

Master thesis

**On the Fly OS and Data Recovery for Android
Smartphones**

To 7748



Anwar Ghani

429-FBAS/MSCS/S08

Supervisor

Dr. Muhammad Sher

Professor

Co-supervisor

Shehzad Ashraf Chaudhry

Lecturer

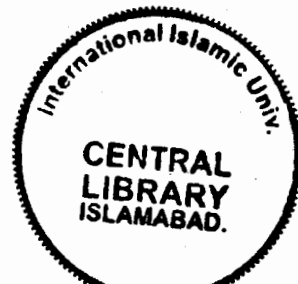
Department of Computer Science

Faculty of Basic & Applied Sciences

International Islamic University Islamabad Pakistan

2011

i



Master thesis

**On the Fly OS and Data Recovery for Android
Smartphones**



vrije Universiteit amsterdam

Anwar Ghani

EURECA Master Mobility Student Exchange program worked with:

Dr. Herbert Bos

Professor

Georgios Portokalidis

PhD Researcher

Faculty of Science

Vrije Universiteit, Amsterdam the Netherlands

2011

Master Thesis
**On the Fly OS and Data Recovery for Android
Smartphones**



vrije Universiteit amsterdam

Home Institute

International Islamic University Islamabad, Pakistan

Host Institute

Vrije Universiteit Amsterdam, Netherlands

Student Exchange Program

**European Research & Educational Collaboration with Asia
(EURECA)**

DEDICATION

DEDICATED TO

My Kids

Muhammad Kashif

&

Yasir Ghani.

Department of Computer Science
International Islamic University Islamabad

DATE: 11-05-2011

APPROVAL CERTIFICATE

This to certify that thesis submitted by Mr. ANWAR GHANI, 429-FBAS/MSCS/S08 is up to the standard and we hereby approve it for its acceptance as a partial fulfillment for the award of Master of Science in Computer Science.

1. External Examiner

Dr. Abdus Sattar

Former D.G

*Pakistan Computer Bureau, Islamabad
Pakistan*



2. Internal Examiner

Dr. Muhammad Zubair

Assistant Professor DCS

*International Islamic University, Islamabad
Pakistan*

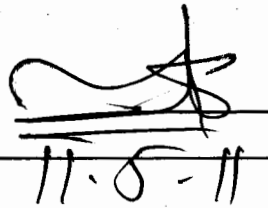


3. Supervisor

Prof: Dr. Muhammad Sher

Chairman DCS

*International Islamic University, Islamabad
Pakistan*

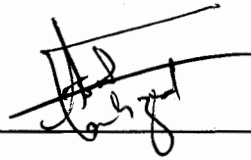


4. Co-Supervisor

Mr. Shehzad Ashraf

Lecturer DCS

*International Islamic University, Islamabad
Pakistan*



This research work is submitted to the

Department of Computer Science,

Faculty of Basic & Applied Sciences,

International Islamic University, Islamabad

***As a partial fulfillment of the requirement for the award of
the Degree of Master of Science in Computer Science.***

Declaration

I hereby declare that I wrote this thesis "**On the Fly OS and Data Recovery for Android Smartphones**" by myself and listed all used sources. It is further declared that I have done this research with the accompanied report entirely on the basis of my personal efforts, and full guidance of my supervisor professor ***Dr. Muhammad sher*** and co-supervisor ***Mr. Shehzad Ashraf*** at home university as well my supervisors at VU university Amsterdam. If any part of this work is proved to be copied from any source or found to be reproduction of any other research from any other training or educational institutions, I shall stand by the consequences.

Anwar Ghani

429-FBAS/MSCS/S08

Abstract

The use of handheld devices such as Smartphones has increased because of their broad application domain. These phones are used by people from every field according to their needs for several reasons; one of the most important is their portability which makes it easier to carry while traveling and at the same time it also provides almost the same functionality as a personal computer.

Besides the above mentioned advantages, it also has some limitations for instance, battery power limitation restricts the use of sophisticated and complex security software on these devices, because such software will not only severely affect the battery life but also the processing speed of these devices.

The main objective of this research work is design a sophisticated recovery system for ANDROID Smartphone that will automatically and transparently recover the phone to a stable position after it has been compromised or corrupted due to some external attack (hacked) or system errors.

The goal of this work is not only to come up with a solution for restoring a Smart phone to a stable position, but also make sure that the restoration process itself is well secured and cannot be interfered. Our approach (Safe recovery), will make the restoration automatic and transparent as it is carried out in recovery mode. This is the reason that this process cannot be interfered by any kind of attack; therefore, we believe that our approach to restore the phone, while it is in recovery mode, is safe.

Table of Contents

| | |
|---|-----|
| Acknowledgment..... | xii |
| Chapter No-01 | |
| 1. Introduction..... | 2 |
| 1.1 Background | 3 |
| 1.2 Objective & Contribution | 6 |
| 1.3 Thesis structure..... | 6 |
| Chapter No-02 | |
| 2. Literature Review..... | 9 |
| 2.1 Security Threats to Smartphones..... | 9 |
| 2.1.1 Platform Vulnerabilities | 10 |
| 2.1.2 Malware, Worm, Trojan horse..... | 12 |
| 2.1.3 Categories & Reasons of Vulnerabilities | 14 |
| 2.2 Counter Measure | 15 |
| 2.2.1 Available Security Tools | 16 |
| 2.3 Is ANDROID the only target?..... | 17 |
| 2.4 How to deal with the resource limitation problem? | 18 |
| 2.5 Dynamic Taint Analysis | 20 |
| 2.6 The problem statement | 20 |
| 2.7 Related Utilities..... | 21 |
| 2.7.1 iTunes for iPhone | 22 |
| 2.7.2 System restore for MSOX..... | 23 |

Chapter No-03

| | |
|--|-----------|
| 3. Methodology & Implementation | 26 |
| 3.1 Overview | 26 |
| 3.2 Automatic Backup Generation | 27 |
| 3.2.1 Working Mechanism of ABG..... | 28 |
| 3.2.2 Working Mechanism of ABG..... | 30 |
| 3.3 Model..... | 31 |
| 3.4 Architecture and Implementation | 32 |
| 3.4.1 Why recovery Mode? | 34 |
| 3.4.2 Our restore utility..... | 36 |
| 3.4.3 How to modify recovery image? | 37 |
| 3.4.4 Android Debugging Bridge | 38 |
| 3.4.5 Algorithm for overall restore process..... | 39 |

Chapter No-04

| | |
|--|-----------|
| 4. Results..... | 41 |
| 4.1 Comparison with Marvin Architecture | 46 |
| 4.1.1 Our approach VS Marvin Architecture..... | 47 |

Chapter No-05

| | |
|--|-----------|
| 5. Summary and Conclusions..... | 50 |
| 5.1 Future Work..... | 51 |
| 6. Appendices | 53 |
| 6.1 Appendix A..... | 54 |
| 6.2 Appendix B | 58 |
| 6.3 Appendix C | 62 |

List of Figures

Chapter No-01

| | |
|--|---|
| 1.1 Web based applications | 2 |
| 1.2 Layered view of ANDROID platform..... | 4 |
| 1.3 Detailed view of ANDROID platform..... | 5 |

Chapter No-02

| | |
|---|----|
| 2.1 Possible exploitable vulnerabilities of ANDROID platform..... | 11 |
| 2.2 Smartphone malware effects..... | 14 |
| 2.3 Appeared malware categories..... | 15 |
| 2.4 The remote monitoring framework..... | 17 |
| 2.5 Marvin Architecture..... | 21 |

Chapter No-03

| | |
|--|----|
| 3.1 Android Emulator 1.5..... | 28 |
| 3.2 Depiction of Timer program invoking ABG..... | 29 |
| 3.3 One server hosting many replicas..... | 31 |
| 3.4 Architecture of the restore process for ANDROID..... | 33 |
| 3.5 HTC Google Android G1..... | 34 |

Chapter No-04

| | |
|--|----|
| 4.1 Battery Consumption..... | 41 |
| 4.2 Output of ABG | 42 |
| 4.3 Test Directory with backup image | 43 |
| 4.4 HTC G1 | 44 |
| 4.5 Menu options in Recovery Mode | 45 |
| 4.6 Error in Recovery Modifications | 46 |

Acknowledgment

First of all I would like to thank the Most Merciful and Compassionate, the Most Gracious and Beneficent whose help and guidance always solicit at every step, at every moment.....yes I am talking about Almighty ALLAH, HE gave me the confidence and blessed me with knowledge which I didn't have.....!

*I pay my humble regards to Prof. Muhammad Sher, my teacher and project supervisor from my home institution **International Islamic University** for his kind patronage and guidance. It was his help and encouragement which made me face all the problems that I may not have the courage to face without his guidance.*

*I would like to pay my humble regards to Prof. **Herbert Bos** for offering his supervision to carrying out my thesis work. I personally feel proud to having him as my supervisor and instructor for my daily progress. I believe that this is his supervision that has broadened my knowledge. His personal interest in this work enabled me to solve most of the problems faced during this research easily.*

*I would like thank my co-supervisor at my home University **Shehzad Ashraf Chaudhry** from the core of my heart for his continuous support, motivation and guidance during my thesis work.*

*I express my sincere thanks to **Georgios Portokalidis** for his immense and untiring help and support throughout my project/thesis work.*

*I would like to thank **Dr. Philip Homburg** for his advice, guidance and cooperation during my project/thesis work.*

*I would like to express my gratitude to local coordinator at Vrije Universiteit Prof. **Patricia Lago** and the **International Office** for their timely guidance and coordination on all issues. I am also thankful to all the faculty members and instructors at the "Parallel and distributed computing systems" group at Vrije Universiteit and faculty members at DCS IIU for their encouragement and appreciation.*

*I would also like to express my heartiest regards to Prof. **Sasikumar Punnekkat** and the whole **EURECA** administration for providing me the opportunity to pursue my thesis work at Vrije Universiteit, Amsterdam.*

At last but not the least, countless thanks to my parents and my siblings for their continuous encouragement and support for my studies.

Anwar Ghani

MSCS International Islamic University Islamabad

Master Mobility

EURECA student Exchange program VU University Amsterdam

Chapter No-01

Introduction

Introduction

With rapid growth of Internet over the last few decades enormous changes have occurred in computer network infrastructure and communication technologies all over the world. This growth resulted in an increased concern for security. Software developed at organizations shifted from desktop applications to web based applications and also these application were ported to mobile world. Today iPhone is leading the market share with 64% shares while ANDROID stands at number 2 after its introduction to the market in 2008.

In On-line services such as banking, e-learning, shopping, education portals, more than 60 % of the software applications of the world are running over the web. Figure 1.1 represents the ratio between web and non-web based software. Thus, the usage of web has become an indispensable part of day-to-day activities, not only on personal computers but also on Smartphones. The ease and comfort that a Smartphone provides makes it increasingly important as a platform to access online services

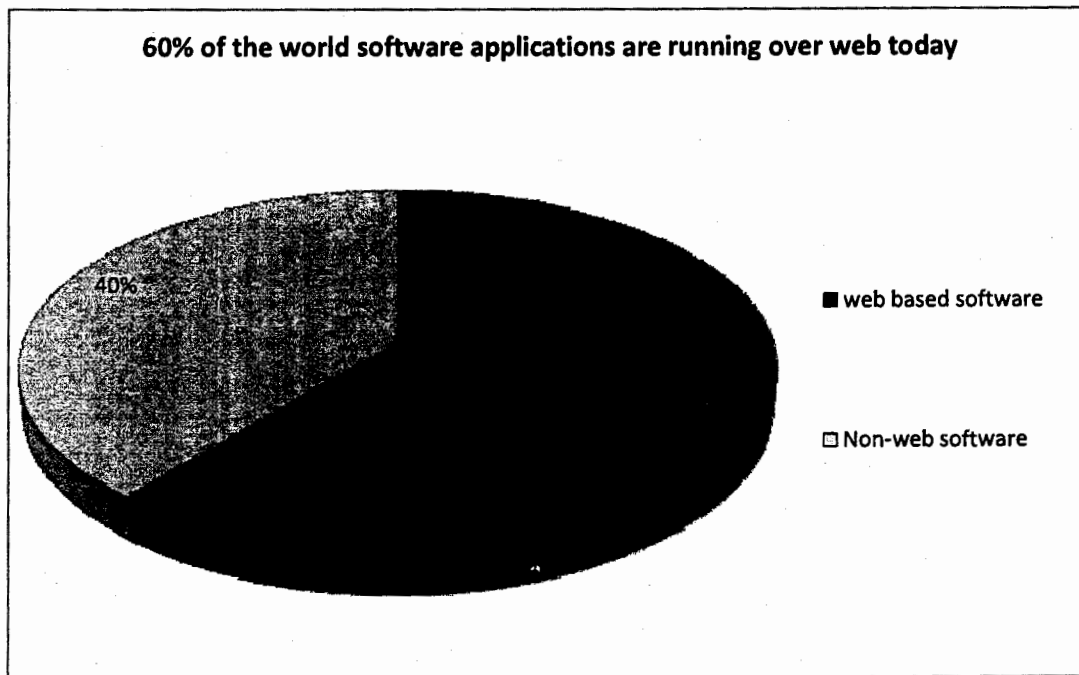


Figure 1.1: Web based applications

Alan Livingston (2004) in his article defines mobile devices as being '*small enough to fit comfortably into a purse, pocket or holster, so you can conveniently keep it with you at all*

times. [1]. Smartphone is actually a hybrid of a mobile and a PDA. It is similar to PDA in many aspects like using some of the same technologies while the most dominant feature of a Smartphone is to make phone calls. It can use any cellular networks just like any other simple mobile. Today wireless technology is becoming cheaper, faster and much more common. As we see the number of home users and institutions have increased, who deploy or adopt Wi-Fi kits to share lines and devices around the house or within the workplace. Future standards should allow wireless broadband data speeds over short distances and the development of wireless metropolitan networks so that user will be able to login to any network.

According to Wikipedia [wl: 1], a **Smartphone** is a mobile phone offering advanced capabilities, often with PC-like functionality (PC-mobile handset convergence). There is no industry standard definition of a Smartphone. But well known Smartphone examples include the iPhone, the Nokia N900 and the phones based on Android. Google Android is an open source project for smart phones (Mobile devices), recently introduced by Google to get into the mobile world. The ease and comfort that a Smartphone provides, makes it increasingly important as a platform to access online services.

The use of handheld devices such as Smartphones is increasing because of their broad application domain. They can support as many applications as a general purpose Desktop Computer. These phones are used by people from different fields according to their needs for several reasons; one of the most important ones is their portability which makes it easier to carry while traveling as compared to any other device like a laptop, and at the same time it also provides almost the same functionality as a personal computer. Therefore, these devices are the attractive targets for attackers. They still have vulnerabilities which can be exploited [4, 5] like embedding a malware in an application for ANDROID. As we discussed already, users employ their Smartphones for day to day activities like online shopping. They enter credit card details and other personal information into the phone's browser. Therefore among other reasons there is also a financial incentive for the attackers in attacking (hacking) these phones.

1.1 Background

On the official developer website ANDROID is defined as [wl: 2] “A software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language”. Figure 1.2 shows the layered view of the ANDROID architecture. A more detailed view is provided in figure 1.3 on the next page.

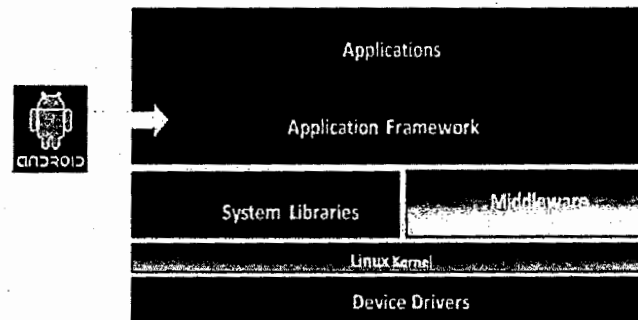


Figure 1.2: Layered view of ANDROID platform

The ANDROID operating system was first introduced by Android Inc. It is based on Linux kernel. Later Android Inc was purchased by Google and then by the “Open Handset Alliance” also called OHA. Google released most of the Android code under the Apache License, a free software and open source license. The goal of the ANDROID alliance is to give the consumers a far better experience than what is available on today’s mobile platforms.

The ANDROID allows developers to write and manage code in Java with many controlling libraries from Google. ANDROID has seen many updates since its first release. These updates serve typically two purposes. First, they fix bugs in the previous release identified by the user community. Second, they add new features and enhance performance of the base operating system.

In the last decade mobile communication has become more personal and ubiquitous communication with nearly three billion users worldwide. Because of this enormous amount of users and the lack of collaborative efforts made it challenging for developers, manufacturers and wireless operator to respond timely to the ever-changing needs of the mobile consumers. Introducing ANDROID is an attempt to position developers and manufacturers to bring new products faster and at lower cost. The ANDROID platform gives mobile operators and manufacturers' significant freedom and flexibility to design products [6].

The most important reason for the popularity of Smartphone is that they are becoming more similar in functionality to general purpose computers with a very small size providing portability. In addition to performing its basic telephony stack, calendars, games and address-books these phones are now being used for web browsing, reading emails, video and audio streaming, video conferencing, presentations and a lot more. A plethora of applications for these Smartphones is available in the market today. Some of them like navigation and location-sensitive information services are becoming increasingly popular [3].

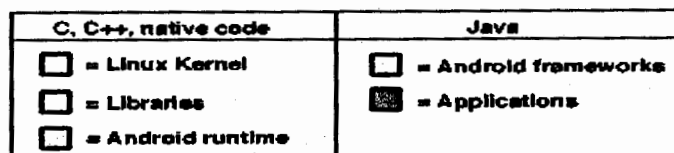
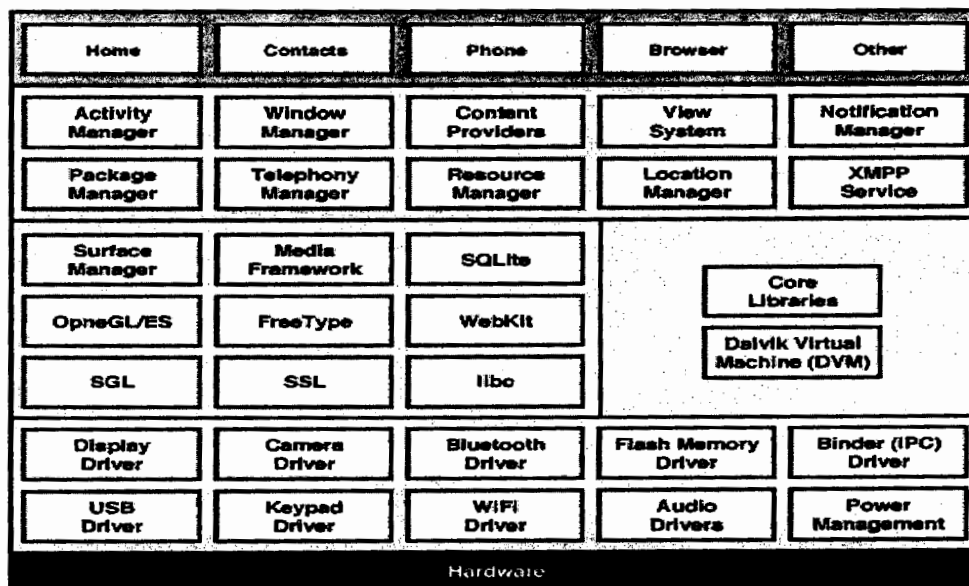


Figure 1.3: Detailed view of ANDROID platform

Figure 1.3 shows the overall architecture of the ANDROID platform. Our work lies in the C, C++ native portion of the architecture. We do not modify the kernel therefore, our work lies above the kernel level. To carry out our work we use the functionality and services of the Core libraries, DVM, and at run time libc. So this places our work in between the kernel layer and below the java based portion including the Android frameworks and Application.

In the coming few years hopefully we will have more stable and secure infrastructure for these phones as lots of people are involved in improving the platform and techniques for developing efficient applications suitable for devices like Smartphone. Also the technological developments will hopefully help to overcome different problems like resource limitation with these devices in order to give freedom to the developers to improve security and performance mechanisms. In the same way the number of mobile users is also continuously increasing which is encouraging not only for the people involved in the mobile industry but also for the businessmen and researchers at different educational or research institutions.

1.2 Objective and Contribution

The main objective of the thesis is to extend the design of Marvin architecture that can be used as a sophisticated recovery system for ANDROID Smartphone which will *automatically/transparently* recover the phone to a stable point after it has been compromised or corrupted due to some external attack (virus, spy software, hacked) or system errors. Marvin architecture basically works on the principals of execution replication where the security functions are decoupled from the phone to a powerful machine called the security server. Marvin architecture is discussed in detail in the later chapters.

Also we believe that our approach will be able to avoid the very famous resource limitation problem related to Smartphones particularly the battery life. The effect of security function decoupling through Marvin Architecture on battery is only 7%, which is acceptable as compared to the heavy and strong security mechanism proposed [4].

The goal of the research is not only to come up with a solution to restore a phone to a stable position but also make sure that the restoration process itself is well secured and cannot be interfered. With our approach (Safe recovery) the restoration process will become automatic and transparent as well as secure. The restoration is carried out in recovery mode so that it cannot be interfered. It is the first ever implementation of restoration technique for ANDROID Smartphone. Since the process is carried out in recovery mode due to which it

cannot be interfered by any kind of attack, therefore we believe that our approach to restore the phone, while it is in recovery mode, is safe.

1.3 Thesis structure

Due to time limitation and scope of the project our implementation is a part of the *Marvin architecture* [4]. We limited our work to restoring an ANDROID phone from a clean image on the security server. In chapter No-02 we will give detailed “Literature Review” discussing different approaches proposed till today and different problems related to these approaches due to which they could not get popularity. We will end the chapter with a brief description of iTunes for iPhone and system restore for MSOX as examples of the system restore.

Chapter No-03 will discuss implementation of our work with a general description of model used. We will give detailed descriptions of different aspects of Marvin Architecture which plays an important role in the current research effort. We will explain in detail the question why we decided to execute our code on the phone while it is in recovery Mode and how we modified the recovery image for the phone and what modifications we have made in order to make it compatible with our approach. In chapter No-04 we will summarize our work. We also provide the conclusion drawn. We also discuss how this approach can be extended to support multiple devices, user identification in order to restore the specific user data even if the phone is changed. We also discussed the feasibility whether this approach can be extended to other devices.

Chapter No-02

Literature Review

Literature Review

Almost every Smartphone has a recovery mechanism for restoring user data, application and setting and preferences in its own way. The technique to restore a Smartphone to a stable point varies with different phones. As these phones are very popular now days, a plethora of knowledge is available regarding different techniques on the web. Mobile computing is a very hot research area today particularly Smartphones which are driving today's market. Different techniques are studied at research and educational institutions to know about their weaknesses and suggest further improvements. One of these devices is ANDROID newly introduced and a very active research area. Now ANDROID is the second largest share holder (25%) worldwide [wl-1]. The reason for its popularity is its open source nature. Everyone can have access to, and modify the source code. Due to this reason it has a very large community on the web ready to help in almost all kind of problems. As ANDROID is a new introduction to the market there are still many areas to be covered particularly security issues.

Here in this chapter we will review literature regarding security issues in ANDROID, the loopholes in the platform, and threats those are being faced by these devices. We will not limit ourselves to ANDROID only but we will also consider issues related to other Smartphones which are related to our work. Along with discussing different threats being faced by the Smartphones, we will study techniques available to counter these threats. We will explain the complete context of our work to make clear the importance and scope of our work.

2.1 Security Threats to Smartphones

Smartphones have become obligatory tools for today's highly mobile workforce. These devices are not only small in size but also relatively inexpensive. They can be used not only for voice calls, simple text messages, and Personal Information Management (PIM) (e.g., phonebook, calendar, and notepad), but also for many similar functions done on a desktop computer. Some important functions include sending and receiving electronic mail, web browsing, storing and modifying documents, delivering presentations, and remotely accessing data. In general Mobile devices, and particularly Smartphones, have expanded beyond the basic telephony and communication origins to become accomplished photo- and video-

creation tools. Many users store and share documents, data, videos, and photographs on their phones [3].

Smartphones are being used to store valuable information like documents, PIN numbers, access codes for offices or secure building, passwords, and Bank Accounts credit card numbers for online shopping etc. it has been shown by a recent survey that two third of PDA user do not use any encryption to protect their data on these devices (Computer Business Review, 2004) [wl: 5].

Most of the Mobile handheld devices today also have specialized built-in hardware, such as a camera, a Global Positioning System (GPS) receiver, and reduced-size removable-media card slots. They also offer a range of wireless interfaces, for example infrared, Wireless Fidelity (Wi-Fi), Bluetooth, and one or more types of cellular interfaces [4]. Some of these interfaces are used for data transfers like infrared, Bluetooth and Wi-Fi. The data transferred from a phone or system to a Smartphone may be infected and may become a security risk. At the same time this operation (Data transfer) can be interfered by an outside attacker.

One such example is the famous attack associated with Bluetooth known as *Blue Bugging*. In the past different phones like the Nokia 6310, the Sony Ericsson T68 and the Motorola v80 have already become the victims of *blue bugging* (an attack that exploits famous Bluetooth bug). Another example is *Lasco*. Lasco is a malware which spreads using Bluetooth and using social Engineering to infect all Symbian Installation Source (SIS)-files [5]. It means that phones having Bluetooth may still be vulnerable to such attacks. It is also possible that some hardware components of the cellular phones like Flash storage (SD card and internal and memory) can be worn out or the battery can be drained easily by keeping the CPU running or by keeping the phone awake all the time[6]. Any vulnerability in the core libraries or kernel module can result in very dangerous attacks. In such situation an attacker can run any malicious code with high privileged mode or even can gain the full control of the device. The fact that the ANDROID source is publically available increases the intensity of this threat because some processes at system level have root access and they run with full privileges. It has been shown in the security analysis that the ANDROID is more vulnerable to local host based exploitation attempts.

2.1.1 Platform vulnerabilities

It is a fact that common users are usually unaware of the internal technicalities and vulnerabilities of their devices. It has been shown in the abovementioned study that the permission system on ANDROID is not sufficient, in which case the installation of an application that has malicious behavior can take place. This can result in high security risk. In the past attacks like buffer overflow, lunched through the phone web browser due to an outdated native library. This increases the probability of injecting malicious code via web browser. Security risk analysis is shown via a very descriptive figure 2.1 below taken from [6] based on the following parameters.

- (1) *Private/confidential content* (pictures, contacts, emails, documents etc.);
- (2) *Applications and services* (phone, messaging, emailing, Internet);
- (3) *Resources* like battery power, communication, memory and processing power (CPU);
- (4) *Hardware* includes the device itself, external memory card, battery and camera.

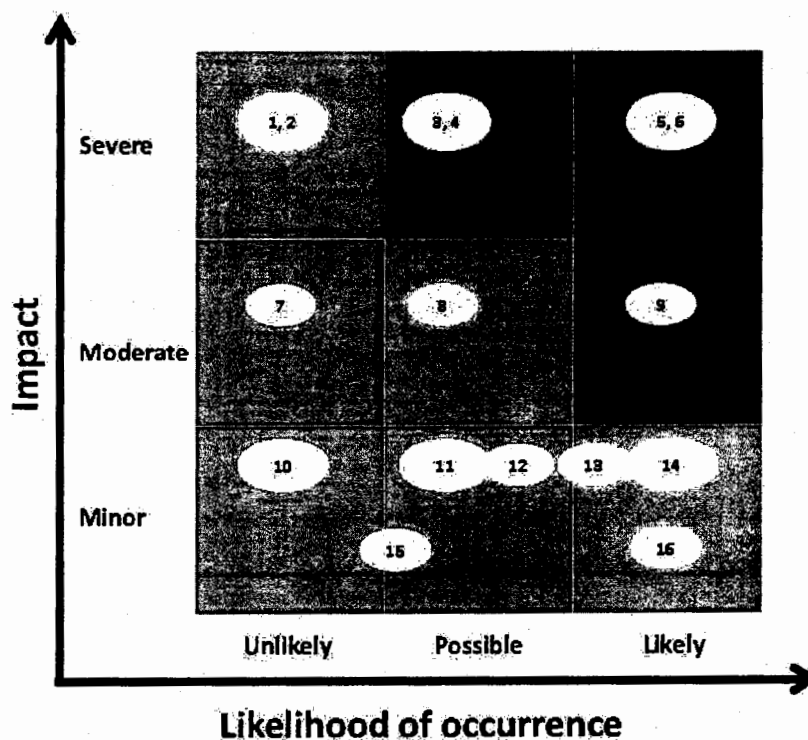


Figure 2.1: Possible exploitable vulnerabilities of ANDROID platform

On looking at the figure it is clear that cells at upper right corner with *red color* show the likelihood of exploitable vulnerabilities. This figure is based on a very recent study on security analysis of ANDROID. It means the platform is still having much exploitable vulnerability. It shows that this platform still needs security mechanisms providing sufficient satisfaction to the common user.

Figure 2.1 represents five most important threat clusters which should be countered by employing proper security solutions/capabilities. These threat clusters are obtained by grouping similar threats assigned with the highest risk [6]. The evaluation is based on assessing the impact and likelihood of various threats exploiting vulnerabilities in Android in order to harm, disable or abuse the confidentiality and/or availability and/or integrity of the above mentioned components.

Threat cluster-1:

Compromising availability, confidentiality and/or integrity by maliciously using the permissions granted to an installed application. This attack scenario is likely to happen and potentially has a high impact on the device.

Threat cluster-2:

Compromising availability, confidentiality and/or integrity by an application exploiting vulnerability in the Linux kernel or system libraries. This scenario was proven possible and our security analysis shows that additional vulnerabilities are likely to be found. Although, it has a low probability of occurring, it carries a potential to inflict severe damage.

Threat cluster-3:

Compromising availability, confidentiality and/or integrity of private/confidential content. Contents on the SD card are not protected by any access control mechanism. Additionally, wireless communication can be eavesdropped remotely.

Threat cluster-4:

Draining resources. There is neither disk storage nor memory (RAM) quota per application. Hogging the CPU is also possible.

Threat cluster-5:

Compromising of an internal/protected network. Android devices can be used to attack other devices, computers or networks by running network or port scanners, SMS/MMS/email worms and various other methods of attack.

In addition to the basic telephony function Smartphones are used for different purposes like social interaction by using a digital shared space among different users where each user can post his/her messages, file sharing, localized dating services, Community building and mobile blogging [3]. The properties like portable size, high accessibility and the capability to create video, audio and text content along with performing effective communications pose threats and security risks. Many competitors from the software development world are porting their famous application on to Smartphones. Examples include famous voice over IP client for general purpose computers namely *Skype*, *Smart VoIP*, *Google talk*, *msn*, and similarly the other famous applications like *Google Search*, *Google maps*, and *navigation software* (e.g. TomTom navigation) etc. they are also being used on Smartphone now, for example, *fring* is used to work as *Skype* client on ANDROID. Similarly *Facebook* and *YouTube* and many other gaming applications for Smartphones are common these days.

2.1.2 Malware, Worm, Trojan horse

Another important threat that Smartphones are facing today is the introduction of malwares. Main focus of these kinds of attacks is Smartphones with Symbian operating system. Reports from very authentic sources like Kaspersky lab, Filar, McAfee, Symantec, Sophos, show that 288 malware were found till the end of 2008. But at the same time reports from F-Secure in Helsinki that the counted number of malware is 418, means there are several malware without any publically available descriptions [7]. The effects of these malware on different components of Smartphones are shown in figure 2.2 on the following page. From left to right the fourth component representing none means these malware are neither harmful nor beneficial or their activities may be unknown.

The propagation channel for these malwares may be an installation file which needs user interaction. That is the reason that most of the Smartphone malware are being categorized as Trojan Horses. Some other sources of propagation may be Bluetooth, MMS etc.

Smartphone Malware Effects

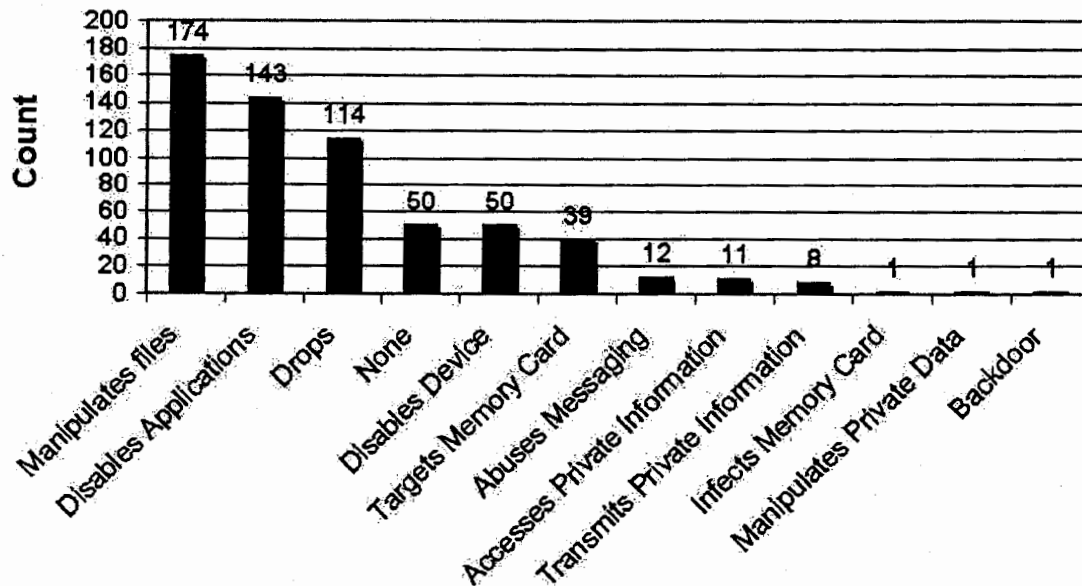


Figure 2.2: Smartphone malware effects

The calculation shows that major contribution to the Smartphone malware is from Trojan horse. 84% of the malware found for Smartphone contain “Trojan Horse”, 15% “Worms” and 1% “Viruses” shown in figure 2.3 on the next page.

It is also shown through practical examples in [7] that a malware with malicious behavior can get into the device if it gets executed through undocumented ANDROID java functions. It is also stated that currently the permission system of the ANDROID could be bypassed which show a very serious threat to this platform.

The threats we explained are notified by Wayne Jansen and Karen Scarfone in their publication [8] as “Electronic eavesdropping on phone calls, messages, and other wirelessly transmitted information is possible through various techniques. Installing spy software on a device to collect and forward data elsewhere, including conversations captured via a built-in

microphone, is perhaps the most direct means, but other components of a communications network, including the airwaves, are possible avenues for exploitation."

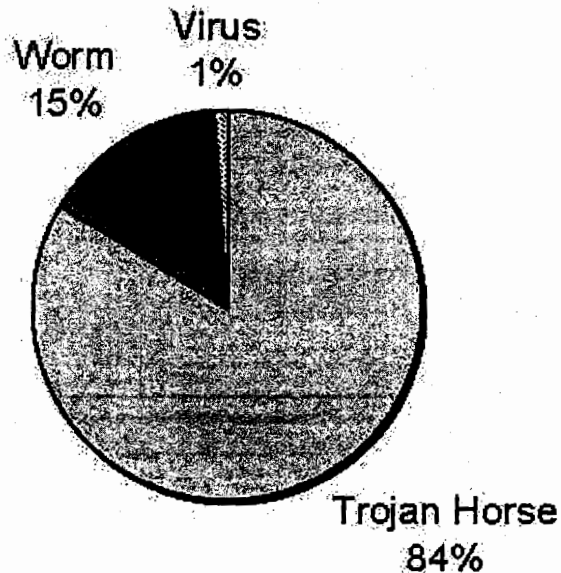


Figure 2.3: Appeared malware categories

2.1.3 Categories and Reasons for Vulnerabilities

In the world of Information Technology things like software evolved over time. As the time passes different holes and weaknesses are identified and counter measures are taken accordingly. As we discussed, the Bluebugging effect, although it has already been taken care of yet still danger is there that it may be exploitable in some other way. From our discussion in the previous sections, we can conclude that the security threats to Smartphone are mainly due to two reasons:

- 1) Their size and portability
- 2) Their available wireless interfaces and their services

Some of these are only related to physical security and the others are software holes and breaches in the platform and device. The detailed categorization of different threats is as follow.

- Loss, theft or disposal
- Unauthorized access
- Malware
- Spam
- Electronic eavesdropping
- Electronic tracking
- Cloning
- Server side Data

The first point (“loss, theft or disposal”) is related to physical security. Phones like ANDROID can become very easy targets of such attacks because of their small and portable size. If sufficient security measures like password and PIN code authentication are not in place, it may be impossible to gain access to the device. This access to the device can result in the exposure of private and sensitive information that is of critical importance to business organizations or personnel.

The rest are different security threats being faced by ANDROID and most other Smartphones because of different exploitable holes in their platform or applications developed by developers with malicious behavior.

2.2 Counter Measures

To counter the challenges presented by attackers to ANDROID and many other Smartphone several techniques have been developed and proposed. To cover different security breaches in the ANDROID platform it has gone through many upgrades. But still it is not sure that the platform is fully secure. There still may be some holes which can be exploited by the attacker to gain unauthorized access to the system. The basic function of an upgrade is to cover the security hole of the previous version and also provide extra functionalities to developers and users. In ANDROID many security threats have been recognized and counter measures are already been taken to stop any attacks which can come through that way. For example, in the past Bluetooth was used as a bridge to inject different malicious software or hack into different devices. This has already been taken care of by setting some rules for Bluetooth in ANDROID. As the device can make the Bluetooth connection as undiscoverable and even if the connection is set to discoverable it is only for short period of two minutes. At the same time in order to inject data into the phone the owner needs to accept the connection. Also, the

phone is secure against the SQL injection attacks except that the contents of SD-card are exposed to the attacker [6].

Among many approaches one based on remote monitoring [9] where the phone is monitored for anomalies remotely. Information also called *feature vector* needed to carry out the monitoring function are sent to the remote machine where it is analyzed for anomalies. The monitoring is done remotely due to the reason that these kinds of devices have the popular limitations related to their capability and hardware. The work is based on Symbian system and windows mobiles. And the aim of this work is to avoid the resource limitation problem associated with these phones.

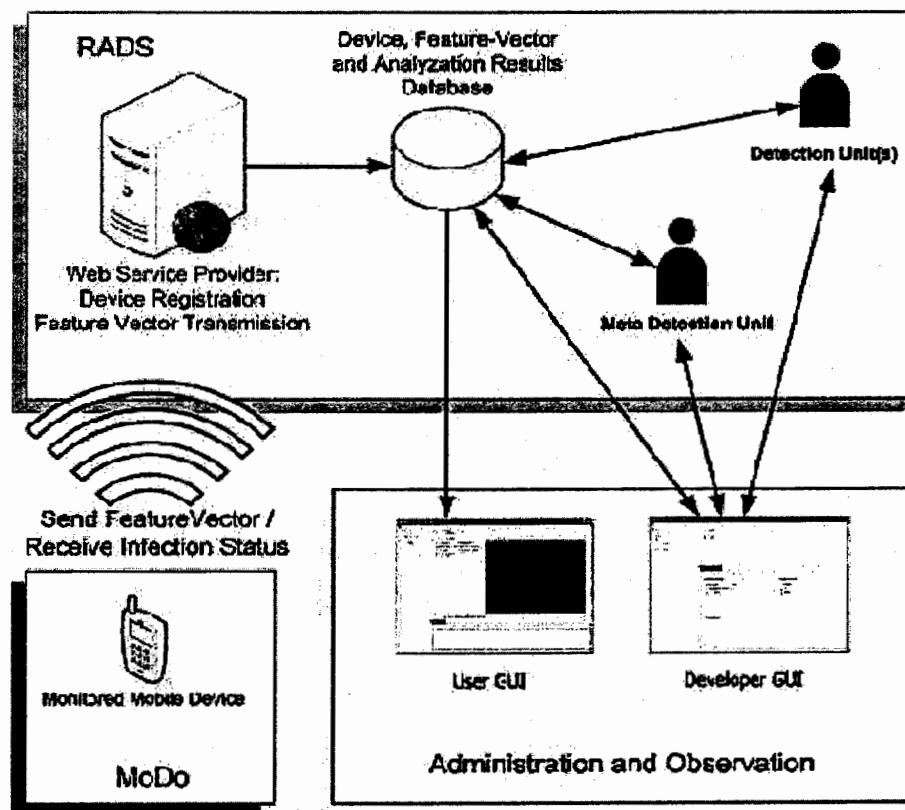


Figure 2.4: The remote monitoring framework [9]

This paper is limited only to monitoring the phone for anomalies to check if there is any threat that should be cured before it becomes a problem. It does not contain the solution how to cure it. The study emphasizes on the using sophisticated artificial intelligence techniques

for the anomaly detection. But our approach is not only to detect threats/anomalies but also to provide measures how to avoid them and how to recover from them when they already started their steps to challenge the system. The focus of our study will be ANDROID.

In addition to this approach many tools have been developed to serve as security techniques for ANDROID. But the point here to make is that almost all of these tools are local to the phone and no matter how light they are as their number is growing they still will affect the phone which can result in the resource limitation problem. Also, if they are developed as light as possible to fit in the mobile environment they may still not be able to cope with complex situations. We will discuss a few of them briefly here. Many of these have been listed in "Top 100 Network Security Tools". The listing is available at (<http://sectools.org>) [wl-6].

2.2.1 Available Security Tools

Some of the currently available security tools, as mentioned in [2], include *Clam* as anti-virus toolkit which is an open source (GPL) designed for UNIX and can be modified to use for ANDROID. Its basic functionality is email scanning on email gateways. *Netfilter*, as a firewall is a set of hooks inside Linux kernel which allows kernel module to register call back functions with the network stack. *Chkrootkit* as a Rootkit Detector scans for signs of rootkits, worms and Linux kernel module (LKM) Trojans. It can inspect binaries, check logs, check network interfaces and also look for hidden files. For intrusion detection *Snort* can be used which is a lightweight network intrusion detection and prevention system. It can be used on IP networks that excel at traffic analysis and packet logging. There are lots of others including *Nmap*, *strace*, *OpenSSH*, *Bash* and *Busybox* etc. If we look at all these tools it is obvious that most of them are specialized in nature and none of them is providing generic means for security. There is a need of system which is more generic and provide as many function as possible at the same time.

Many techniques had been proposed for dealing with different kind of attacks like intrusion detection [9, 10, 11, 12, 13, 14, and 15] in more or less static form. But mobile applications are shifting from standalone environment to collaborative and more dynamic nature resulting in internal exposure [16]. To deal with the problem we need a dynamic and reliable technique. But again we will say that implementing such dynamic and sophisticated technique on the phone may result in other problems.

Today more than 60% of the world software applications are running on the web. Most of these applications are being ported to the mobile world. Even in some countries like Germany the number of inhabitants is exceeded by the number of mobile devices [9]. At the same time the number of mobile applications is increasing day by day. To meet the market demand and the user's requirements the developer are developing new and complex applications with extended functionalities or rather these functionalities are being incorporated in the existing applications. As a result, the complexity of the applications grows which affects the battery life and other hardware resources of the device. In devices like Smartphones battery is the most critical resource.

2.3 Is ANDROID the only Target?

ANDROID is the recent introduction in the mobile world which became popular very quickly in the community. Because of the facilities, functionalities and interfaces provided by ANDROID, it is becoming an important target for attackers. To provide as much functionality as a general purpose computer, also implies an increase in the complexity of the software used on these phones. Since these phones are being used by people for their daily activities like shopping, web browsing, emails and presentations they may contain sensitive and private information like PIN code, Passwords or Credit card details. Also the increase in software complexity may result an increase in the exploitable vulnerabilities and bugs [7]. Due to these reasons these phones are not only becoming attractive targets for hackers. They also offer financial incentives in case of successful attacks.

These problems are not only associated with ANDROID but rather shared by the whole Smartphone family. All of these phones are exposed to these kinds of attacks in one way or the other. Windows mobile also share some problems with the normal PC's. If we talk about normal PC viruses each day more than 30,000 new viruses are introduced. Many attacks which are possible on PC's are also threats for windows mobiles. The iPhone is also vulnerable to these kinds of attacks [wl-3] because it belongs to the same family as ANDROID and shares problem like resource limitation. To overcome these problems the iPhone has gone through many developments. New models and software versions have been introduced. But still the latest 3Gs iPhone is not secure and has been cracked [wl-4]. It means the use of these phones for commercial transactions is still a security concern which may lead to the exposure of private and sensitive information. Especially for online shopping like

buying applications, books, garments, or any sort of shopping, credit card numbers and passwords are required to be entered in the phone based web browser which could become compromised and may be misused if there are no sufficient security measures available.

2.4 How to deal with resource limitation problem?

Having looked at these problems the first solution that comes to one's mind is to have very good and generic security (anti-virus etc) software on the phone which shall ensure the security of these phones. We know that the number and complexity of mobile applications is increasing day by day, hence it needs generic and sophisticated security software. We also know that generality and sophistication of any software leads to complexity and increase in size, and it will require more resources like battery life and high processing speed to run on a mobile device.

Unfortunately, adding more functionality to mobile devices along with heavy security mechanism affects other resources as well as it reduces the processing speed of the device and in particular, the battery life of a mobile device is directly dependent on the types of applications are being run on the phone. If the applications running on the phone are heavy in terms of graphics or processing, it will consume the battery power very fast thereby reducing the battery life of the device. Since battery life is very important it should be used as efficiently as possible.

Now there are a number of security challenges. First we want to add security against attacks to the phone. Second we want to make security solution cheap in terms of resource usage. In this thesis we will assume the solution provided in the "Marvin architecture" which decouples the security functionality from the phone and instead pushes it to a powerful machine called a security server which runs an exact replica of the phone with heavy security measures as we do not have such limitations on this machine as on mobile devices [7]. Marvin proves that the problems due to limitation of resources can be solved by moving the security function to a machine which is more powerful so that we can apply heavy and more sophisticated security mechanisms on the Phone's replica and also it doesn't suffer from resource limitations like power or battery life. The replica runs under heavy security on the security server duplicating the exact execution of the phone. The phone is synchronized with the replica through the Marvin protocol.

The replica on the security server is monitored under sophisticated security mechanism and if there is anything (virus, any intrusion attack etc) dangerous found i.e, the security checks indicate an attack, the user is notified by the security server that the phone has been compromised. The security server also suggests that the phone should be restored as soon as possible. If the user does not have a chance to start the restoration immediately, at least he/she can limit or even stop using the phone until it has been recovered. In Marvin Architecture a prominent technique to determine attacks and the amount of data that is compromised and to which extent another technique called "*Taint analysis*" is suggested [17].

Since our thesis work is based on *Marvin Architecture* shown in figure 2.5, the contribution of the thesis intends to address one of the basic and important problems in the architecture as to how restore a phone after an attack is launched and detected. This architecture enabled us to carry out the detection and prevention on a powerful machine with power detection and prevention mechanism (software) so that problems like resource limitations are not faced. As we discussed the *Marvin Architecture* affects the battery life of ANDROID Smartphone to the extent of only by 7%. This is not a significant decrease in favor of a strong and advanced security technique.

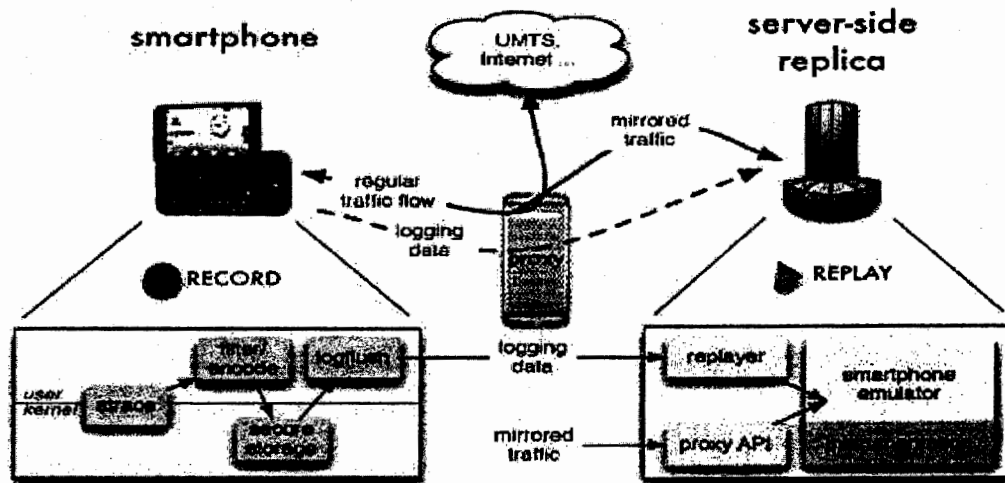


Figure 2.5: Marvin Architecture

A high-level overview of the Marvin architecture is illustrated in Figure 2.5. We sketch the basic idea here. A tracer on the phone intercepts system calls of, and signals to, the set of processes that need protection and record them in a log file. The file is store in a secure location on the phone. This set comprises all processes on the phone that may be attacked. It is typically a large set that includes the browser, media players, system processes, and so on). A replayer on the security server subsequently replays the execution trace, exactly as it occurred on the phone, while subjecting the execution to additional instrumentation. The transmission of the trace is over an encrypted channel. The proxy in the middle redirects the incoming and outgoing traffic to the security server in order to decrease the burden of communication for synchronization on the phone and save the battery life.

2.5 Dynamic Taint Analysis

Let discuss briefly the security technique proposed in Marvin Architecture to run on the security server and monitor the replica for different anomalies that is Dynamic Taint Analysis. We will not go into details since this topic is not the focus of our work. Dynamic taint analysis technique can automatically detect overwrite attacks, which include most types of exploits. It a powerful intrusion detection mechanism which can detect *buffer overflow*, *format string attacks*, *double frees*, *heap smash* and many more which changes the control flow of a program. It also does not need source code or special compilation of the program it is monitoring and hence it can work with commodity software. In order to protect vulnerabilities this mechanism uses automatic signature generation, as manual signature generation is not able to respond quickly enough to attacks like worms. Interesting thing about this technique, it doesn't produce false positives which shows that the technique is very reliable.

2.6 The Problem Statement

Smartphones are getting popularity very rapidly in the market today because of their broad application domain. They are used everywhere in educational research, in business community and by home users because it is portable. It is not only a mobile that was famous for its basic telephony function. Rather it resembles general purpose computers in almost all the functionalities in addition to their basic telephony function. Resembling in functionality does not mean that these phones are also the same in processing power or hardware resources as general purpose computers. This is the reason that these phones share some problems with

general purpose computer but at the same time they are not equivalent to these machines in counter measures. The functionality resemblance is at the software level but when it comes to the hardware level these phones suffer from resource limitation problems.

It means that the attacks which could be launched against a general purpose computer and may not be successful can be launched against these phones and may be successful because of the imbalance in counter measures between Smartphone and general purpose computer. They are also facing sophisticated security attacks like injection of spy software, virus attacks or hacking. At the same time these phones also suffer from resource limitation problems like battery life, processing power etc, which restrict the developers to provide a generic security solution to stop all the threats. After realizing that the problem cannot be solved, as long as, the execution and implementation of the security software is local to the phone. Therefore a solution is proposed in [4] by the name of *Marvin Architecture*. This architecture decouples security function from the phone to a powerful machine called security server by running replica of the phone on this machine under heavy security mechanisms. And whenever an attack is detected on the security server it is signaled back to the phone asking the user to take appropriate actions.

Now Marvin architecture did its job by decoupling the security function from the phone in order to avoid the resource limitation problem associated with the ANDROID and all other Smartphones. But what to do when an attack is detected? How to implement a restore mechanism through which we can restore the phone after an attack occurs? How to use system image taken from the replica on the server to restore the phone?

2.7 Related Utilities

Most of the mobile devices and other automated systems have the option to restore the to its factory setting. But this type of restore will not restore the data, applications and preferences that the user has stored on the phone or machine. Rather it will simply wipe the machine or device and will come to the point it was in on the first day when it was bought. It is desirable that every Smartphone and any other system that deals with information storing and processing must have a recovery/restore mechanism. For example Sony Ericson Xperia, Nokia N-97 one of many Nokia's Smartphones, the famous iPhone all have a recovery mechanism. Similarly if we talk about machines, all computers based on Microsoft windows have a check pointing mechanism for restoring to a previous stable state. Here we will

discuss two such systems, one that represents the Smartphone family iTunes for iPhone and other system restore in Microsoft windows which will represent machines other than Smartphones.

2.7.1 iTunes for iPhone

iTunes [wl-7] is a proprietary digital media player application, used for playing and organizing digital music and video files. The program is also an interface to manage the contents on Apple's popular iPod, the iPhone and iPad. *It is also used to download applications for the iPhone and iPod touch running iPhone OS 2.0 or later.*

iTunes was introduced by Apple Inc. on January 09, 2001 at the Macworld Expo in San Francisco and its latest version iTunes 9, was announced at Apple's September 2009 keynote "Rock and Roll". With iTunes an iPhone user can restore his/her contacts, calendars, emails, photos, music and videos using iTunes user account.

Each time a user uses "sync your iPhone" [wl-8] option, the data, settings, and other information on the phone are automatically backed up on the computer to which it is connected (e.g personal computer). If the user encounters a situation in which he/she needs to restore, though, all you need to do is download this back up to your phone and you'll be off and running again. It's a nice and easy process but the user needs to sync his phone periodically. Each time if there is some critical update or data change the phone needs to be backed up, so that it can be restored if anything bad happens in case.

At the start of this iTunes restore process the user is asked for account information and registration details. If the phone is hacked then the hacker may be able to read personal and account information of the user. It means this process though restores the phone, but it still looks vulnerable to such threats.

2.7.2 System Restore for MSOX

System Restore is a feature/component available on all Microsoft's Windows namely windows Me, Windows XP, Windows Vista and Windows 7 operating systems. Basically it works on the principle of check pointing. It allows for the rolling back of system files, registry keys, installed programs, etc., to a previous state which help in preventing a malfunctioning state or state of failure.

The server family of windows operating systems from Microsoft does not support system restores. The system restore in Microsoft windows usually uses the mechanism of *shadow copy* [wl-9]. This technique helps the system monitor block level changes and back up any file located anywhere in the system.

There is some manual work involved in System Restore. For example, the user may create a new *restore point* manually, roll back to an existing restore point, or change the System Restore configuration. Moreover, the restore itself can also be roll backed undone all the changes made by the restore process. This is a very reliable process of backing up and restoring a system. Although old restore points are deleted in order to keep the volume's usage within the specified limits. It also can affect performance of the system by continuously monitoring changes to some data and recording them for future restore. For many users, this can provide restore points covering the past several weeks but the problem is when user wants to restore the system to near past may be two or three days or one week, will not be possible. Users concerned with performance or space usage may also disable System Restore entirely. In this case if something goes wrong files stored on volumes not monitored by System Restore are never backed up or restored.

System Restore backs up system files of certain extensions (.exe, .dll, etc.) and saves them for later recovery and use. It also backs up the registry and most drivers so that it can reply the system state at that specific point in time. The restore process is different in windows XP and Vista. In windows XP it can use up to a maximum of 12% of the volume's space where as in Vista it uses up to 15% since it is designed for larger volumes. There is one problem when user wants to keep choices for backup at so many different points in time; it will consume large disk space. Running such technique on Smartphone is not feasible for several reasons. It can affect the battery life very drastically and also it can create memory problems for these devices while keeping backups for a little long time.

Chapter No-03

Methodology & Implementation

Methodology & Implementation

In this chapter we are going to discuss few things including the basic overview of our project, a general model that we have used in conducting our work, the basic architecture and the way we have implemented it, in the respective sections. We have used Linux distribution *Ubuntu 9.10, 10.10* with *HTC G1* ANDROID phone and the C and java as programming languages. We have used Jboss as the application server for this project. Also root access is needed on the device to test our work. For this purpose we rooted access our phone using the method available on the official developer's website [wl-10].

3.1 Overview

The idea behind this implementation is to have a security system for ANDROID that is controlled at a central base with powerful security measures. The central base is also called security server. It is a powerful machine with heavy security software running on it. This machine can host many replicas of different phones at the same time. It is just like a one-to-many association between security server and the phones. These replicas replicate the actual execution of a phone to which it is associated. There is no need to have a dedicated machine for each replica. In the same way all the replicas on a server are controlled by one security software. This software monitors all the executions and whenever it finds something suspicious it will immediately signal it to the phone associated with this specific replica to invoke its security and recover the phone to a stable position.

Now let's have a brief discussion of the security mechanism on the security server. In the digital world a huge number of attacking techniques are produced every day which can affect different system in hours or even minutes. Different studies showed that 288 Smartphone malware were found by the end of 2008 [7]. To overcome this problem we need a fully automated and filtering system which automatically detects such threats. *Dynamic Taint Analysis* [17] is one of the sophisticated and heavy security mechanisms dealing with such situations.

Knowing that Dynamic taint analysis (DTA) can detect and defend any vulnerability in the system; we do not have to worry about how an attack is detected and how different exploits can be protected. All we need is to concentrate on *how to recover the phone after it has been compromised due to some system corruption or some internal or external attack?* But again we have to say that this is a heavy mechanism and is not feasible to run on devices like

Smartphone which are power crucial machines. It will deplete the battery of the phone in seconds. But when the security decoupling mechanism as suggested in Marvin Architecture is used then *this technique is the perfect choice for such devices as it makes use of the security server. Let's first discuss how to generate system backup from ANDROID emulator running on the security server. This image can be used later for restoration process of the phone.*

3.2 Automatic Backup Generation

Before going into the specific details of the restore process's implementation and architecture we will discuss another topic crucial to our work that is "Automatic Backup Generation" or ABG. In this process we will discuss how to generate system backup of the replica (Emulator) running on the security server. For this purpose we have used ANDROID emulator 1.5 as shown below.

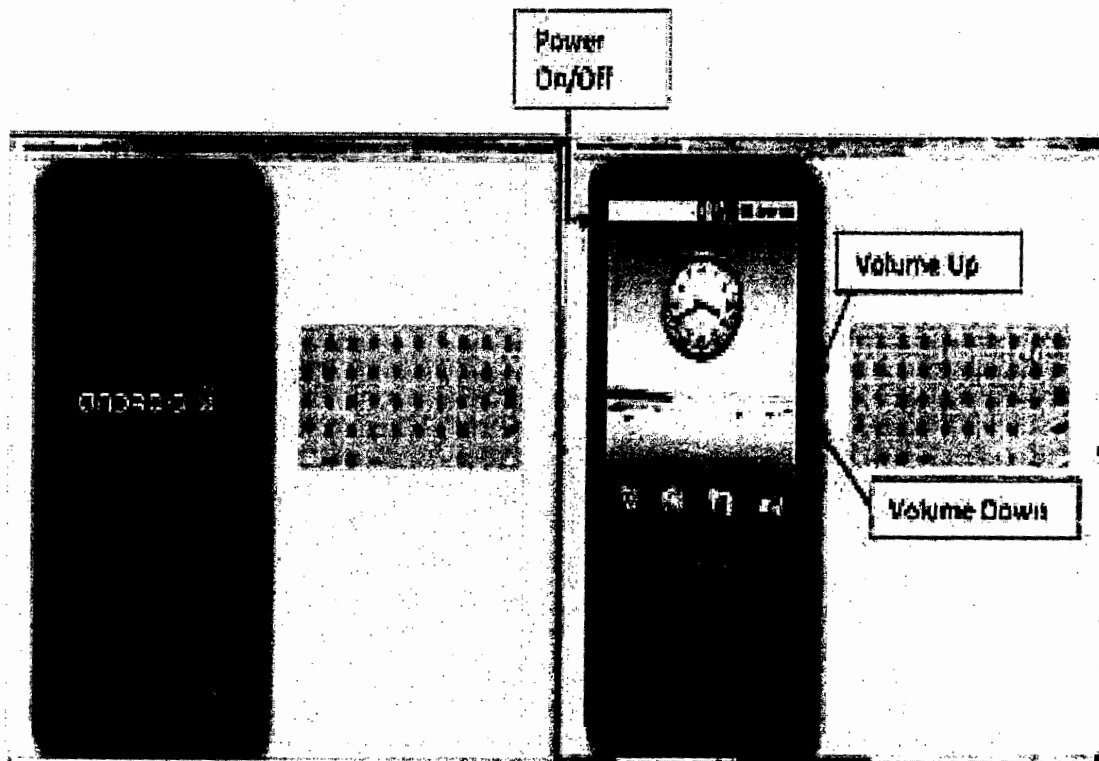


Figure 3.1: Android Emulator 1.5

ANDROID emulator can emulate any function that an ANDROID Smartphone can perform except making calls to other phones. It has a keyboard of type "QWERTY" just like personal computers. It perfectly emulates HTC G1 which we have used to test our work on.

3.2.1 Working Mechanism of ABG

Basically automatic backup generator (ABG) will work as a background process. This will not affect the normal execution of the replica by giving some output messages. It will quietly take image of the system invoke after a specified period of time. Of course this protocol can be changed to in many different ways. We will give a little bit details about this topic later in this writing.

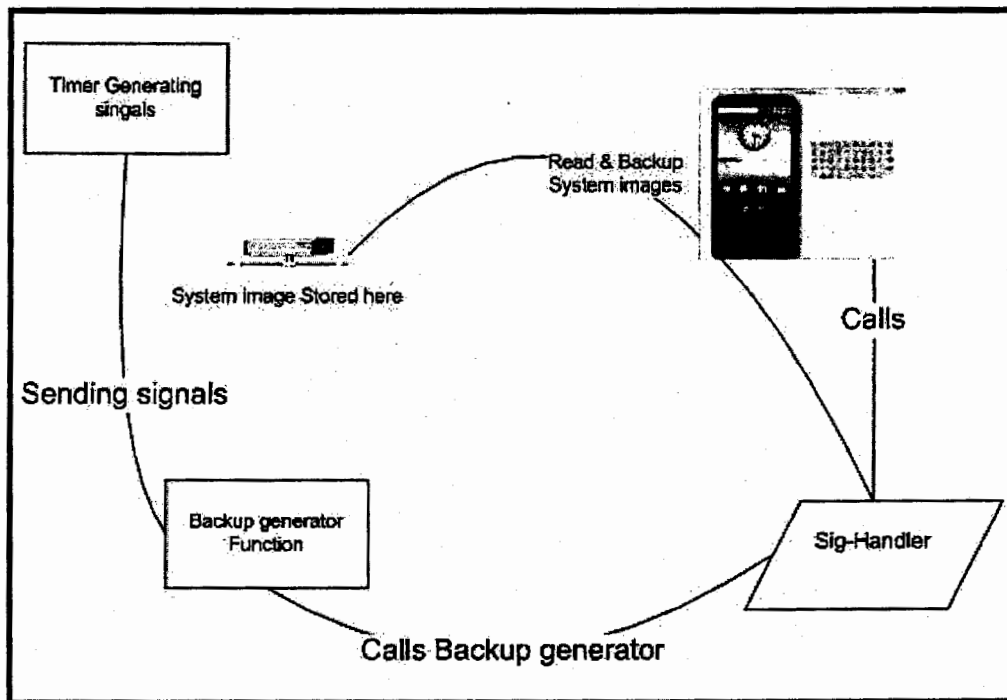


Figure 3.2: Depiction of Timer Program invoking ABG

We have set a timer that will invoke the backup generator on the specified time. The backup generator will then locate and read the image being used by the emulator as its partitions like /system, /userdata, or /boot. When the phone is synchronized with the replica all the execution trace is sent to the emulator (replica) and replayed. The changes are committed to the emulator that the user has made after the previous sync to these partitions which are basically images in ".img" format. This time runs on the security server locally and will periodically trigger the automatic backup generation.

The timer will generate a signal which will be intercepted by a handler function. This handler will then in turn invoke the backup generator function to start storing the current system

image of the replica. This image will of course later be used for restoring the phone in case it has been affected by some attack or system errors. One such example can be found in MS windows where it is known as check pointing technique.

The function will read the different partition of the replica like /system, /userdata, and /boot block-by- block and store it to permanent storage. This storage may be internal to the replica or may be some directory on the security server.

The sole purpose of the script is to generate signals. It has nothing to do with the rest of the architecture. We generated the signal through the script periodically but the protocol can be changed or even omitted depending upon the design decision. In the backup generation is activated right after the completion of any sync operation. Whenever a modification is synchronized from the phone with the replica, the operation will be preceded by backup generation operation automatically. It does not affect the architecture whatever the design decision for the backup generator is.

3.2.2 Algorithm for Automatic backup Generator

1. *get time of the day (from System)*
2. *to set timer*
 - i. *add a period value to the time*
3. *call the check timer with the new time value & period value*
4. *keep comparing the new value with the system time*
 - i. *when both values become equal*
 - ii. *generate backup (with a date & time stamp)*
 - iii. *increment timer by period value*
5. *Repeat steps 3 and 4.*

3.3 Model

The general and abstracted depiction of our work is shown in Figure 3.1. It looks similar to the famous client server model but actually it is not a client server model.

In client server model the execution is mostly carried out at the server by a client request but in this model the execution is fully local to the phone and is just replicated on the server.

The phone is independent in executing its applications or storing or modifying its data in any way. The connection lines are just the representation of the fact that one server can replicate the execution of many phones at the same time. Also these connections represent UMTS communication of the phone with server. This type of communication is used for synchronizing the phone with the associated replica on the server in order to make the execution smooth. As shown in *Marvin Architecture* [4], the incoming and outgoing traffic does not need to be synchronized with the replica, as the traffic is automatically mirrored on the server side replica from the proxy server in between the phone and the replica. In this way the communication between the phone and the replica (security server) is reduced so that to avoid any extra delays. All that needs to be synchronized with the replica is local execution and modification made by the phone.

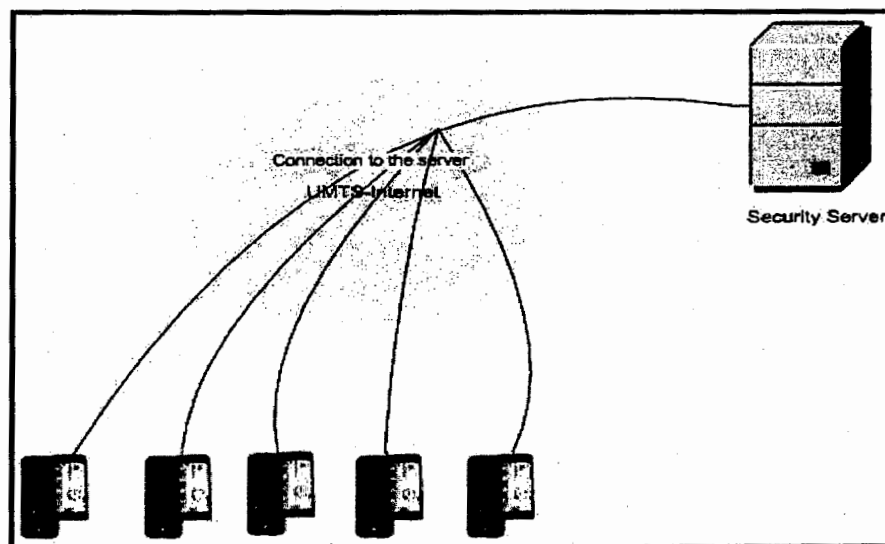


Figure 3.3: One server hosting many replicas

The normal execution of the phone or the results of execution are not affected by security server. All the security server is doing is to mirror the execution of the phone. The outgoing or incoming traffic to the phone is never affected by security server.

The connection lines between the security server and different phone represents association of each phone to its replica on the security server.

This model depicts the situation when the phone is up and running normally without being compromised. In this case any modifications made by the phone to the data are synchronized with the replica on the security server. When the phone is compromised in any one of ways we discussed, this approach needs to be changed in order to restore the phone to a stable position and at the same time avoid any external interference. We have decided to connect the phone to security server indirectly instead of a direct connection. We will use TCP/IP connection for communication with security server and indirect connection through the user machine just in case when we want to start recovery. We are going to discuss this in detail in the next section.

3.4 Architecture and Implementation

In this section we are going to discuss the architecture and implementation of our work. The basic view of our system is as shown in Figure 3.2. It works on the basic decoupling principle of Marvin Architecture when the execution is normal before the attack occurs.

The incoming and outgoing traffic is automatically synchronized with the replica from the proxy server in between the phone and the server side replica. It reduces the communication for synchronization of the phone with the replica. However the modification made locally must be synchronized. For this purpose the phone uses UMTS communication link.

But in the situation when an attack is detected by the security mechanism that is in place on the server side which may be DTA or any other technique the server sends an attack detection signal to the phone. Upon receiving the signal the user is prompted whether to restore the phone or not. If the user says "NO" he/she is advised to limit the usage of the phone or even to stop using the phone until it is restored to a stable position. If the user has access to his computer and an internet connection and says "YES" he/she is advised to connect his phone to his computer and run the restore utility (Described under section 3.3.2). This time

the communication is through a TCP/IP network between the user computer and the security server.

Running the restore utility will establish a communication link with the security server. When the connection is established a request for clean image will be issued to the server. The security server will redirect the request to the image directory and will retrieve an image of the running a replica for this phone. Since the directory contain clean images taken from the replica and stored for the purpose. One of these images will be used at this moment to bring the phone to a stable position. So when the server will retrieve the backup image from the image directory it will be sent out the requesting machine.

How the backup images will be taken we will discuss it in the coming sections. But here we will at least remind that there should be some protocol for taking a backup image. There must an agreement on when to take the image from the replica and store it in the image directory, for example when modification to the phone takes place or after some specific time period.

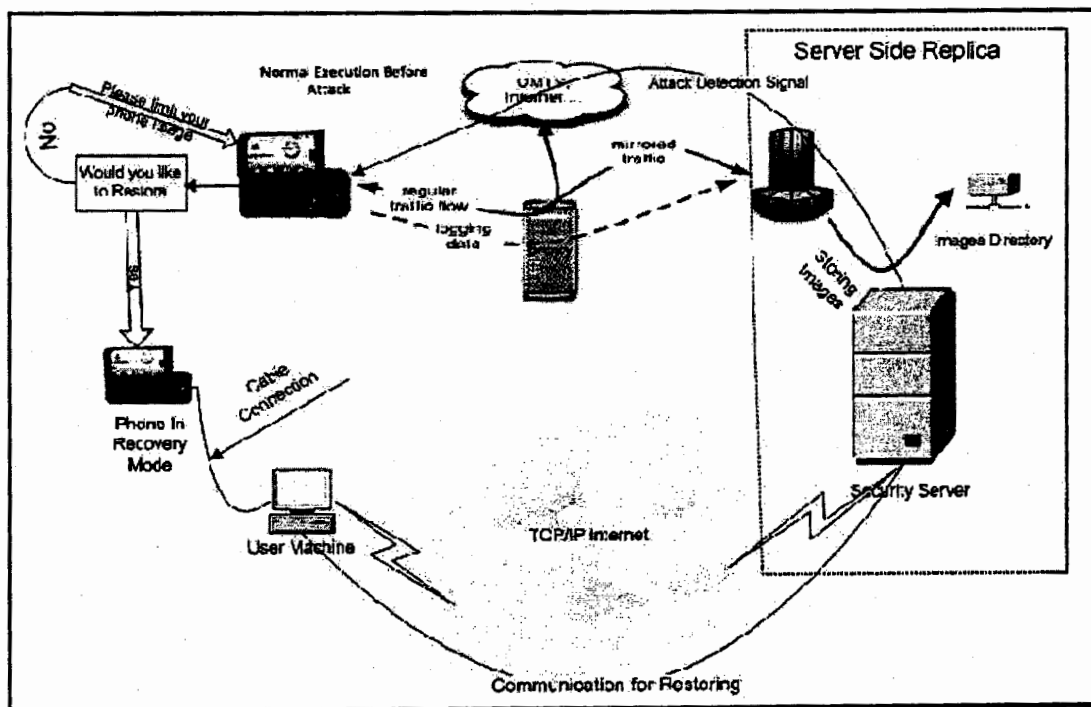


Figure 3.4: Architecture of the restore process for ANDROID

Here we are not going into that detail. It simply depends on the design decision whatever the designer decides that result in the best and smooth operation of the system. After receiving the clean images from the server, the phone will be restarted into the recovery mode and these images will be pushed into the SD card of the phone in order to be used for flashing the partition with these images in recovery mode. Since we have made appropriate modifications to the recovery image and with a little interaction from the user it will use the images in the SD card to flash them on to the partitions.

We have used HTC Google G1 Smartphone to test our works. To carry out the testing we need to have root access on the phone which is not enabled by default on the phone. So we

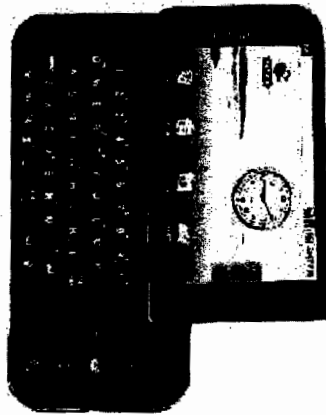


Figure 3.5: HTC Google Android G1

jailbroke (getting root access) the phone manually in order to have full access to system level changes and carry out our work smoothly without any interruption.

3.4.1 Why recovery Mode?

The **Recovery Console** is a feature of the ANDROID operating system. It provides the means for administrators to perform a limited range of tasks using a command line like interface. Its primary function is to enable administrators to recover from situations where operating system does not boot as far as presenting its graphical interface. Different recovery has different available functions or can be modified according to the requirements from the source code provided by Google. As such, the Recovery Console can be accessed by holding down Power+Home button at start up. We have modified the recovery image in order to implement our approach.

The focus our work is to enhance the security of the Smartphones specifically ANDROID. Our attempt is not only to secure the phones but also the process of restoration itself.

Restoring the phone while it is in recovery mode will ensure that no interface like Bluetooth, GPS, infrared or Wi-Fi is up, which in turns means that no attacker or intruder will be able to hack into the phone. In this way any kind of external interference can be stopped. This will ensure the authenticity of the restore process itself. This is not only true for hacking attacks but also for virus or spy software as they are not allowed to run in the limited functionality environment (recovery mode). If the phone is attacked, for example, we cannot even be sure that the modifications made during the restoration or even the restoration process itself is not compromised and clean. Any change or modification to personal data could be revealed to the attacker and the attacker may get access to any sensitive information during this process. It may include username password, personal data, contacts or credit card details.

To avoid these threats we need to follow a strategy that can ensure the maximum security of the restoration process itself so that we can be sure that the phone is restored clean and is stable. It must be secure enough to stop the attacker from getting any details or getting any information about the modifications that are being made to the phone for the purpose of restoration.

The recovery mode just makes sure the interfaces Bluetooth, wifi or GPS are not available for anyone. As we have already mentioned, some models of Nokia had suffered from such problems called *blue bugging*; caused by a bug in the Bluetooth implementation of those models [4]. Here is the most important question: if we stop all above mentioned ways will it be safe to restore the phone? If we consider the above discussion was about an attacker from outside, what if there is threat from inside, like virus, which may copy sensitive information and then later send it somewhere or even delete sensitive information or data from the phone or control the critical system functions.

We need to stop all these ways in order to stop an attacker from interfering. To achieve this goal we need to put the phone in a Mode where the above mentioned functions like wifi, GPS or Bluetooth do not work at all. Then in the same Mode, carry out the restoration process. We cannot stop an attacker once he gets the control of the device from enabling or disabling any feature. *This is the reason we have decided to carry out the restoration process while the phone is in recovery Mode, because in the recovery Mode none of the interfaces (Wi-Fi, Bluetooth, and GPS) work. Therefore there is no chance that the attacker can use them to hack the restoration process or even virus software is able to run.*

3.4.2 Our restore utility

When the phone is in recovery mode very limited functionality is available to the user. No interface such as Bluetooth, infrared, Wi-Fi, or the network operator work. In this mode we are unable to avail the full functionality. Now communicating with the security server through normal protocol to restore the phone seems impossible unless, we have a bridge/interface in the middle to connect the device with Security Server. For this purpose we needed a piece of code that will run on the user machine while the user machine and the device communication takes place via a USB cable using Android Debugging Bridge (adb).

This utility will perform two main tasks. Firstly, it is responsible for communicating with both the security server and the phone. For communicating with the security server it uses the TCP/IP connection available on the user machine and for communicating with the phone it uses the Android Debug Bridge [wl-12] ADB (to be discussed later). Secondly, it issues request for clean backup images from the security server, and when received, it pushes them to the phone's SD card using the cable connection. This piece of code will have to be run by the user from his/her personal computer after the phone has been connected to it in order to start recovery. On the server there should be ANDROID backup directory containing backup image taken over time depending on the protocol established. These images represent different modifications those are being made to the phone before taking these backup images. A latest backup clean image has to be sent by the server as a response to the request issued by this utility.

As we discussed, the clean image when downloaded at the user machine needs to be copied to a memory that the phone could use as it is not possible for the phone to use images from the user machine. But we have to remember if the phone is running in normal mode then copying images to phone's memory will not be safe in case the phone is attacked. So before copying the downloaded images to the phone internal memory we need to put the phone in recovery Mode in order to avoid any problems. As we know, in the recovery Mode the attacker will not be able to how the recovery mechanism is working.

After rebooting into recovery Mode we need to move the downloaded image into the SD card of the phone in order to be used for completing the restore process. Since the functionality is limited in recovery Mode we will not be able to access the phone's memory without making some modifications to the recovery image that is used in the phone. We need to modify the

recovery image which could replace the recovery image in the phone. How the recovery image is modified? We will discuss in the next section.

We have tried to make the process interaction less but there are situations where we need some interaction from the user to complete the process. The restore process cannot be started unless the user says yes and connects the phone to the personal machine. The user does not need to know about the internal details; such as where is the backup image located and how is it transferred to the phone? The process as it seems to the user is carried out between the user machine and the phone. The process is fully automated as our restore utility does not require any other interaction from user after the initial start and until the phone is restarted.

3.4.3 How to modify recovery image?

To carry out the restoration process as we discussed we needed to make some modifications to recovery image of the phone. To execute code/program in recovery mode, the recovery image needs to be modified accordingly, since the operating system is not operational in this mode and as we discussed it works in a limited functionality environment like a command line interface. For this purpose we used the ANDROID source by downloading the whole source code of the ANDROID project from the official website [wl-11]. Once you have the source, you can modify any of its components. From the source we then modified the source code for recovery for testing our work.

One of the basic modifications was to enable adb (*Android Debug Bridge*). Enabling “adb” in recovery Mode provides the flexibility to perform different operations on the phone by accessing the phone’s shell through the user machine to which the phone is connected via a USB cable. We are basically interested in two basic operations, pull and push. These operations are equivalent to copy-from and copy-to operations. We use these operations to copy the downloaded images to phone’s SD card memory to be able to use them for the restoration purpose from recovery console of the phone.

In addition to the above modifications we make some other modification to the recovery as well, it means enabling “adb” for push and pull operation is not quite enough to complete the task. Therefore we also modify recovery source in order to be able to execute (to call our function that can read the images for the SD card) our code while in recovery Mode. Our modified recovery code will read the images copied to the phone’s SD card memory to recover the damaged partitions by install the new stable image on them. After

installing/flashing the partition with these images all the changes that are made to the phone after taking this backup image will be rolled back and device will go into a previous state that is stable.

The recovery image is an “.img” image file which is almost impossible to modify directly because the contents of the image is not a normal text or readable pattern and is not easy to understand. To modify the recovery image we need to be able to read the contents of the image. It means before making any modifications to the recovery image we need to unpack the “.img” file in order to retrieve its contents, and then it will be possible to make modifications or replace some contents of the file. We used two Perl scripts to unpack, edit and repack these images available online [wl-13]. These scripts take source path of the input file and destination path for the out file to unpack or repack images. After executing the scripting by providing the right arguments we unpack the images and are able to see the required contents.

3.4.4 Android Debugging Bridge

The following definition is taken from the android developers’ website in [wl-12]. “Android Debug Bridge (adb) is a versatile tool lets you manage the state of an emulator instance or Android-powered device.” It is a client-server program that includes three components:

- A client, that runs on the development machine and can be invoked from the shell using adb command.
- A server, that runs on the development machine as a background process. It is responsible for managing the communication between the client and the adb daemon running on an emulator or device.
- A daemon, that runs on each emulator or device as a background process.

The “adb” support a large set of commands to be used while the device (is connected through USB) or the emulator is running on the user machine. Since we need to use the “adb” while the device is in recovery mode we either need a recovery image which already has “adb” enabled in it or we have to explicitly enable it. Enabling “adb” in recovery mode will give us access to a limited number of functions like “adb shell”, “adb push”, “adb pull”, “adb reboot”, “adb reboot recovery” etc. but to run these command we must have root access on the phone. For emulator we have the root access by default but not on the G1 phones.

Complete details on how to get root access on ANDROID HTC G1 are available in [wl-10] and many other resources on the web. After getting root access we can make changes to different parts of the source code.

One important aspect that we can flash any image on any partition with root access without modifying the recovery image by executing just one command i-e

```
flash_image <partition name> <source file on sdcard>
```

As the second argument suggests, we must first push the source file (image) to the sdcard using the “adb push” command. After this we can get into the phone shell using “adb shell” to execute the “flash_image” command. After flashing the image we can use “adb reboot” to boot into any specific mode to check if our command executed successfully or not. Since modifying recovery is time consuming job, so we did most of our testing with manual command to check the output quickly.

3.4.5 Algorithm for over all Restore process

1. *User machine sends a request for clean image*
2. *Request is received at server*
3. *In response the server searches for a latest clean system image*
4. *Sends the image to the requesting Node.*
5. *On receiving the image the download utility reboot the phone into recovery mode (using adb)*
6. *Pushes the image into SD-Card (using adb push)*
7. *User selects the restore option from recovery menu*
8. *Reboots on completion*

Chapter No-04

Results

Results

In this chapter we are going to add some results in the form of screenshots representing different states of the system during execution. Although the ABG will run a background process, still we will include some screen output results in order to show that our proposed solution is feasible. To check the integrity of the backup images we use them to flash the phone to check if it works fine or producing some error. While doing this one must be careful to take backup of the phone first, as a slight mistake during this process of testing images can break the phone and it may not be usable in the future. In such case the backup can be reinstalled on the phone to bring it to its original state.

Since it has been shown in [4] that the effect of Marvin Architecture based system on the phone's battery life is only 7%, the same result holds true for our approach since it does not carry out any execution directly on the phone. Here is the figure taken from [4] approving the statement.

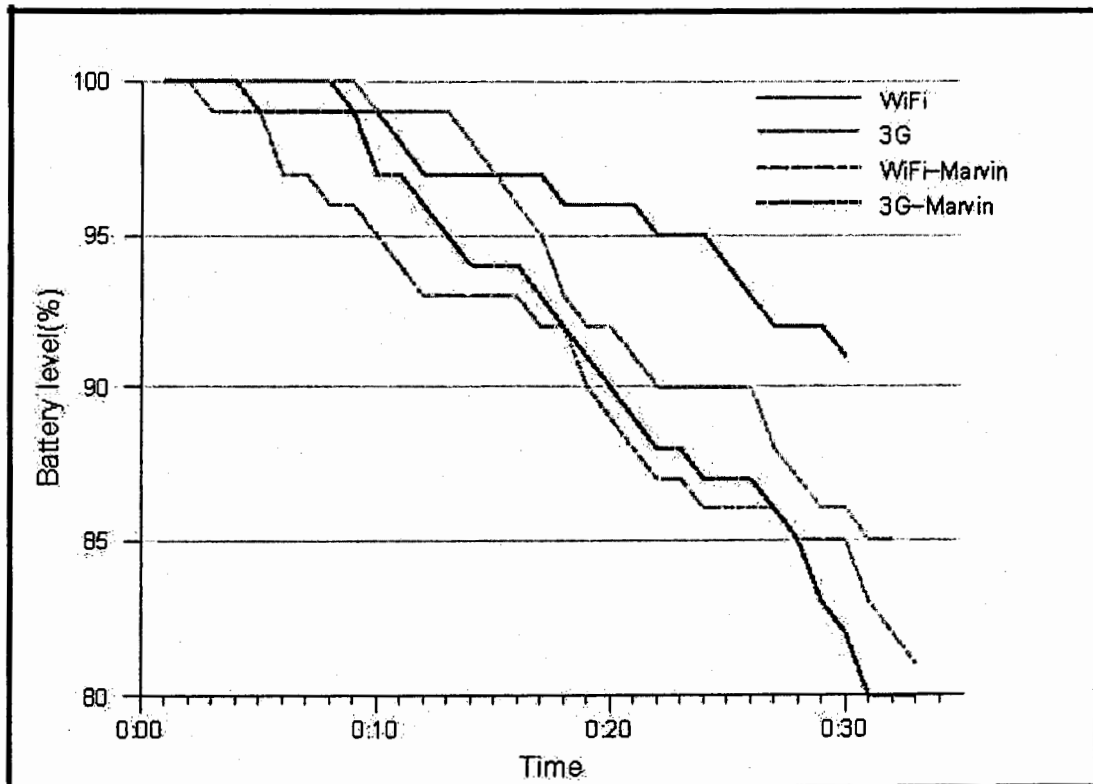
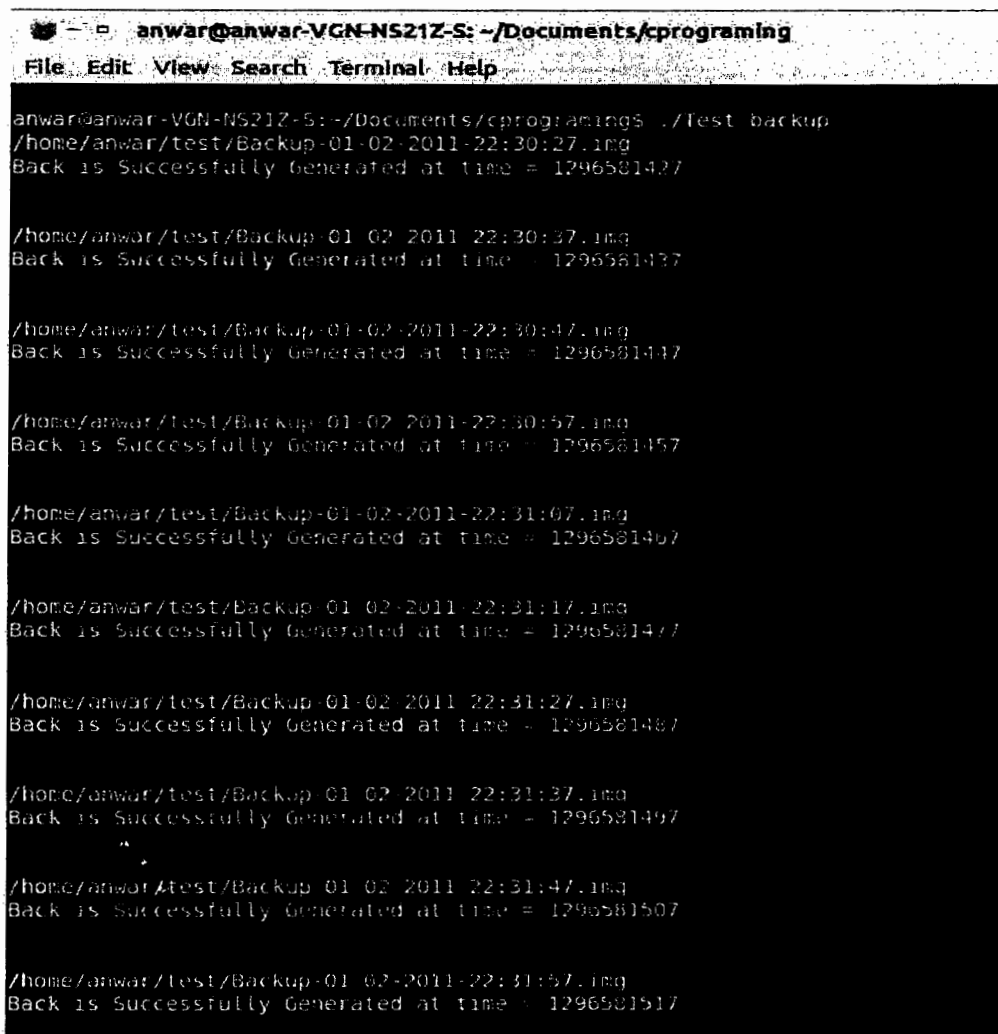


Figure 4.1: Battery Consumption

To check the feasibility of our proposed model we implemented the Automatic Backup Generator (ABG) to see if it works fine and produces the desired results. The following screenshot shows the execution of ABG which generates a backup after every 10 seconds, set on the timer. It shows the path of the generated backup with date and timestamp. It also outputs the generation time.

To check the integrity of the image we used the “hex” editor to open the both the source and destination images and compare the contents of both. Successful opening of the backup image indicates that the image is not corrupted during the backup process and the content comparison shows that the content of the image are intact.



```
anwar@anwar-VGN-NS21Z-S: ~/Documents/cprogramming
File Edit View Search Terminal Help

anwar@anwar-VGN-NS21Z-S: ~/Documents/cprogramming$ ./Test_backup
/home/anwar/test/Backup-01-02-2011-22:30:27.img
Back is Successfully Generated at time = 1296581437

/home/anwar/test/Backup-01-02-2011-22:30:37.img
Back is Successfully Generated at time = 1296581437

/home/anwar/test/Backup-01-02-2011-22:30:47.img
Back is Successfully Generated at time = 1296581437

/home/anwar/test/Backup-01-02-2011-22:30:57.img
Back is Successfully Generated at time = 1296581457

/home/anwar/test/Backup-01-02-2011-22:31:07.img
Back is Successfully Generated at time = 1296581467

/home/anwar/test/Backup-01-02-2011-22:31:17.img
Back is Successfully Generated at time = 1296581477

/home/anwar/test/Backup-01-02-2011-22:31:27.img
Back is Successfully Generated at time = 1296581487

/home/anwar/test/Backup-01-02-2011-22:31:37.img
Back is Successfully Generated at time = 1296581497

/home/anwar/test/Backup-01-02-2011-22:31:47.img
Back is Successfully Generated at time = 1296581507

/home/anwar/test/Backup-01-02-2011-22:31:57.img
Back is Successfully Generated at time = 1296581517
```

Figure 4.2: Output of ABG

The following screenshot verifies the execution shown in the previous screenshot. This is an image of the test directory with the backup images generated by the execution of ABG. The name backup is followed by a date and time stamp, and shows that the backup is generated on the specific date at the specified time. This timestamp will make the process of searching for latest image much easier when required for recovery.

The source code for Automatic Backup Generator (ABG) is given in Appendix-A with the name of "ABG.c".

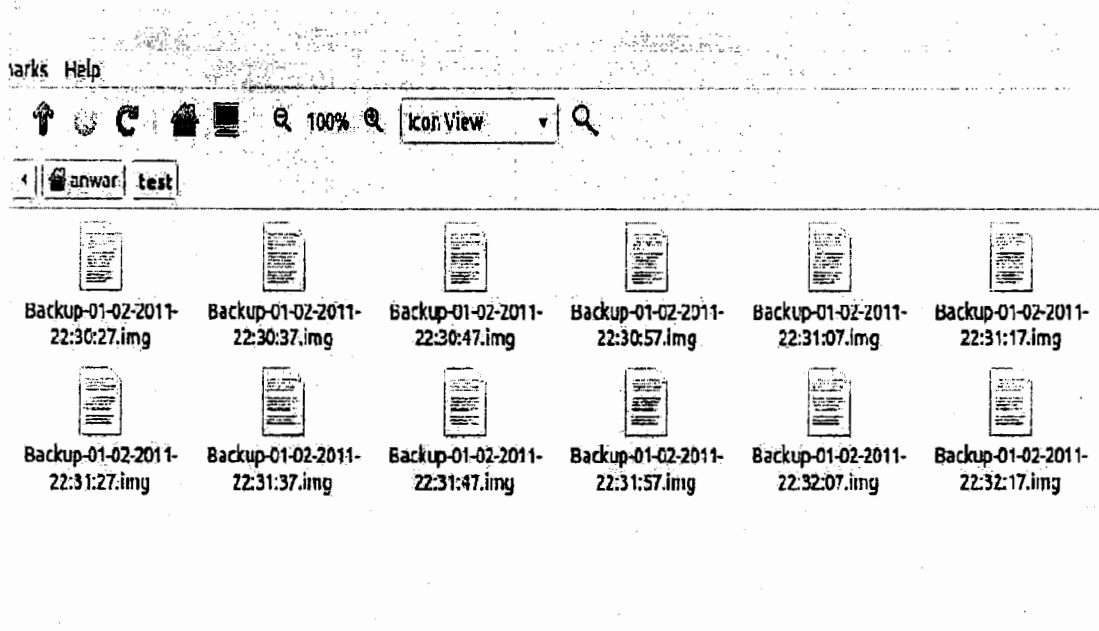


Figure 4.3: Test Directory with generated backup image

We will not go into details of the Marvin Architecture here as we have discussed it in detail in chapter 2. Now we are going to give some screen shots from testing our work on the system. We have used HTC G1 for this purpose.

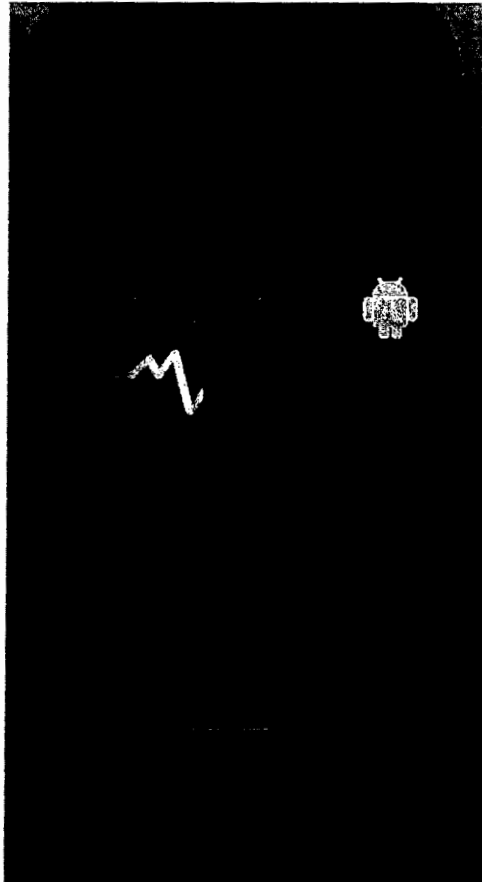


Figure 4.4: HTC G1

Recovery console offer a minimum function environment to the user to perform functions like restore update etc. Also user can apply any update in zip format available on the sdcard of the phone. It must be remembered that some of these functions can be performed using the fastboot utility available and can be built from android source code. It can execute different command like flashing image, rebooting the device in different mode etc. All we had to do is

to add a menu item to the menu list and execute our code against that menu item. The same can be done against an existing menu item.

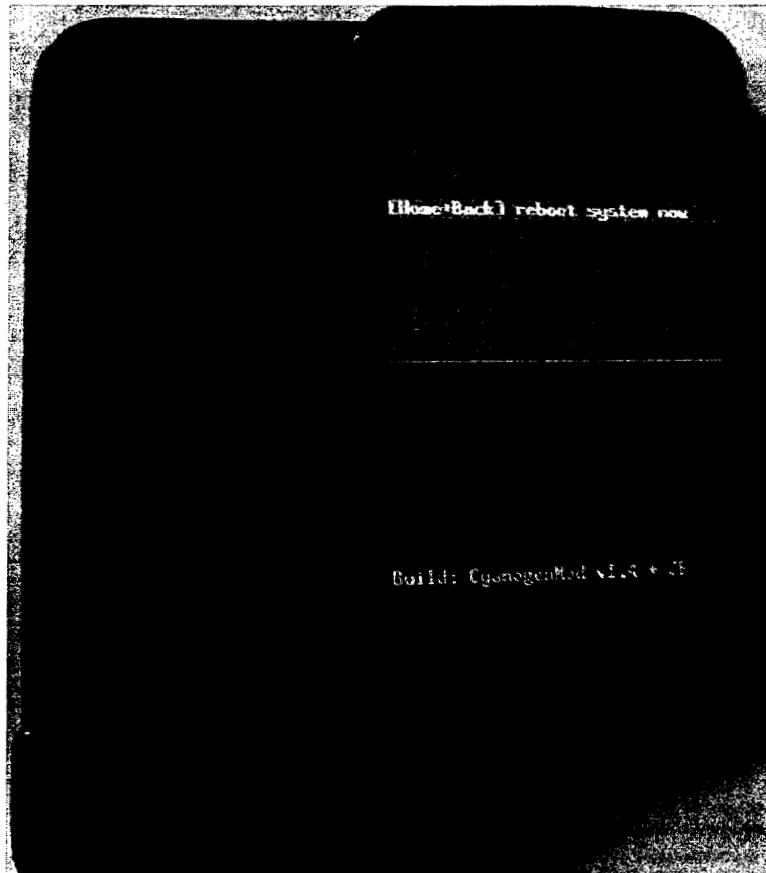


Figure 4.5: Menu options in Recovery Mode

This result shows a successful installation of a modified recovery image on the phone. It means we can execute any code in recovery mode by modifying the recovery image accordingly. This verifies that it is possible to carry out any function in this mode but within the restriction limits imposed.

Since ANDROID is an open source operating system, if we look on the web many developers offering their modified recovery images with different functionalities and feature to the ANDROID users. This is becoming a common practice in the world of ANDROID to have a Smartphone with custom features you like.

In the case of some run-time errors in the code modification of the recovery image when it is flashed installed on the device, it gets stuck at the following screen and will remain on this screen forever. This verifies that the process of modification is incorrect.

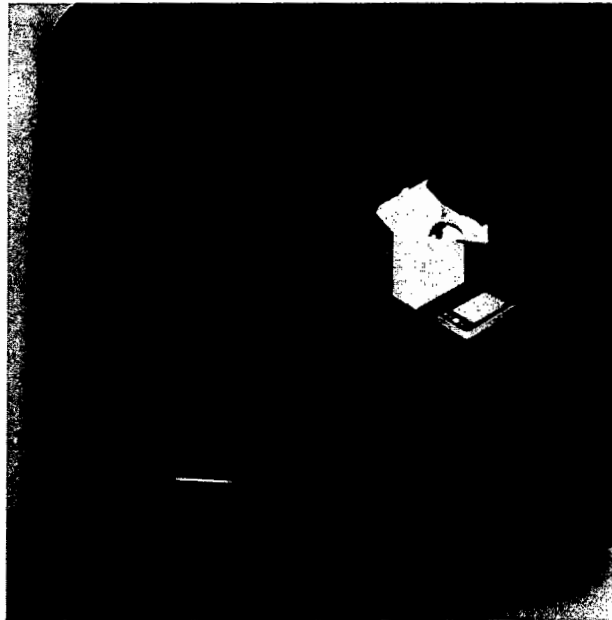


Figure 4.6: Error in Recovery modifications

We have successfully implemented and tested Automatic Backup Generator. The aim was to test whether a system image can be stored as backup and used for restoring the phone at some times later when required.

We have included some other source code for different scripts those are being used in testing the feasibility of this process, and the code for modified recovery in different Appendices.

4.1 Comparison with Marvin Architecture

We are going to compare Marvin architecture with our approach to show how our approach overcomes different weaknesses of the architecture. We do not challenge the results in produced by Marvin Architecture in terms of avoiding resource limitation problem as well as attack detection. It means the results stand the same for our approach except that our approach incorporates the recovery facility to restore the phone to an earlier stable position,

whereas in Marvin Architecture users only have the option to restore the phone to factory setting. In that case all the data, setting and preferences will be lost. The comparison is shown in the following section.

4.1.1 Our approach versus Marvin Architecture

- ❖ Our approach guarantees a transparent and automatic backup generation process where user need not be aware of where the backup is located and how is it generated? While in Marvin Architecture there is no provisions for backing up the system to be used for restore.
- ❖ In our approach the maintenance is done by the security server and the user does not need to worry about the maintenance and security of the backup.
- ❖ In our approach the phone is restored in recovery mode in order to ensure the security of the restore process itself.
- ❖ In our approach restore is through the user machine connected to internet to avoid any battery power consumption and also to ensure the security of the process itself. Where as using the phone UMTS connection for restore is not feasible due to the reason that with this approach it can significantly affect the battery life as well as the process itself become compromised if the phone is already under attack.
- ❖ Incorporates all strengths of Marvin like avoiding resource limitation problem and improving security with some of its own features, like restore and avoiding external interference during the restore process.

From the above comparison it is clear that using Marvin architecture can avoid resource limitation problem as well as improving security but at the time has no provision for protecting user data, applications, setting and preferences. On the other hand our approach proved to be as secure as Marvin, with a transparent and automatic recovery of the phone to recent stable point there by protecting the user from data loss. Our approach looks more safe as the system image is taken from the replica on the security sever instead backing it up from the phone directly.

The comparison is summarized in the following table:

| Marvin Architecture | VS | Our Approach |
|--|----|---|
| One way synchronization from Phone to Server | | Two way communication i.e from phone to server and from server to phone. |
| No mechanism for restoring the phone after an attack | | Can be used to restore the phone with no power consumption from the battery. |
| All the communication is through UMTS internet which is not suitable for restore because battery power will be consumed very fast. | | Restore is through the user machine connected to internet to avoid any battery power consumption. Also to ensures the security of the process itself. |
| Has the advantage of avoiding resource limitation problem and improving security | | Incorporates all strengths of Marvin with some of its own features, like restore. |
| | | No External Interference from attacker, nor a malicious software |

Table 4.1: Comparison proposed approach with Marvin Architecture

Chapter No-05

Summary and Conclusions

Summary and Conclusion

4.1 Summary

This work demonstrates the crucial role of recovery/restore of ANDROID phones from a replica on security server and gives an approach that is automatic and transparent to restore the phone from a compromised state to a stable one. Our work is a valuable extension to the *Marvin Architecture* in order to accommodate a restore facility. This work represents a very flexible and reliable approach for recovery/restore which insures security, reliability and availability. The popularity and application complexity of mobile devices (Smartphone) is increasing as the vendors are striving to provide maximum functionality like online shopping, web browsing, checking emails, video conferencing storing and editing documents and photos etc. and this is the reason they are becoming the first and attractive targets for different attacks. These attacks are also a source of financial incentives for the attackers because of the broad application domain enabling these devices to be used for different financial transactions. Therefore they need a highly advance security system to insure that these operations are secure to be performed on these devices. But the implementation of such advance system is restricted by the resource limitations like better life, processing power etc associated with these devices.

Our approach is based *Marvin architecture* which decouples the security function from the phone to an isolated powerful machine with sophisticated security mechanisms called the security server. This architecture is responsible to avoid the resource limitation problem and any other restriction that is a hurdle in the way of implementing an advance security technique for Smartphones. Our work is an extension to this architecture to provide the restore facility to the users.

We aimed to implement a system that not only provides tight security and advance detection system but at the same time do not increase the processing burden on the phone so as to decrease the load on the battery power. The communication problem between the device and the replica for synchronization is solved to some extent by Marvin architecture by synchronizing the in coming and out going traffic directly with the replica from a proxy server between the device and the security server. To sum up main objectives of this work are:

- Sophisticated security system
- Heavy security against intrusion/attack detection
- Decreased processing load on phone

Automatic backup generation system is a system which generates backups of the replicas on security server after a specified period of time. It is totally a design decision and depends on whatever protocol is used. Whenever there is a problem, the system is able to rollback to an older state using a clean backup image of the system just like Microsoft Windows “checkpoint” technique. With this rollback feature we are not only able to restore the device to a stable position but also it help to clean the system from any anomalies occurring after the backup has been taken.

To insure that the backup image to be used for restore is clean the mechanism of *Taint analysis* could be used. This technique uses back tracking in order to find out which data has been affected by an attack and to what extent.

4.2 Future work

Much more needs to be done. This thesis gives the design possibility to one such system. Currently we have tested our approach with only one replica and one device but in future it can be extended to support multiple replicas on the same server machine serving my devices at the same time.

Moreover this approach can also be extended to a system that can identify the user with some authentication mechanism and restore customized user’s data instead of the whole system. In this way the un-necessary communication between the server and the phone can be minimized to a significant level. The user will able to restore the desired application, setting or preferences.

This technique can cope with situation to restore data even if the phone is broken or even stolen. The user needs to have some special means by which he/she will be able to block the stolen phone. One such example is the use of IMEI number of the stolen phone, and restore the data to his/her new phone with different IMEI number. The same is true for broken phone except this one will not need to be blocked. Although this can also be accomplish with Google account authentication associated with each ANDROID Smartphone.

This technique is not limited to ANDROID only. It can be implemented for other phones like Smartphones from Nokia, Sony Ericson, and Motorola etc to address the same resource limitation issue.

Appendices

Appendix-A

Timer for Automatic Backup Generator

Here we are going to give the source code for the timer which trigger the automatic backup generation of the replica. This code will run completely as background process with no interaction from the user. It will generate copies of the three main partitions “/boot”, “/system”, and “/userdata” with a date and time tag. This tag can later be used to search for the latest clean image. Then it will be downloaded to the user machine and flashed on to the device.

Here is the code for ABG:

```
/* this ABG.c code will run on the security server where a
replica (emulator) replicating the execution of the device
(android phone) is running. The function of this code is
generator backup of the system from the replica and store it
in a directory specified for backup images to be stored in.
These images may later be used for the phone's restoration
process if the phone is attacked or corrupted due to system
error to rollback it to a previous stable point.
```

```
*/
```

```
#include <sys/time.h>
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

```
long int count=0; /*a global variable holding the time of
The next backup generation */
```

```
int delay=1*10; /* delay is a global variable represents
The time between two back up */
```

```
int main()
{
```

```
    struct timeval tv;
    count=SetTimer(tv,delay); //set up a delay timer
```

```
    while(1)
        CheckTimer(count,delay); //Generate bakups periodically

return 0;
}
/* This method will set the timer for the first time to
initiate the backup process. It will be called once the
program start.*/

int SetTimer(struct timeval tv, time_t sec)
{
    gettimeofday(&tv,NULL);
    tv.tv_sec+=sec;
    return tv.tv_sec;
}

int CheckTimer(long int x,time_t sec) // Checks the current
time against the set timer
{
    struct timeval ctv;
    gettimeofday(&ctv,NULL);

    if (ctv.tv_sec == x)
    {
        generate_backup(); //call for Backup
        printf("Back is Successfully Generated at time = %ld
\n",x);
        gettimeofday(&ctv,NULL);
        count=ctv.tv_sec+sec;
        //sleep(10);
    }

return 0;
}

/* This function will open the images directory and read
images to a backup directory to be used for restore later.*/

int generate_backup()
{
    Char file_path[100],file_name[30],time_buffer[20];
    Char file_extension[5];

    struct timeval tv;
```

```

time_t curtime;
gettimeofday(&tv, NULL);
curtime=tv.tv_sec;
char *temp1,*temp2,*dest_path;

strcpy(file_path, "/home/anwar/test/");
strcpy(file_name, "Backup-");
strftime(time_buffer, 30, "%d-%m-%Y-
%T", localtime(&curtime));
strcpy(file_extension, ".img");

temp1=strcat(file_path, file_name);
temp2=strcat(time_buffer, file_extension);
dest_path=strcat(temp1, temp2);

char *source_path="/home/anwar/Desktop/newimage.img";
FILE *rfile;
FILE *wfile;
char *buffer;
unsigned long fileLen;

//Open the source file
rfile = fopen(source_path, "rb");
if (!rfile)
{
    fprintf(stderr, "Unable to open file %s",
source_path);
    return;
}

//Get file length of source file
fseek(rfile, 0, SEEK_END);
fileLen=ftell(rfile);
fseek(rfile, 0, SEEK_SET);

//Allocate memory
buffer=(char *)malloc(fileLen+1);
if (!buffer)
{
    fprintf(stderr, "Memory error!");
    fclose(rfile);
    return;
}

//Read source file contents into buffer
fread(buffer, fileLen, 1, rfile);
fclose(rfile);

//Open destination file
wfile = fopen(dest_path, "wb");
if (!wfile)

```

```
{
    fprintf(stderr, "Unable to open destination %s",
dest_path);
    return;
}

//write file from buffer to destination
fwrite(buffer, fileLen, 1, wfile);
fclose(wfile);

free(buffer);
return 0;
}
```

Appendices

Appendix-B

Unpacking & Repacking Recovery images

Unpacking recovery image

To unpack recovery image and avoid the manual efforts and error chances we have used the following perl script. This script unpacks the recovery image into two different files. The first one is "kernel" which is a ".gz" format. The second one is "ramdisk" which is also of the same format but further extracted into a directory with the same name. We can change any of the contents in this directory to reflect our changes in recovery mode.

This script takes two arguments. One is the recovery image to be unpacked and the second argument is the path to the output files.

```
#!/usr/bin/perl -W

use strict;
use bytes;
use File::Path;

die "did not specify boot img file\n" unless $ARGV[0];

my $bootimgfile = $ARGV[0];

my $slurpvar = $/;
undef $/;
open (BOOTIMGFILE, "$bootimgfile") or die "could not open boot img
file: $bootimgfile\n";
my $bootimg = <BOOTIMGFILE>;
close BOOTIMGFILE;
$/ = $slurpvar;

# chop off the header
$bootimg = substr($bootimg,2048);

# we'll check how many ramdisks are embedded in this image
my $numfiles = 0;

# we look for the hex 00 00 00 00 1F 8B because we expect some trailing
padding zeroes from the kernel or previous ramdisk, followed by 1F 8B
(the gzip magic number)
while ($bootimg =~ m/\x00\x00\x00\x00\x1F\x8B/g) {
    $numfiles++;
}

if ($numfiles == 0) {
    die "Could not find any embedded ramdisk images. Are you sure
this is a full boot image?\n";
} elsif ($numfiles > 1) {
```

```

    die "Found a secondary file after the ramdisk image. According
to the spec (mkbootimg.h) this file can exist, but this script is not
designed to deal with this scenario.\n";
}

```

```
$bootimg =~ /\.(*\x00\x00\x00\x00)(\x1F\x8B.*)/s;
```

```
my $kernel = $1;
my $ramdisk = $2;
```

```
open (KERNELFILE, ">$ARGV[0]-kernel.gz");
print KERNELFILE $kernel or die;
close KERNELFILE;
```

```
open (RAMDISKFILE, ">$ARGV[0]-ramdisk.cpio.gz");
print RAMDISKFILE $ramdisk or die;
close RAMDISKFILE;
```

```
print "\nkernel written to $ARGV[0]-kernel.gz\nramdisk written to
$ARGV[0]-ramdisk.cpio.gz\n";
if (-e "$ARGV[0]-ramdisk") {
    rmtree "$ARGV[0]-ramdisk";
    print "\nremoved old directory $ARGV[0]-ramdisk\n";
}

```

```
mkdir "$ARGV[0]-ramdisk" or die;
chdir "$ARGV[0]-ramdisk" or die;
system ("gunzip -c ../$ARGV[0]-ramdisk.cpio.gz | cpio -i");
```

```
print "\nexttracted ramdisk contents to directory $ARGV[0]-ramdisk/\n";
```

```
”
```

Repacking the image

After modification we can repack the image of course through manual process but again to avoid the efforts as well as error chances we have use another perl script “repack-bootimg”. This script takes three arguments. First is the kernel file in “.gz” format the second one is the “ramdisk” directory to which modification has been made and the third one is the name path to the output image.

```
“
```

```
#!/usr/bin/perl -W
```

```
use strict;
use Cwd;
```

```
my $dir = getcwd;
```

```
my $usage = "repack-bootimg.pl <kernel> <ramdisk-directory>
<outfile>\n";

die $usage unless $ARGV[0] && $ARGV[1] && $ARGV[2];

chdir $ARGV[1] or die "$ARGV[1] $!";

system ("find . | cpio -o -H newc | gzip > $dir/ramdisk-
repack.cpio.gz");

chdir $dir or die "$ARGV[1] $!";

system ("mkbootimg --cmdline 'no_console_suspend=1 console=null' --
kernel $ARGV[0] --ramdisk ramdisk-repack.cpio.gz -o $ARGV[2]");

unlink("ramdisk-repack.cpio.gz") or die $!;

print "\nrepacked boot image written at $ARGV[1]-repack.img\n";

"
```

Appendices

Appendix-C

Modified Recovery

Here we are including the source code for modified recovery. We have modified three files from android source namely recovery.c, recovery_ui.c, default_recovery_ui.c. These files can be found at the following path in the ANDROID source.

Project-Source/bootable/recovery/

Although we have used the manual testing to flash the recovery partition mostly as this method is quick for testing. However we did test the source code modification to see if we can modify the code and execute some thing we want to in the recovery mode successfully.

We can add any number of menu items to the recovery menu and can perform any function on click on the specified menu item. The user either has to click on the menu item with the help of the track ball or press the shortcut key for that item.

Bibliography

References

- [1]. Paul Anderson, Adam Blackwood, "Mobile and PDA technologies and their future use in education", JISC Technology and Standards Watch, page 3, 11, 13, / November 2004.
- [2]. Aubrey-Derrick Schmidt, Hans-Gunther Schmidt, Jan Clausen, Ahmet Camtepe, Sahin Albayrak, Kamer Ali Yüksel and Osman Kiraz. "Enhancing Security of Linux-based Android Devices", TU Berlen DAI-Labour Germany, page 5, / September 19, 2008
- [3]. Russell Beale "Supporting Social Interaction with Smart Phones," IEEE Educational Activities Department Piscataway, NJ, USA, Volume 4, page 5 / April, 2005.
- [4]. G Portokalidis, P Homburg, N FitzRoy-Dale, K Anagnostakis, Herbert Bos, "Protecting Smartphones by means of execution replication". Technical report IR-CS-54, (Vrije Universiteit Amsterdam), Page 1, 2, 4, 5, 6 / September, 2009.
- [5]. William Enck, Machigar Ongtang, Patric McDaniel, "On lightweight mobile phone application certification". In proceedings of the 16th ACM conference on Computer and Communications Security (CCS'09), Chicago, Illinois, USA, pp. 235-245. /2009
- [6]. A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, "Google Android: A State-of-the-Art Review of Security Mechanisms", Department of information and system Engineering Ben Gurion University Israel. pp. 17- 21 / December, 2009,
- [7]. Aubrey-Derrick Schmidt, Hans-Gunther Schmidt, Leonid Batyuk, Jan Hendrik Clausen, Seyit Ahmet Camtepe, Sahin Albayrak, Can Yildizli, "Smartphone Malware Evolution Revisited: Android Next Target?", 4th IEEE International Conference on Malicious and Unwanted Software (Malware 2009), Montreal, Quebec, Canada, page 1,2,3,5
- [8]. Wayne Jansen, Karen Scarfone. "Guide lines on cell phone and PDA security". Recommendations of the national institute of standard and technology, National Institute of Science and Technology (NIST) US Department of Commerce, Special Publication 800 -124, / Oct 2008

- [9]. A.-D. Schmidt, F. Peters, F. Lamour, and S. Albayrak, "Monitoring smartphones for anomaly detection," in MOBILWARE 2008, International Conference on MOBILE Wireless MiddleWARE, Operating Systems, and Applications, Innsbruck, Austria, 2008.
- [10]. A. Bose, X. Hu, K. G. Shin, and T. Park, "Behavioral detection of malware on mobile handsets," in Proceeding of the 6th international conference on Mobile systems, applications, and services. Breckenridge, CO, USA: ACM, 2008, pp. 225–238.
- [11]. D. C. Nash, T. L. Martin, D. S. Ha, and M. S. Hsiao, "Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices," in PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops. Washington, DC, USA: IEEE Computer Society, 2005, pp. 141–145.
- [12]. H. Kim, J. Smith, and K. G. Shin, "Detecting energy-greedy anomalies and mobile malware variants," in MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services. New York, NY, USA: ACM, 2008, page 239–252.
- [13]. G. Jacoby and N. Davis, "Battery-based intrusion detection," in Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE, vol. 4, 2004, page 2250–2255.
- [14]. T. K. Buennemeyer, T. M. Nelson, L. M. Clagett, J. P. Dunning, R. C. Marchany, and J. G. Tront, "Mobile device profiling and intrusion detection using smart batteries," in HICSS '08: Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences. Washington, DC, USA: IEEE Computer Society, 2008, p. 296.
- [15]. M. Miettinen, P. Halonen, and K. Hatanen, "Host-Based Intrusion Detection for Advanced Mobile Devices," in AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06). Washington, DC, USA: IEEE Computer Society, 2006, pp. 72–76.

- [16]. Machigar Ongtang, Stephen McLaughlin, William Enck, Patrick McDaniel, "Semantically Rich Application-Centric Security in Android", Annual Computer Security Applications Conference, Honolulu, Hawaii, / December 2009.
- [17]. J. Newsome and D. Song. "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software", In proceeding of the Network and Distributed system security symposium (NDSS 2005), /2005.
- [18]. Junyao Zhang, Article on "Android vs iPhone", April 12, 2010.
- [19]. *Michael K. Cheng*, "iphone jailbreaking under the DMCA: towards a functionalist approach in anti-circumvention", 2010.

Web Links

- [wl-1]. Smartphone. [Online] Available at <http://en.wikipedia.org/wiki/Smartphone>
- [wl-2]. What is ANDROID? [Online] Available at official ANDROID developers website: <http://developer.android.com/guide/basics/what-is-android.html>
- [wl-3]. H. Moore. Cracking the iphone (part 1). [Online] Available at <http://blog.metasploit.com/2007/10/cracking-iphone-part-1.html> [October 2007].
- [wl-4]. Karl Flinders. iPhone 3GS crack released. [Online] Available at: <http://www.computerweekly.com/Articles/2009/07/06/236777/iPhone-3GS-crack-released.html> [Monday, July 06, 2009]
- [wl-5]. Computer Business Review, 2004. PDA security: mobile employees risk corporate data. **Computer Business Review Online** [online]. Available at: http://www.cbronline.com/features/pda_security_mobile_employees_risk_corporate_data. [31 August 2004]
- [wl-6]. INSECURE.ORG, "Top 100 network security tools," 2006. [Online] Available at: <http://sectools.org/>
- [wl-7]. iTunes. [online] Available at <http://en.wikipedia.org/wiki/iTunes>

[wl-8]. Sam Costello. How to restore iPhone from backup. [online] Available at

<http://ipod.about.com/od/iphonetroubleshooting/ss/restore-iphone.htm>

[wl-9]. System Restore. [Online] Available at http://en.wikipedia.org/wiki/System_Restore

[wl-10]. Why should you root your Dream G1? [Online] available at: <http://forum.xda-developers.com/showthread.php?t=442480>

[wl-11]. Building android from source. [Online] available at:

<http://source.android.com/download>

[wl-12]. Android Debug Bridge(ADB). [Online] available at:

<http://developer.android.com/guide/developing/tools/adb.html>

[wl-13]. How to: Unpack, Edit, and Re-pack Boot Images. Available at [http://android-dls.com/wiki/index.php?title=HOWTO: Unpack%2C Edit%2C and Re-Pack Boot Images](http://android-dls.com/wiki/index.php?title=HOWTO:_Unpack%2C_Edit%2C_and_Re-Pack_Boot_Images)

