CHALMERS |  UNIVERSITY OF GOTHENBURG

Penetration Testing of Android-based Smartphones

Master of Science Thesis in the Programme Networks and Distributed Systems

Naresh Kumar
Muhammad Ehtsham Ul Haq

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, June 2011.

Penetration Testing of Android-based Smartphones

Naresh.Kumar
Muhammad.Ehtsham Ul Haq

Examiner: Tomas. Olovsson

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2011

TO MY PARENTS AND FRIENDS
-Naresh Kumar

ACKNOWLEDGEMENT

This thesis work is carried out at Chalmers University of Technology Gothenburg, Sweden. We want to thank our examiner & supervisor Tomas Olovsson, an Associate Professor at Chalmers University of Technology, for all his kind support in the form of guidance, and suggestions to us.

We also want to thank Chalmers staff for providing us Smartphones to carry out this Technical work and our parents for their moral support during this work

ABSTRACT

The purpose of this work has been to perform a security analysis of Android-based Smartphones. Smartphone usage and adaptation are increasing day by day with a variety of applications. These applications can be very critical in nature such as mobile banking, and mobile payment systems and users are often unknowing about the security risks involved in such applications.

Android, an open source operating system, is rapidly increasing in the Smartphone industry. It has already beaten the most popular mobile operating systems, like RIM, iOS, Windows Mobile and even Symbian, which ruled the mobile market for more than a decade.

In this thesis, we have analysed the architecture of the Android operating system and tested its security through penetration testing. We have picked the most popular and recommended tools to test the security in the TCP/IP suite and different attacks have been performed on three different Android versions. The thesis also contains a discussion about our findings, how secure the Android system is and how much trust can be placed on it while using it.

Keywords: Android, Penetration testing, Smartphones.

TABLE OF CONTENTS

List of Abbreviations

ARP	Address Resolution Protocol
GPLv2	General Public Licensee Version 2
GPS	Global Positioning System
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IP	Internet Protocol
MAC	Media Access Control
NIST	National Institute of Standards and Technology
OS	Operating system
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Chapter 1

Introduction

Smartphone growth and adaptation is increasing rapidly due to their rich and versatile functionality. The versatility and convenience of these devices took them an ahead from other apparently similar devices like PDAs (Personal Digital Assistants) or MIDs (Mobile Internet Devices).

Nowadays, a Smartphone is not just used to talk; rather it gives functionality of a Pager, PDA (Personal Digital Assistants), MID, GPS, MP3 Player, etc., and provides a range of services like entertainment, electronic banking, reading e-books or attending office meetings online. Such a variety of services can only be delivered with the combination of strong compact hardware and fast reliable software including a good Operating System.

Currently, the Android is one of the most popular open source operating systems for Smartphones. It was originally developed by Google in 2005. Further development, the Android Open Source Project (AOSP) was established by Google and other members of Open Handset Alliance. Android is based on the Monolithic kernel (Modified Linux kernel) and contains all advanced features like multi-touch, video calling connectivity, multimedia messaging and web browsing. Several features and functions help to increase usage of data and services but also open the risk for introducing new vulnerabilities.

According to a survey that was released in February 2011 by a customer intelligence firm, Market Force, 33% of the individuals don't have a Smartphone, 34 % intend to purchase one having an Android operating system in the upcoming six months. From these potential customers, 21% said that they would buy an IPhone, 12% said that they would buy a Blackberry and 25% did not decided, what to buy [1]. This survey shows the increasing interest of potential customers in Android based Smartphones.

Android Smartphones rapid growth and adaptation makes it more attractive for hackers. To protect against attacks, as many system vulnerabilities as possible should be found and patched on a forehand. To detect the vulnerabilities of a system, penetration testing is a very important tool which helps finding security holes in the system. A penetration test, occasionally called pentest, is a method of evaluating the security of a computer system or network by simulating an attack from a malicious source, known as a Black Hat Hacker or Cracker [2]. The methodology for how to perform penetration tests is given by National Institute of Standards and Technology [3], see chapter 1.5. Along with penetration testing, a general overview about the Android security mechanism is described to give the reader an idea of how it works.

1.1. History

Mobile operating systems have been in use since the creation of the first mobile phone but those operating systems were targeted for specific devices. Most new devices used to come with improved operating systems but those improvements were often unnoticeable from the view of an ordinary user. With the advent of the Smartphone, the need for secure and robust multi-tasking mobile operating systems increased instantly.

The development of hand-held operating systems began for devices like Smartphones, personal digital assistants (PDAs) and mobile internet devices (MIDs). Although it is not well-defined in the industry what a Smartphone really is, IBM Simon is considered to be the first Smartphone [4] which had advanced features for its time where functionality such as Fax, Pager and PDAs as introduced. Modern Smartphones have much more advanced hardware and stronger operating system support. Nokia Corporation introduced the “Nokia Communicator Series”, which is the most prominent milestone towards existing Smartphones. GEOS were initially used by Nokia in its communicator series and later in 2001 the Symbian OS was deployed in the latest model of the time which was the Nokia 9210 Communicator. Since then, Nokia with its Symbian OS led the Smartphone industry for the next ten years.

Android, which was released in 2008, emerged as the major rival of Symbian OS and BlackBerry's RIM OS. According to Gartner statistics, in one year (2009 to 2010), Android based Smartphones increased their market share from 3% to 23% in the fourth quarter of 2010 [5]. As the Smartphones become more and more popular, new application development and quick releases of new functionality is becoming the key factor for the growth of Smartphones.

1.2. Problem Statement

The Android operating system is based on the Linux kernel which is an open source system [6] and provides a network stack for communication. To assess the security of the Android network stack, we have explored the Android operating system architecture, and created a penetration test methodology based on known attacks against TCP/IP and tools, which are needed to make successful penetration tests.

Most studies regarding security of smart phones have mainly focused on the application layer, such as viruses, worms, MMS exploitation and Cross-Service Attacks. Research work on mobile operating systems began in 2000 and 2001 [7][8], which shows the security of memory protection, permission-based file access control, etc. And because of Symbian’s leadership in over a decade of the Smartphone industry, most papers published have been focusing on the Symbian OS platform and describes the degree of robustness of the network stack [9][10][11]. Along with the penetration testing, we have analyzed the security mechanisms of the Android OS in this paper.

1.3. Objective

The main goals of this thesis are.

- Explore and describe the Android architecture.
- Explore the security mechanisms present in the Android OS.
- Identify and evaluate security problems in the Network stack of the Android OS with the help of penetration testing.

1.4. Scope

In this thesis, we have described the Android OS with respect to its core architecture and TCP/IP network stack security issues. To perform the security analysis, we used white box testing. First, the target system was identified and then penetration tests were performed, starting from well-known vulnerabilities to more specific deep penetration attacks.

Our focus was on WLAN network connectivity while performing the penetration tests. Like any other operating system, the Android OS has versions. Therefore, our target was not only one specific version, rather, we tested three versions, 1.6 (Donut) which is the oldest and used to be quite popular, 2.1 (Éclair) widely used today and 2.2 (Froyo) who's adaptation is rapidly increasing. We have analysed the improvements that were made in different versions by testing the three versions.

1.5. Penetration Testing Methodology

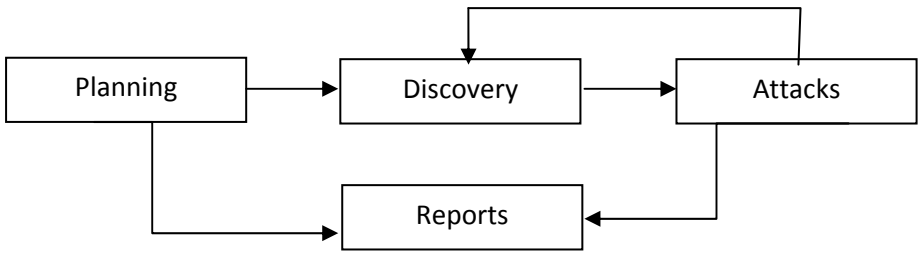

Figure 1: Penetration Testing Methodology by NIST[12]

In the NIST planning phase, test goals are set, policies are recognized, and management tasks are completed and result in reports. The discovery phase describes network port and service identification of the target system. At last, attacks are performed and reported. This flow works in a cycle to assess the security of the target system [12].

Our penetration testing methodology is performed in different stages. The first stage gives a description of the Android OS such as it is based on Linux kernel, etc. Second stage shows discovered closed and open ports, running or disable services and last stage describes the real vulnerability that exists in the system by performing some well-known attacks.

1.6. Thesis outline

Chapter 1

This chapter includes the introduction of the thesis work, objective, problem statement and Penetration methodology.

Chapter 2

This chapter describes details the Android OS, its versions history, general architecture and Android security mechanisms.

Chapter 3

This chapter describes port scanning, and its output, and summarizes our findings.

Chapter 4

This chapter describes different types of attacks we have performed against the Android Network stack, their result and conclusions.

Chapter 5

This Chapter contains an overall conclusion and results.

Chapter 2

Android Overview

Android is a software stack, which contains many things like an operating system, middleware and applications for users [13]. Android Inc. was developing it in Palo Alto, California in October 2003, and it was purchased by Google Inc. in August 2005 who started its further development. A consortium of several companies was formed in November 2007, the Open Handset Alliance, which released its first Android Smartphone in October. 2008 named as HTC G1.

From the start, the Linux kernel was used as a root. The Android system architecture is very complex due to the collection of ~185 changed sub-components written under ~19 different open source licenses [14]. It is not compatible with the X-windows system functionality which is present in the desktop version of Linux and also does not the use Linux libraries [15] but uses its own modified java virtual machine for application handling. It does not provide any kind of support to run conventional java applications.

2.1. Android Version History

In the Android development process many updates have been made, and versions have been released with new features and bug fixes. The version's history is given in the following table.

Version #	Name	Linux Kernel version	Release Date
1.5	Cupcake	2.6.27	30 April, 2009
1.6	Donut	2.6.29	15 September, 2009
2.0 / 2.1	Éclair	2.6.29	26 October, 2009
2.2	(Froyo)	2.6.32	20 May, 2010
2.3	Gingerbread	2.6.35	6 December, 2010
3.0	Honeycomb	2.6.36	January, 2011

Table 1: Android versions

2.2. Android Architecture

The Android architecture can be divided into four sections that are described below.

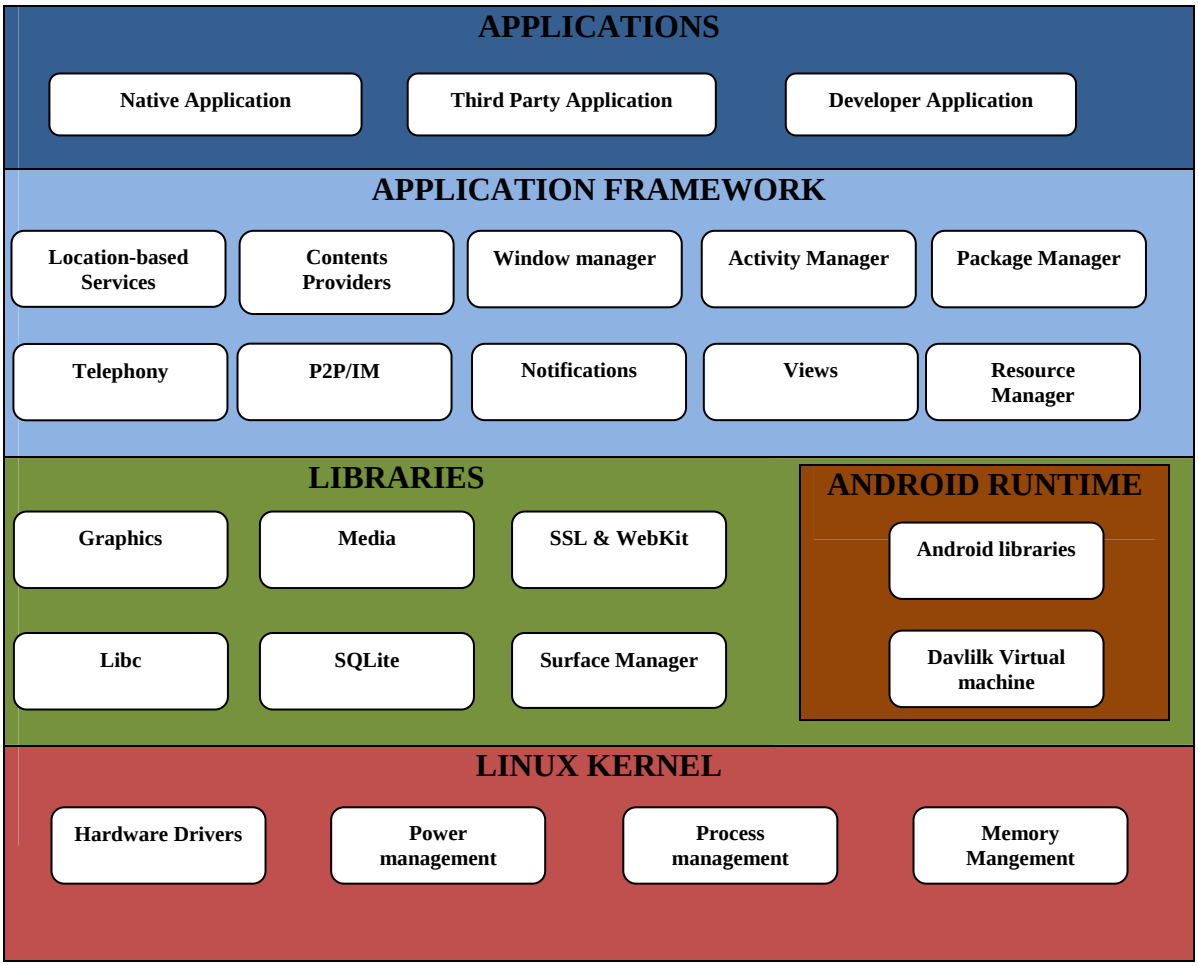

Figure 2: The Android architecture [16]

2.2.1. Application layer

The top section contains set of built-in applications such as Calendar, Contacts, e-mail, and a Web browser and also contains custom application. These applications are developed in the Java and also have the ability to run in a multi-process environment such as listening to music while reading an email. The users are able to communicate with these applications directly without having the knowledge of how operating system works. Some of the applications are already installed in the Smartphone while others can be freely downloaded from the Internet.

2.2.2. Application framework

The application framework is a component-based framework where an application developer can build new, rich and modern applications. This architecture is designed in such a way that it provides reusable components, for example, scrollbar elements, which are used by many components. The Application framework is completely written in the Java language and

consists of a huge set of the classes, interfaces and packages. There are four main types of Android application components and the details of these components are given below.

Activities

Activities are tasks that the user performs through the graphical user interface. In an application, an activity can be a single activity for example to display a list of contacts from the contact application, or it can be a sequence of activities (functions and windows) such as when sending a message, a user has to get the number from the contact list and then click on the send button.

Services

Services are also called Components, and do not contain any user interface or GUI thread. Service processes run in the background. Multiple services execute and perform different tasks at the same time, for example, music listening while slow fetching of data over the network means long-running tasks performed without any user interaction [17]. Services are declared in the xml file "AndroidManifest.xml".

Broadcast Receivers

A Broadcast Receiver is used to receive and respond to broadcast announcements. Most of the broadcasts are initiated by the system such as by giving notification that the battery power is very low. A Broadcast receiver can have any user interface but makes use of a notification manager or starts an activity to display alert messages when a broadcast takes place.

Content provider

The Content provider is used to provide application's data to different components. The data can be accessed by different applications and stored on difference places such as a file system or on the web. Applications are able to modify the data with the help of a content resolver interface which is a class in Android system.

2.2.3. Libraries

The Android system provides some C/C++ libraries. Different components of the Android system can utilize these libraries. All these libraries are accessible with the help of the application framework [18][19].
Some of the important libraries of Android system are given below.

- **System C Library:** This library, also called bionic library is the main library in the Android system, and it is a BSD-derived implementation. It is useful for Application developers to create applications.
- **Media Libraries:** This library depends on PacketVideo's OpenCORE, which provides multimedia features for device development. It provides playback and streaming in different standard formats of audio, video and images such as MPEG, PNG, and AMR, etc.
- **Surface Manager:** It handles 2-D/3-D windows for different applications to provide the graphics facility and also provide OpenGL library.
- **Libwebcore:** This library supports two types of functionality: Web browser and Embeddable web view.

2.2.4. Android runtime

The Android run-time environment consists of two things: first, a set of core libraries that gives functionality, which is available in the Java programming language, and the second is Dalvik, which is Google's own virtual machine specially designed for less power usage and space and used by the applications in the Android system. Each application runs within its own instance of the VM, thus the application processes are completely isolated from each other**.**

Dalvik uses the low level functionality from the Linux kernel such as threading and memory management. Before execution of applications, Java files are converted into Dalvik Executable (.dex) format for minimal memory footprints. This byte code format is not similar to the standard byte code but is more compact and uses less space. It is important since the device is small and contains less memory than conventional machines. When building an application, a standard development process is followed: source code (.java) is compiled into Java byte code after that a dx tool is used, which converts this into .dex file format that is capable of running in the Android device.

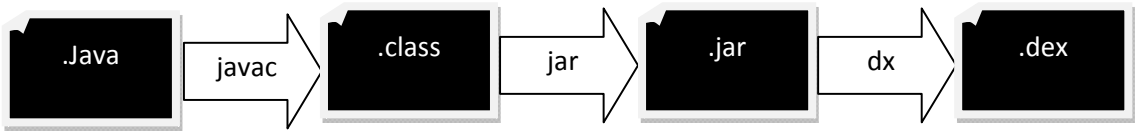

Figure 2.1: .java to .dx Conversion[18]

2.2.5. The Linux Kernel

Android is built on the GPLv2 licensed 2.6 [20] Linux kernels with approximately 115 patches, which provide basic functionality such as memory management, network stacks for communication, etc. The reason to build the Android on the Linux kernel is that it is stable, provides support for shared libraries, has a variety of device drivers, hardware features and contains process management.

2.3. Security Mechanisms of Android

The Linux kernel has features like user identifiers, pre-emptive multi-tasking, etc., that is used to enforce security between applications and system files. Unlike in desktop Linux systems where all application belonging to a user executes with same user id, in the Android system each application is assigned a unique identifier and separate instance of its own virtual machine so the application code runs in its own memory and process. For sharing data between application a "sharedUserId" feature used in the Android system.

Android systems have a permissions based mechanism to enforce security restrictions on applications. At installation time, a user has the chance to set flags to allow different permissions on applications. By default, a normal level of protection is granted to every

application in a manifest file. Initially, the application is not able to access resources such as using the GPS, contact lists, writing to another application, accessing network services, etc. The installer package shows the lists of permissions where a user can allow or deny these permissions. When a user sets the permission then it can easily access the resource and it is not possible to revoke these permissions until the application is uninstalled.

Android security mechanisms also use the concept of sand-boxing to implement secure multiprocessing of applications. A reference monitor is used for inter-component communication (ICC) between applications. The application configuration is also written to the manifest file.

Digital certificates are used in application development by developers to sign their code. For this, a private key is used which creates trust between applications. Signing of applications does not deal with a certificate authority, and self-signed certificates are accepted identifying the author of the application. If a developer tries to install an unsigned application, then the system will not allow this.

The application also has public and private application components. Private application components are only accessible to other components, but not from other applications. Private component permission is set in the Manifest file by the developer. Public components are accessible to other applications but a developer has the choice to assign permissions to these components.

Chapter 3

Port Scanning

The second step of our penetration test was to perform port scanning in order to detect active devices, open ports and running services. Port scanning is a technique through which a network administrator or even an attacker can determine what types of services are running and are publicly accessible, such as an FTP server or a mail server. Applications can contain vulnerabilities if they offer services to the network, and are therefore possible targets for attacks. The TCP/IP stack can also contain vulnerabilities, regardless of whether any applications are running or not, ports can be categorised into three ranges [23] :

- The Well Known Ports (0-1023)
- The Registered Ports (1024-49151)
- Dynamic and/or Private Ports (49152-65535)

Well known ports are used for privileged services such as Web servers using port 80 (assigned by IANA) and many system services use these ports. It is often enough to scan these ports to find an interesting service, against which an attack can be performed. Registered ports are used by normal programs. Most well-known applications have reserved ports in this range.

Dynamic ports, also called as private ports use the range 49,152 to 65,535. Third party applications and programs are able to use these ports without risking a collision with any well-known applications.

Port scanning can be done though a number of different techniques that are described in the white paper of “Port Scanning Techniques and the Defense Against Them” [24].

We have selected the Nmap tool for port scanning as it is a popular tool. Before describing these techniques and results, it’s important to know the mechanism of how a TCP connection is established and terminated, since lots of scans violate this process. In the next chapter, we explain how a half-open connection can be used to exploit a target system's resources and how an established connection is terminated abnormally.

3.1. Three way handshake

To establish a reliable communication link using TCP/IP between two machines (client and server), a handshake process must be completed before normal communication begins. The handshake process begins with the client sending a packet with the SYN flag set to the server. In the reply, the server responds with a SYN/ACK packet. The port must be open to complete the handshake process i.e. some service should be running on the particular port. Finally, the client sends an ACK packet back to the server to complete the three way handshake process. The Handshake process is shown in the following figure.

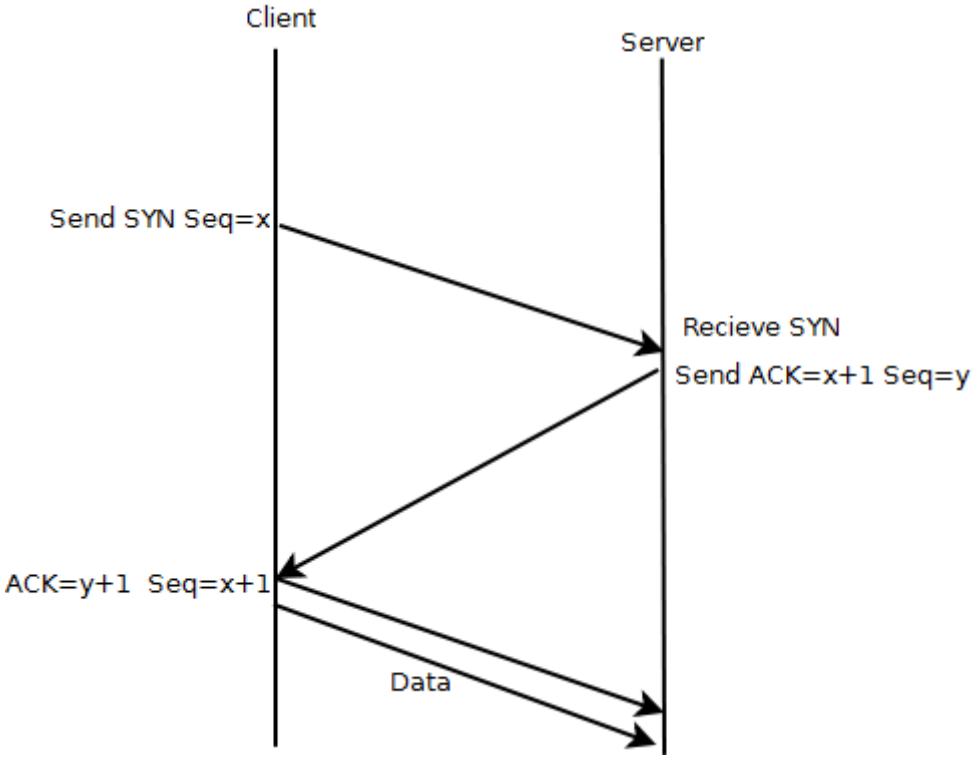

Figure 3: TCP Connection Establishment [25].

After this connection establishment, both the client and the server are able to transfer the application layer data. To check a particular port's status that whether it is open or not, it is not necessary to complete the handshake process. Only sending the SYN packets and analyzing the response is enough to know the port's status.

3.2. TCP Connection Termination

It is done in the four steps [26].

- When a client wants to be disconnected it sends a segment which contains FIN message with a last received sequence number.
- On receiving this segment, server sends an ACK to the client to acknowledging its termination request.
- When the server is finished with its transmission, it creates its own FIN packet and then sends it.
- The client acknowledges this last FIN packet and then connection is closed.

3.3. Test Scenario

In the test scenario, our Smartphones have two services running, a Web Server and an FTP Server named http-proxy and ccproxy-ftp on ports 8080 and 2121 respectively. These services were installed by us.

Figure 3.1: Lab Test Scenario

We used a D-Link (DWL-G700AP) Access Point as our central wireless service provider for communication. Two computer systems were also used: one to generate different kinds of attacks and another to analyze the traffic that was generated during an attack. We used Ubuntu 10.04 on both machines. A traffic analyzer (Wireshark 1.2.7) was installed on both machines to analyze the information. There are many freely available command line tools to check and exploit vulnerabilities, and we selected some widely used and comprehensive tools: Nmap for port scanning, Hping3, Dsniff and packit (a Linux command line tool) for attacks. These tools were installed on one of the Ubuntu 10.04 system to generate the attacks.

3.4. Port Scanning Techniques

3.4.1. Ping Sweep scan

Ping Sweep scans are actually not port scans but can give information about whether the target device is active or not. It also tells if the target system is protected by a firewall. This scan is accomplished by two frames (ICMP echo request and ICMP echo reply). Scan command, output and results are given in following table.

Command	#nmap -sP 192.168.0.101-103
Output for Android 1.6	Nmap scan report for 192.168.0.101 Host is up (0.13s latency). MAC Address: 00:18:41:E8:6F:79 (High Tech Computer) Nmap done: 1 IP address (1 host up) scanned in 0.51 seconds
Output for Android 2.1	Nmap scan report for 192.168.0.102 Host is up (0.10s latency).

	MAC Address: 10:1D:C0:CF:7C:A0 (Unknown) Nmap done: 1 IP address (1 host up) scanned in 0.53 seconds
Output for Android 2.2	Nmap scan report for 192.168.0.103 Host is up (0.12s latency). MAC Address: 38:E7:D8:F8:BD:28 (HTC) Nmap done: 1 IP addresses (1 hosts up) scanned in 0.52 seconds
Tools	Nmap, it uses –sP option to detect active host.
Results	All the three hosts are alive and their MAC Addresses are found.

3.4.2. Operating System fingerprinting

OS fingerprinting means to find which operating system is running on the target system. Nmap does many tests like SYN scan, IP ID sampling and initial TCP sequence number, and window size analysis, and finally compares the result with its own database of systems (nmap-os-db) and tries to make a guess about the target operating system [27]. Scan command, output and results are given in following table.

Command	#nmap -O 192.168.0.101-103
Output for Android 1.6	Nmap scan report for 192.168.0.101 Host is up (0.053s latency). Not shown: 997 closed ports PORT STATE SERVICE 2121/tcp open ccproxy-ftp 8080/tcp open http-proxy MAC Address: 00:18:41:E8:6F:79 (High Tech Computer) Device type: general purpose Running: Linux 2.6.X OS details: Linux 2.6.9 - 2.6.31 Network Distance: 1 hop
Output for Android 2.1	Nmap scan report for 192.168.0.102 Host is up (0.0035s latency). Not shown: 999 closed ports PORT STATE SERVICE 2121/tcp open ccproxy-ftp 8080/tcp open http-proxy MAC Address: 10:1D:C0:CF:7C:A0 (Unknown) Device type: general purpose Running: Linux 2.6.X OS details: Linux 2.6.9 - 2.6.31 Network Distance: 1 hop
Output for Android 2.2	Nmap scan report for 192.168.0.103 Host is up (0.0066s latency). Not shown: 997 closed ports PORT STATE SERVICE 2121/tcp open ccproxy-ftp 8080/tcp open http-proxy MAC Address: 38:E7:D8:F8:BD:28 (HTC) Device type: phone

	Running: Linux 2.6.X OS details: Android 2.2 (Linux 2.6.32.15) Network Distance: 1 hop
Tool	Nmap ,it uses –O option for OS fingerprinting.
Results	In OS fingerprinting, For Android 1.6 and 2.1, Nmap shows the device type as general purpose running Linux 2.6.X. For Android 2.2, the exact details i.e. OS, version, device type, are shown. It does not mean that 1.6 have any mechanism to hide its operating system, instead there is no exact match in Nmap database. We tried this scan with Nmap 5.0 also, and we got the same results for 1.6, but it does not recognise Android 2.2, rather it shows that the device is a Mac system. We, therefore conclude that in Nmap 5.0, there is no match for Android 2.2 and in Nmap 5.5, there no exact match for Android 1.6 and 2.1.

3.4.3. The TCP connect scan

In the connect scan, the tool performs a complete three-way handshake and then immediately a RST packet will be sent to close the connection. If the port is closed, the Smartphones sends a RST as a response to the SYN packet. Scan command, output and results are given in the following table.

Command	#nmap -sT 192.168.0.101-103
Output of Android 1.6	Nmap scan report for 192.168.0.101 Host is up (0.042s latency). Not shown: 998 closed ports PORT STATE SERVICE 2121/tcp open ccproxy-ftp 8080/tcp open http-proxy MAC Address: 00:18:41:E8:6F:79 (High Tech Computer)
Output of Android 2.1	Nmap scan report for 192.168.0.102 Host is up (0.027s latency). Not shown: 999 closed ports PORT STATE SERVICE 2121/tcp open ccproxy-ftp 8080/tcp open http-proxy MAC Address: 10:1D:C0:CF:7C:A0 (Unknown)
Output of Android 2.2	Nmap scan report for 192.168.0.103 Host is up (0.027s latency). Not shown: 998 closed ports PORT STATE SERVICE 2121/tcp open ccproxy-ftp 8080/tcp open http-proxy MAC Address: 38:E7:D8:F8:BD:28 (HTC)
Tools	Nmap, it uses –sT option for connect scan.
Results	The results of this scan are similar to most operating systems. This

	scan shows all open ports with running services, in this case the ports opened by the applications we installed. It also explicitly mentions the number of closed ports.

3.4.4. SYN Scan

A SYN Scan is done by sending a single packet with a SYN flag set to the target but the three-way handshake is never completed. It is also called “half-open” scan. A SYN-ACK reply shows an open port and a RST reply shows closed ports.

The main advantage of this technique is that if an IDS (intrusion detection system) is running, which can detect a complete connection being initiated from outside, it might not report a SYN scan because RST is sent right after receiving the response from the server. In addition, there is normally no connection entry created in the target operating system’s using SYN scan since the three-way handshake is never completed and no connection was established. Scan command, output and results are given in the following table.

Command	#nmap -sS 192.168.0.101-103
Output of Android 1.6	Nmap scan report for 192.168.0.101 Host is up (0.042s latency). Not shown: 998 closed ports PORT STATE SERVICE 2121/tcp open ccproxy-ftp 8080/tcp open http-proxy MAC Address: 00:18:41:E8:6F:79 (High Tech Computer)
Output of Android 2.1	Nmap scan report for 192.168.0.102 Host is up (0.0016s latency). Not shown: 999 closed ports PORT STATE SERVICE 2121/tcp open ccproxy-ftp 8080/tcp open http-proxy MAC Address: 10:1D:C0:CF:7C:A0 (Unknown)
Output of Android 2.2	Nmap scan report for 192.168.0.103 Host is up (0.027s latency). Not shown: 998 closed ports PORT STATE SERVICE 2121/tcp open ccproxy-ftp 8080/tcp open http-proxy MAC Address: 38:E7:D8:F8:BD:28 (HTC)
Tools	Nmap, It uses –sS option for SYN scan.
Results	SYN scan results are similar to the connect scan because no IDS is running on target Android smart-phones. It shows open ports with their names and number of close ports.

3.4.5. FIN Scan

In the FIN scan, an attacker sends a packet with the FIN flag set. If the port is closed, a RST will be received. However, the system will ignore the packet if the port is open. FIN scan is

often used to check if a stateless firewall is protecting the system. Scan command, output and results are given in the following table.

Command	#nmap -sF 192.168.0.101-103
Output of Android 1.6	Nmap scan report for 192.168.0.101 Host is up (0.022s latency). Not shown: 998 closed ports PORT STATE SERVICE 2121/tcp open|filtered ccproxy-ftp 8080/tcp open|filtered http-proxy MAC Address: 00:18:41:E8:6F:79 (High Tech Computer)
Output of Android 2.1	Nmap scan report for 192.168.0.102 Host is up (0.0069s latency). Not shown: 999 closed ports PORT STATE SERVICE 2121/tcp open|filtered ccproxy-ftp 8080/tcp open|filtered http-proxy MAC Address: 10:1D:C0:CF:7C:A0 (Unknown)
Output of Android 2.2	Nmap scan report for 192.168.0.103 Host is up (0.019s latency). Not shown: 998 closed ports PORT STATE SERVICE 2121/tcp open|filtered ccproxy-ftp 8080/tcp open|filtered http-proxy MAC Address: 38:E7:D8:F8:BD:28 (HTC)
Tools	Nmap , it uses –sF option for Fin Scan.
Results	It tells about closed port on target machines. It shows Open|filtered for open ports or ports behind a firewall. There is no built-in firewall in Android operating system and therefore Open|filtered means an open port on the target.

3.4.6. XMAS Scan

This scan is done by sending a packet in which the FIN, PSH and URG flags are set. A closed port will always send RST but if the port is open, the packet will be dropped silently. Scan command, output and results are given in the following table.

Command	#nmap -sX 192.168.0.101-103
Output for Android 1.6	Nmap scan report for 192.168.0.101 Host is up (0.021s latency). Not shown: 998 closed ports PORT STATE SERVICE 2121/tcp open|filtered ccproxy-ftp 8080/tcp open|filtered http-proxy MAC Address: 00:18:41:E8:6F:79 (High Tech Computer)
Output for Android 2.1	Nmap scan report for 192.168.0.102 Host is up (0.011s latency). Not shown: 999 closed ports

	PORT STATE SERVICE 2121/tcp open|filtered ccproxy-ftp 8080/tcp open|filtered http-proxy MAC Address: 10:1D:C0:CF:7C:A0 (Unknown)
Output for Android 2.2	Nmap scan report for 192.168.0.103 Host is up (0.0069s latency). Not shown: 997 closed ports PORT STATE SERVICE 2121/tcp open|filtered ccproxy-ftp 8080/tcp open|filtered http-proxy MAC Address: 38:E7:D8:F8:BD:28 (HTC)
Tools	Nmap, it uses –xS option for xmas scan
Results	Nmap shows Open|filtered ports on machines, when we analyzed the traffic using wireshark, the packet sent to these ports were dropped which means that the ports are open.

3.4.7. NULL Scan

The Null scan is the used to determine if a port is open or not. To do this, a TCP packet is sent with no flags (SYN, RST or ACK is required by the standard). If the reply is RST, then the corresponding port is closed, otherwise it is open. Scan command, output and results given in the following table.

Command	#nmap -sN 192.168.0.101-103
Output for Android 1.6	Nmap scan report for 192.168.0.101 Host is up (0.045s latency). All 1000 scanned ports on 192.168.0.101 are open|filtered MAC Address: 00:18:41:E8:6F:79 (High Tech Computer)
Output for Android 2.1	Nmap scan report for 192.168.0.102 Host is up (0.076s latency). All 1000 scanned ports on 192.168.0.102 are open|filtered MAC Address: 10:1D:C0:CF:7C:A0 (Unknown)
Output for Android 2.2	Nmap scan report for 192.168.0.103 Host is up (0.76s latency). All 1000 scanned ports on 192.168.0.103 are open|filtered MAC Address: 38:E7:D8:F8:BD:28 (HTC)
Tools	Nmap, it uses –sN option for Null scan.
Results	When we sent TCP packets with no flags set to Smartphones, the Nmap tool gave a result that all ports are open|filtered but there are so many ports in the closed state as we already know from the other scans, so we analyzed the traffic using wireshark. We were then able to see that Android sends response packets with the [RST, ACK] flags set for closed ports and drops packets sent to open ports. We can say that this scan was unsuccessful or does not give any useful information while using Nmap tool but when we analyse the traffic using Wireshark tool, we can conclude which ports are open and which ones are closed.

3.4.8. ACK Scan

ACK scanning is the most useful technique to determine port status, but it can only determine whether the ports are filtered or not. The open and close status will be determined by other scans. Scan command, output and results are given in the following table.

Command	# nmap -sA 192.168.0.101-103
Output for Android 1.6	Nmap scan report for 192.168.0.101 Host is up (0.085s latency). All 1000 scanned ports on 192.168.0.101 are unfiltered MAC Address: 00:18:41:E8:6F:79 (High Tech Computer)
Output for Android 2.1	Nmap scan report for 192.168.0.102 Host is up (0.0021s latency). All 1000 scanned ports on 192.168.0.102 are unfiltered MAC Address: 10:1D:C0:CF:7C:A0 (Unknown)
Output for Android 2.2	Nmap scan report for 192.168.0.1032 Host is up (0.0021s latency). All 1000 scanned ports on 192.168.0.103 are unfiltered MAC Address: 38:E7:D8:F8:BD:28 (HTC)
Tools	Nmap, it uses –sA option for Ack scan.
Results	This scan was successful, its shows unfiltered port which means there is no firewall installed in the OS.

3.4.9. UDP Scan

UDP port scanning is used for checking for open and closed UDP ports. As UDP does not use a three way handshake, only ICMP messages are returned in response. If the ICMP message type is 3, the corresponding port is closed and any other reply indicates an open port. Scan command, output and results are given in the following table.

Command	nmap -sU -F 192.168.0.101-103
Output for Android 1.6	Nmap scan report for 192.168.0.101 Host is up (0.098s latency). All scanned ports on 192.168.0.101 are closed MAC Address: 00:18:41:E8:6F:79 (High Tech Computer)
	Nmap scan report for 192.168.0.102 Host is up (0.011s latency). Not shown: 999 closed ports PORT STATE SERVICE All scanned ports on 192.168.0.102 are closed MAC Address: 10:1D:C0:CF:7C:A0 (Unknown)
Output for Android 2.2	Nmap scan report for 192.168.0.103 Host is up (0.19s latency). All scanned ports on 192.168.0.102 are closed MAC Address: 38:E7:D8:F8:BD:28 (HTC)
Tools	Nmap, its uses –sU option for UDP scan.
Results	All the three target devices show that all scanned UDP ports are

	closed.

3.5. Tools Used

Nmap 5.5: Nmap is a popular and widely useful tool for port scanning. It includes a network scanning guide and it is a free and open source utility for network discovery. It has support to run on a variety of Operating Systems like UNIX and Windows. It contains features like host discovery, port scan and OS detection.

Wireshark (1.2.7): We analysed the traffic using Wireshark (1.2.7), which is a free network protocol analyser. Packets were captured from scans and analysed to deeply understand the data being exchanged. Wireshark was running on one ubuntu machine to capture and analyze the wireless traffic during our tests.

3.6. Summary

Port scanning was used to discover information about the target systems. Information that was gathered in this step will be useful to further enhance the penetrating tests towards specific attacks.

It is important to mention that the Android system was not recognised and matched by Nmap 5.0 in its database of systems rather it gives a range of Linux kernel versions for 1.6 and 2.1 but no information about 2.1. When we tested with Nmap 5.5, we still got the same results for 1.6 and 2.1 but for 2.2, we got the exact match of operating system with even exact match of the Linux kernel that is 2.6.32.15.

Most of the results of this port scanning are similar to that of Linux with the corresponding kernels i.e. Android 1.6 and 2.1 with Linux kernel 2.6.29 and Android 2.2 with Linux kernel 2.6.32.

The typical risk related to these scan results is low according to CAPEC-300 documentation [28]. As we have followed CAPEC-300 documentation for complete port scanning, we also tried the some other scans but did not find any useful information.

Chapter 4

Attacks on the Android Network Stack

TCP/IP implementations have in the past had serious security flaws like being susceptible to buffer overflow attacks and flaws like TCP sequence number prediction, etc. These security flaws have been identified and patched over time. The current TCP/IP suite is not the result of a straight-forward planned process instead it is the result of an evolution that is still continuing today. Most of the time this process has been a post active solution i.e. whenever vulnerability was found, action was taken to overcome that particular vulnerability. Pro-active steps could be more useful to find vulnerabilities that arise due to new functionality and development of new applications.

To begin with, we first looked at Smartphone vulnerabilities from the past, for example Symbian systems that were vulnerable to spread Malware by sending SMS messages, iPhone Smartphones had problems like “Fixed SMS vulnerably'[29]” though which a hacker could gain full control, and the Brador virus, which was found in Windows CE operating system of packet PC devices, which lead to a complete operating system restart[30] and Windows Mobile Operating System which is vulnerable to ARP spoofing and SYN flooding attacks [31]. These different types of vulnerabilities show typical risks in Smartphone systems, and therefore security testing is an important tool to minimize such threats in the Smartphones.

Android is based on several components for communication, and its network stack is taken from the Linux kernel which was compressed in size and with a lot of changes. It consists of a large number of network protocols for communication like TCP, UDP, and BNEP (Bluetooth Network Encapsulation Protocol), etc.

The communication between mobile devices can be interfered with in different ways, some of them are given below and our attacks also fall in to these categories.

Eavesdropping: In this attack the traffic is sniffed through different network sniffing tools. After an interception of a packet, an attacker is able to extract information, e.g. the information that is exchanged during the three way handshake process, sequence numbers, ACK number, destination port etc. This information can be used to interfere with the communication between the target devices.

The Man-in-the-Middle Attack: An attacker can modify the contents of packets and take over a session between two communicating devices by intercepting the communication. Most often, a network packet sniffer is used to perform this attack. The man-in-the middle can be an internal user or someone from an outside network. Along with session hijacking, the goals for a man-in-the middle can be to eavesdrop information, cause denial of services, etc.

Denial of Service Attack: This attack aim to stop the authorized user to access any services or information. Most denial of service attacks are caused by traffic flooding over the Internet.

The table below describes the attacks with respect to OSI layers.

Transport layer	SYN flooding, ACK flooding, Null flooding and UDP flooding.
Network Layer	ICMP flooding, Ping of Death, Smurf, and Land Attack.
Data link layer	Gratuitous ARP Request, Fake Arp Reply using Broadcast.

Table 4: Attacks with respect to layers

4.1. SYN Flooding

SYN Flooding is an old attack performed by violating the TCP/IP handshake mechanism. In this attack, the attacker sends multiple SYN messages with spoofed IP addresses to the victim server's open ports. The server then responds with a SYN-ACK packet. The attacker continues with more SYN packets instead of sending an ACK packet. Consequently, a lot of half-open TCP sessions are established. However, a server has a table of fixed size for half-open (pending) connections. And once it is full, no new connections can be accepted, therefore this lead to denial of service for legal users. Attack command and results are given in the following table.

Command	**sudo hping3 -S 192.168.0.101 -p 8080 --flood** **sudo hping3 -S 192.168.0.101 -p 2121 –flood**
Result for Android 1.6	When we performed SYN flooding on the open ports, it did not respond to any hosts on the network until the flooding of SYN packet was stopped. It was also observed that the Smartphone became relatively slow in response to applications when compared to the response before the attack.
Result for Android 2.1	SYN flooding attack is successful on 2.1, and networked applications were not able to access open services. When we checked system respond times, we observed no difference or delays in response of applications.
Result for Android 2.2	Response to SYN flooding is same as for other versions but the device is almost irresponsive!! .

4.2. ACK Flooding

This attack is performed by sending a large number of TCP packets with the ACK flag set, so the targeted operating system will consume all CPU time to check earlier related SYN messages against these ACKs. Attack command and results are given in the following table.

Command	**sudo hping3 -A 192.168.0.104 -p 8080 –flood** **sudo hping3 -A 192.168.0.104 -p 2121 –flood**
Result for Android 1.6	When we performed ACK flooding on the open ports, it did not respond to any hosts on the network until the flooding of ACK packet was stopped. The Smartphone also comes to a halt after 5 to 10 seconds if this attack continues and it gives response as soon as we stop flooding with ACK packets.
Result for 2.1	ACK flooding saturates the network interface of 2.1 but there is no difference in regular usage of application that doesn’t require network communication.

Result for 2.2	Target Smartphone did not respond to any network packet that was destined for its address. In addition the Smartphone becomes completely unresponsive in any functionality.

4.3. TCP Null Flooding

Null flooding is used to increase server delay for the services which are running on the server. By sending large amount of TCP packets with no flag set to mobile, the attacker can successfully produce a denial of service attack. Attack command and results are given in the following table.

Command	**sudo hping3 -Y 192.168.0.101 -a 192.168.0.103 –flood**
Result for Android 1.6	During this attack, system performance became extremely slow and after some time mobile was stuck with this message “Force close or Wait “ of application that is currently running. When we click on “Force close” or “Wait” then after some seconds this message appears again. Due to flooding, the mobile was unable to provide service to any host.
Result for Android 2.1	When we sent a large amount of TCP packets with no flag to the Android 2.1, it did not respond to any hosts on the network. There is no difference in performance of normal application usage.
Result for Android 2.2	This attack is successful on 2.2 also and the system response time is significantly changed during this attack.

4.4. RST Attack

RST is used to abort a TCP connection. By sending packet with RST set and a correct sequence number with the spoofed address, an attacker can abort the active connection. Attack command and results are given in the following table.

Command	**sudo packit -s 192.168.0.99 -d 192.168.0.102 -S 56893 -D 2121 -F R -q 2676420498 -c 1 -b 20 -e 00:09:5b:e3:6d:7e**
Result for Android 1.6	We analyzed the traffic through Wireshark and noticed the sequence number with the command mentioned above; we generated an RST packet with a correct sequence number and successfully aborted the established connection.
Result for Android 2.1	Result is same on this version too.
Result for Android 2.2	Result is same on this version too.

4.5. UDP Flooding

In this attack, a flood of UDP packets is sent an open port of server. The mobile server may be unable to respond to the client. Attack command and results are given in the following table.

Command	sudo hping3 -2 192.168.0.101 –flood
Result for Android 1.6	This attack is successful, when we sent the ping packet to Smartphones , we got very slow response. Smartphone's performance is slightly decreased when we uses the applications
Result for Android 2.1	This attack is successful, same like 1.6.
Result for Android 2.2	Mobile give very slow response to ping request and its performance significantly disturbed when we used the applications of the Smartphones.

4.6. ICMP Flood

This attack is performed by sending large amount of ping packets (ICMP echo) to the target system. Target System may become busy and not respond to legal users. Attack command and results are given in the following table.

Command	sudo hping3 192.168.0.101 -1 –flood
Result for Android 1.6	Target did not reply to any packets and system performance became very slow.
Result for Android 2.1	Target system did not reply to any packets and performance of the system was normal.
Result for Android 2.2	Like other previously described attacks, 2.2 have the same behaviour as 1.6 and its response to ICMP flooding reduced performance.

4.6 Ping of Death

Ping of death has been quite successful in the past. This attack is performed by sending over-sized packets, larger than permitted IP size, i.e. 65535 bytes to the system. The goal with this attack is to halt or shut down the system. Attack command and results are given in the following table.

Command	We use a program that is written in C language to generate ping of death attack [32].
Result for Android 1.6	This attack is unsuccessful and the victim is not accepting the last offset of packet. It silently dropped the packet.
Result for Android 2.1	Its behaviour is same as for 1.6.
Result for Android 2.2	Its behaviour is same as for 1.6.

4.7. LAND Attack

In this attack malicious packets are sent to the target device with the same source and destination IP address. Many operating systems in the past ended up in a loop that lead to a halt of the system but newer systems should be patched. Attack command and results are given in the following table.

Command	sudo hping3 -1 192.168.0.103 -a 192.168.0.103
Result for Android 1.6	Android is protected against the Land attack. It just drops the packet of ICMP that has same source and destination address.
Result for Android 2.1	The packet is silently dropped.
Result for Android 2.2	The packet is silently dropped.

4.8. Teardrop Attack

This attack is performed by sending fragmented packets to the target. These fragmented IP packets overlap and can cause problems while the receiver tries to reassemble the datagram. Attack command and results are given in the following table.

Command	We used a program that is written in C language to generate teardrop attack [32].
Result for Android 1.6	Android 1.6 is not prone to the teardrop attack. It silently drops the packets that have overlapping IP fragments during reassembly of the received fragmented packet.
Result for Android 2.1	Android 2.1 has same behaviour as for Android 1.6.
Result for Android 2.2	Android 2.2 has same behaviour as for Android 1.6.

4.9. Gratuitous ARP Request

It is an unsolicited request that does not need any reply from the network host. When a host announces its presence in the network then this Gratuitous ARP request is performed.
The attack can be done by sending forged MAC address to the victim containing another host's IP address. The purpose is to hijack the traffic of victim. Attack command and results are given in the following table.

Command	sudo packit -t arp -A 1 -x 192.168.0.102 -X 00:09:5b:e3:6d:79 -y 192.168.0.102 -Y ff:ff:ff:ff:ff:ff -e 38:e7:d8:f8:bd:28 -E ff:ff:ff:ff:ff:ff -i

	eth0 -c 0
Result for Android 1.6	When we sent faked gratuitous ARP request to the target, it updated its ARP cache. The target device now contained the wrong MAC address for that IP address. When it receives a new correct gratuitous ARP message then it gets a correct MAC address for that host.
Result for Android 2.1	We observed the same results for 2.1.
sResult for Android 2.2	We observed the same results for 2.2.

4.10. Fake ARP Reply using Broadcast

The Attacker sends spoofed ARP messages to a victim. The purpose of this attack is to hijack the traffic of a victim by sending fake ARP requests using the IP address of the host but with the attacker's MAC address. When this fake ARP request arrives in the victim system, it will change entry in ARP table to these new IP and MAC addresses. Attack command and results are given in the following table.

Command	sudo arpspoof -i eth0 192.168.0.101
Result for Android 1.6	In response of the ARP request we sent fake ARP reply. This Fake reply message contains attacker system MAC address with forged IP address. When target operating system sends the traffic to forged IP address then all traffic redirected to attacker system.
Result for Android 2.1	The behaviour against this attack was similar as of Android 1.6.
Result for Android 2.2	The behaviour against this attack was similar as of Android 1.6.

4.11. Summary

Although Android is a relatively new operating system, we can think about it as a mature system because it is based on the Linux kernel. Most conventional attacks were not successful, for example, Ping of Death, Land attack and Teardrop.

One security flaw in the Android system that we didn't see any special kind of protection against, was handling of flooding attacks, which were quite successful, and the system were unable to give a response to network during these attacks. We also noticed a significant difference in their behaviour against flooding attacks. Android 1.6's performance decreased, due to the user interface was affected during the attack and its response time increased. The most interesting observation during the experiment was that although Android 2.1 response time was not at all affected by the flooding attacks, there was a significant increase in

response time of Android 2.2 during the time. This is surprising since Android 2.2 is based on a newer Linux kernel (2.6.32) and it is expected to be less prone to attacks.

It was observed that the flooding attack against a closed port is also successful, and during this attack, the system is unable to communicate with any other device and performance is even worse than the performance during flooding of an open port.

Android is vulnerable to most ARP attacks like gratuitous ARP flooding, which can be used to broadcast wrong MAC entry for a particular IP address. This MAC address can be an unknown host who is not present on the network or it can be the address of the attacker's machine. This kind of attack can be used to reroute traffic or to deny any service to the victim. It does not matter whether this ARP reply is unicast or broadcast, in both cases the Android Smartphone enters this received fake IP-MAC binding in its ARP cache. Like most operating systems, Android doesn't have any mechanism to prevent this kind of attacks.

Chapter 5

Results and Conclusions

Android is an open source platform that provides lots of features for application developers. There is always a risk that a hacker can install some malwares which affects system functionality. However, the Android system provides a security mechanism which allows setting security permissions (normal, dangerous, signature level) according to the application's needs, and whether it wants to share some resources or not. By using the security permission mechanism a user can minimize the security risks for software which may contain bugs, malicious software, etc. and be able to protect his/her sensitive data.

Security assessment of the Android system is important because of its high usage is high on the Smartphone market. To asses this, we have performed penetration testing of Android Smartphones connected to a WLAN network. The first phase of our work was to gather information about the Android system such as that its version history, architecture and security mechanism.

First, we performed various scans to find vulnerabilities in three different Android versions. When doing this, we were able to gather information regarding the target host such as whether it is active or not, and see the status of different ports and the services running on it. We have also shown that it is possible to perform OS fingerprinting over the network so an attacker can determine the version of the operating system, which in turn can be used to find security flaws in the connected device.

Secondly, we performed different types of attacks against the Android network stack. Starting with Flooding Attacks which can be done in different ways like SYN, UDP, ICMP flooding. These attacks can lead to denial of service by consuming all resources such as bandwidth, memory and CPU.

We have also performed some historical attacks like Ping of Death, Teardrop and the Land Attack. The Android system is secure when it comes to dealing with oversized ICMP datagram’s and it just ignores the oversized packets. It does the same with the Teardrop and Land attacks.

Most link-layer attacks were successful on the Android system. ARP poisoning was successful and it was performed by sending fake ARP updates to the target host. Consequently traffic was redirected to a destination chosen by the attacker. DoS (Denial of service), Session hijacking and Man-in-the-Middle attacks were successful at link layer by sending false MAC addresses.

The main purpose of this work has been to perform penetration testing of Android Smartphones. The Android system does not contain hardware encryption, security keys and data recovery facilities, just to mention some shortcomings in the Android system. However, some security mechanisms are used in the Android system which is able to control the security risk, for example the functionality for users to grant privileges to applications when

installing them. Therefore, we can conclude that the Android system is a relatively trusted OS and our findings in this report should inspire the mobile OS vendors to address the security flaws we have found and improve system functionality, especially the behaviour when the devices are flooded with various kinds of traffic.

5.1. Future Work

Due to the growing number of malware, cross scripting attacks, SQL injection and buffer overflow attacks, future work is needed:

- Security analysis of Android Libraries and run-time environment.
- Analysis of security in Application layer protocols of Android-based Smartphones.

There are so many C/C ++ and java libraries in Android system such as the media, graphic library, etc. Their run-time environment is different than normal applications in the Smartphone and therefore, more effort should be spent to investigate the security issues in this area.

References

[1] Consumers Now More Likely to Buy Androids Than iPhones, http://www.mark etforce.com/2011/02/consumers-now-more-likely-to-buy-androids-than-iphones/, Accessed on February, 2011.

[2] Penetration test, http://en.wikipedia.org/wiki/Penetration_test, Accessed on February, 2011.

[3] Special Publication 800-115, Technical Guide to Information Security Testing and Assessment, September 2008 (replaces SP800-42), Accessed on March, 2011.

[4] Schneidawind, J: "Big Blue unveiling", USA Today, November 23, 1992, page 2B

[5] Smartphone Market Share,http://www.theglobeandmail.com /news/technology/mobile-technology/google-topples-symbian-from-smartphones-top-spot/article1888557/?cmpid=rss1, Accessed on February, 2011.

[6] Re: GPLv3 Position Statement, http://lkml.org/lkml/2006/9/25/161, Accessed on February, 2011.

[7] Jukka Ahonen, "PDA OS Security: Application Execution", Publications in Telecommunications Software and Multimedia TML-C7, ISSN 1455-9749, Helsinki University of Technology.

[8] Arto Kettula, "Security Comparison of Mobile OSes", http://www.tml.tkk.fi/Opinnot/Tik-110.501/2000/papers/kettula.pdf, Visited February 2009.

[9] Sheikh Mahbub Habib, Cyril Jacob, Tomas Olovsson, "A Practical Analysis of the Robustness and Stability of theNetwork Stack in Smartphones", 11th IEEE InternationalConference on Computer Science & InformationTechnology.

[10] Windows Mobile 6.0, http://msdn.microsoft.com/en-us /library/bb1584 86.aspx, Accessed on April 2009.

[11] Windows Mobile 6.5 Release, http://www.microsoft.com/presspass/press/2009/feb09/02-16MWCPR.mspx, Mobile World Congress 2009, Barcelona, Spain, 16th Feb, 2009.

[12] Special Publication 800-115, Technical Guide to Information Security Testing and Assessment, September 2008 .

[13] http://developer.android.com/guide/basics/what-is-android.html,visited on 30 April 2011.

[14] Android - Opportunity, Complexity, and Abundance, http://ducks.blackducksoftware.com/~whitepapers/WP-AND-0910-UL-AC-W.pdf,visited on 4 may 2011.

[15] The Android Linux Partnership, http://www.brighthub.com/computing/linux/articles/34302.aspx, visited on 4 May 2011.

[16] http://www.edparsons.com/2007/11/android-and-lbs-in-the-stack-at-last/,visited on 24 May 2011.

[17] http://www.scribd.com/doc/39096336/Android-Theory

[18] http://developer.android.com/guide/basics/what-is-android.html

[19] http://www.ibm.com/developerworks/opensource/library/os-android-devel/

[20] Android - Opportunity, Complexity, and Abundance, http://ducks.blackducksoftware.com/~whitepapers/WP-AND-0910-UL-AC-

W.pdf,visited on 4 may 2011
[21] http://www.scribd.com/doc/25036401/A-Security-Overview-in-Google-s-Android-Phone
[22] http://developer.android.com/guide/publishing/app-signing.html

[23] Information about TCP/IP port assignments, http://support.microsoft.com/kb/174904, Accessed on February 2011.
[24] http://www.sans.org/reading_room/whitepapers/auditing/port-scanning-techniques-defense_70
[25] The Transmission Control Protocol, http://condor.depaul.edu/jkristof/technotes/tcp.html, Accessed on March 2011.
[26] http://condor.depaul.edu/jkristof/technotes/tcp.html
[27] OS Detection, http://nmap.org/book/man-os-detection.html, Accessed on March 2011.
[28] CAPEC-300: Port Scanning, http://capec.mitre.org/data/definitions/300.html, Accessed on March 2011.
[29] Http://www.trustedreviews.com/news/O2-Says-Apple-to-Patch-iPhone-SMS-Vulnerability, visited on 24 may 2011.
[30] http://www.pcworld.com/article/117278/pda_virus_found_in_the_wild.html, visited on 23 May 2011.
[31] http://ojs.academypublisher.com/index.php/jnw/article/view/0410968975/1396, visited on 23 May 2011.
[32] http://packetstormsecurity.org/files/view/14097/spike.sh.zip