabilities in Google's Android SDK Retrieved April 23, 2009, from http://www.coresecurity.com/content/advisory-google

DeLacey, B. (2007). Google Calling: Inside Android, the gPhone SDK. Retrieved April 22, 2009, from http://www.onlamp.com/pub/a/onlamp/2007/11/12/google-calling-inside-the-gphone-sdk.html

Dignan, L. (2008). Research firm: Google Android SDK has multiple vulnerabilities. Retrieved May 11, 2009, from http://blogs.zdnet.com/security/?p=921

Edge, J. (2009). Android application security. Retrieved May 8, 2009, from http://lwn.net/Articles/317067/

Ellison, C., & Schneier, B. (2000). Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure (Publication. Retrieved May 13, 2009: http://www.schneier.com/paper-pki.pdf

Enck, W., & McDaniel, P. (2008). Understanding Android's Security Framework (Publication. Retrieved May 8, 2009: http://siis.cse.psu.edu/slides/android-sec-tutorial.pdf

Gohring, N. (2008). Google Android Antivirus Available, and Unnecessary? Retrieved May 9, 2009, from http://www.csoonline.com/article/460867/Google_Android_Antivirus_Available_and_Unnecessary

Google. (2008). The T-Mobile G1 gets hijacked! Retrieved May 11, 2009, from http://androidcommunity.com/the-t-mobile-g1-gets-hijacked-20081124/

Greenberg, A. (2009). Privacy Groups Target Android, Mobile Marketers. Retrieved May 11, 2009, from http://www.forbes.com/2009/01/12/mobile-marketing-privacy-tech-security-cx_ag_0113mobilemarket.html?partner=relatedstoriesbox

Halley, B. (2008). How DNS cache poisoning works Retrieved May 11, 2009, from http://www.networkworld.com/news/tech/2008/102008-tech-update.html?tc=sec

Hoffman, S. (2008). Google's New Android Phone Vulnerable To Attack. Retrieved May 11, 2009, from http://www.outsourceit2philippines.com/news-outsource/Googles-New-Android-Phone-Vulnerable-To-Attack.htm

Insecure.org. (2006). Top 100 Network Security Tools. Retrieved May 16, 2009, from http://sectools.org/

Kameka, A. (2009). Android widget support poses security risk. Retrieved May 12, 2009, from http://androinica.com/2009/03/16/android-widget-support-poses-security-risk/

Lowe, S. (2008). T-Mobile G1 Android-Enabled Phone Announced. Retrieved April 23, 2009, from http://au.gear.ign.com/articles/912/912821p1.html

MARKOFF, J. (2008). Security Flaw Is Revealed in T-Mobile's Google Phone Retrieved April 12, 2009, from http://www.nytimes.com/2008/10/25/technology/internet/25phone.html?_r=1&ref=technology

Max. (2008). First Antivirus for Google Android G1 phones. Retrieved April 23, 2009, from http://www.bestsecuritytips.com/news+article.storyid+694.htm

Meyer, D. (2008). Researcher warns of Android browser vulnerability. Retrieved May 11, 2009, from http://www.zdnetasia.com/news/security/0,39044215,62047654,00.htm

Mills, E. (2009). Android phones await security patch. Retrieved May 8, 2009, from http://www.zdnetasia.com/news/security/0,39044215,62050966,00.htm

Mocana. (2008). NanoPhone Suite for Android (Publication. Retrieved May 8, 2009: http://mocana.com/pdfs/NanoPhone-Android.pdf

Moor, C. (2008). Android Security Team On Android Updates And Fixes (Publication. Retrieved May 9, 2009: http://www.talkandroid.com/398-android-update-info-rc29-rc30/

Mottl, J. (2008). Security Flaw Strikes G1 Android Phone. Retrieved April 12, 2009, from http://www.internetnews.com/dev-news/article.php/10792_3780801_1

Nelson, R. (2009). Google announces Secrets app for Android. Retrieved May 9, 2009, from http://androinica.com/2009/04/08/google-announces-secrets-app-for-android/

Perez, S. (2009). Android Vulnerability So Dangerous, Owners Warned Not to Use Phone's Web Browser: Updated. Retrieved May 11, 2009, from http://www.readwriteweb.com/archives/android_vulnerability_so_dangerous_shouldnt_use_web_browser.php

Poulsen, K. (2001). *New SubSeven Trojan unleashed. Journal*. Retrieved April 26, 2008, from http://www.theregister.co.uk/2001/03/13/new_sub_seven_trojan_unleashed/

Rothman, W. (2008). Why Android is Bad for Business. Retrieved May 8, 2009, from http://gizmodo.com/5053925/why-android-is-bad-for-business

Shankland, S. (2008). Google details 'reboot' bug, Android security fixes. Retrieved May 11, 2009, from http://www.builderau.com.au/news/soa/Google-details-reboot-bug-Android-security-fixes/0,339028227,339293211,00.htm

Speckmann, B. (2008). The Android mobile platform (Publication. Retrieved May 8, 2009: http://www.emich.edu/compsci/projects/Master_Thesis_-_Benjamin_Speckmann.pdf

Spectrum. (2008). Thoughts on Google Android (Publication. Retrieved May 9, 2009: http://www.spectrumdt.com/documents/SDTAndroidTechnicalWhitePaper_001.pdf

Strickland, J. (2008). How the Google Phone Works. Retrieved April 23, 2009, from http://electronics.howstuffworks.com/google-phone.htm/printable

Takahashi, D. (2008). Hacker finds a security hole in the Google Android software on the T-Mobile G1. Retrieved April 13, 2009, from http://venturebeat.com/2008/10/25/hacker-finds-a-security-hole-in-the-google-phone/

Trapani, G. (2009). Secrets For Android Secures Passwords And Sensitive Data. Retrieved May 9, 2009, from http://www.lifehacker.com.au/2009/04/secrets-for-android-secures-passwords-and-sensitive-data/

COPYRIGHT