UNIVERSITY OF AGDER

SECURITY ON THE MIS AND THE INTERFACING MODEM

EMMANUEL MENSAH AZADAGLI

Supervisors
ASSOCIATE PROFESSOR GEIR MYRDAHL KØIEN
SENIOR SOFTWARE DESIGNER MAGNUS REFTEL

This Master's Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

University of Agder, 2012
Faculty of Engineering and Science
Department of ICT

Abstract

The Modem Interface Services is a software system that provides bridge solution for the modem in Android mobile platform developed by ST-Ericsson. The functions of the Modem Interface Service are implemented using different software components.

The goal of this master thesis was to perform security analysis on the Android mobile platform developed by ST-Ericsson. The main objectives were to detect security vulnerabilities in the core component that integrates the modem into Android platform.

In performing this analysis, three security analysis methodologies were employed. TVRA methodology was used to identify architectural vulnerabilities whiles static code analysis was applied to find vulnerabilities due to implementation errors. In order to show different means of exploiting the weaknesses in the component, attack trees methodology was employed to discover vulnerabilities due to design errors as well as implementation flaws.

The use of these methods made it possible to identify vulnerabilities in the target component and its environment. Based on the results obtained, possible mitigations strategies were proposed.

Preface

This master thesis is submitted in partial fulfilment of the requirements for the degree Master of Science in Information and Communication Technology at the University of Agder, Faculty of Engineering and Science in Grimstad, Norway.

The thesis is written for ST-Ericsson as information on security in the MIS and the interfacing modem. The research work has been very challenging involving a great deal of data gathering, analysis and interpretation. The research was carried out under the supervision of Associate Professor Geir Myrdahl Køien at University of Agder, Norway and a senior software designer, Magnus Reftel from ST-Ericsson at Grimstad, Norway.

To begin with, I want to extend my profound gratitude to my supervisor Associate Professor Geir Myrdahl Køien for his immense assistance, patience and leadership. It would not have been possible to do this research without his ideas and counsel.

In the course of the project, Magnus Reftel from ST-Ericsson has been very supportive in giving technical advice and also helped with revision of the report. I will like to thank him for the role he played in making this research work a success.

Additional thanks goes to Stian Haugen, technical manager at ST-Ericsson for contributing to the success of this research work.

Grimstad

01 June 2012

Azadagli Emmanuel Mensah

Contents

List of Figures

List of Tables

1 Introduction

Mobile devices technology has evolved rapidly in recent years leading to increased resources, high connectivity and usable interfaces. This development also results in the rise in application development and distribution for mobile devices. With the advent of Android open platform, every mobile platform developer uses different software design to integrate the modem which provides platform services [6, 27]. In line with this, ST-Ericsson also developed Modem Interface Services (MIS).

The MIS is a software system developed to provide bridge solution for the modem to be integrated into a Linux Host system. The MIS consists of many components. Each component has a specific function in the platform and contributes to over all purpose of the MIS. This thesis is focusing on security analysis of the MIS and the interfacing modem.

1.1 **Background Information**

The MIS is designed to integrate the modem into Android platform. There are many components that constitute the MIS. The Radio Interface Layer (RIL) component in the MIS is the main component integrating the modem in Android [27]. Its function is to enable the modem supported services. It uses AT command to invoke and control the modem services. Figure 1 presents architectural view of the telephony and the RIL interfaces in Android. As seen in the figure, the RIL interface in Android is divided into three modules. The first module is a generic RIL and it forms part of the Android application framework. The RIL daemon is the second unit and the vendor RIL is the third component. The RIL daemon receives all RIL related requests and forwards them to the vendor RIL. The vendor RIL in the architecture is modem specific and its implementation depends on the modem. The RIL component in MIS correspond to vendor RIL module in figure 1 and it is the focus of this analysis. The RIL is responsible for receiving all the requests from Android telephony framework and maps

the requests to the right AT commands. The commands are then forwarded to the modem. Similarly, AT commands received from the modem are changed to corresponding replies for Android telephony framework to understand [27].

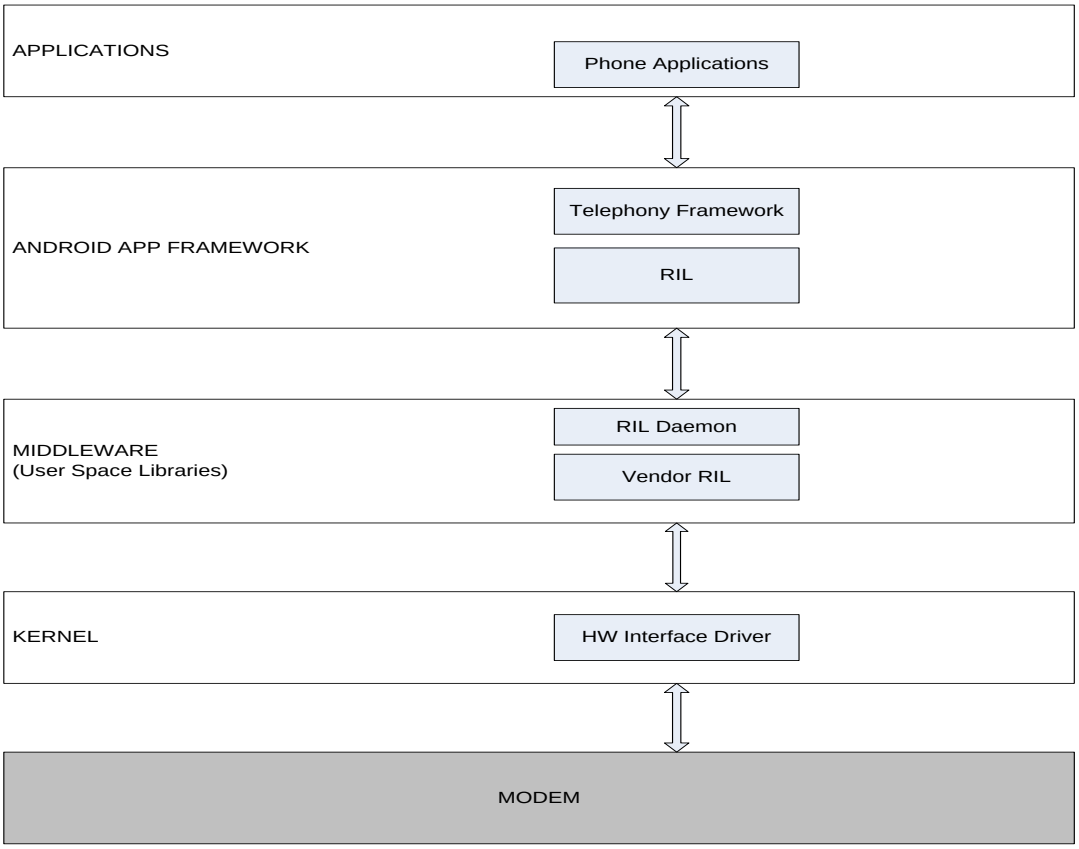

Figure 1: Architectural view of telephony framework and RIL interfaces in Android

This figure is drawn based on the one in [27]

The RIL is designed to support all the mandatory APIs in accordance with the Host operating system telephony API specification [6].

In Android, the operating system manages access to system resources. However, if this is not managed precisely, it will affect the user or the resources on the mobile device. The restriction on access to resources can be realized using different means. Android uses permission –based security mechanisms to protect sensitive APIs. The protected APIs in Android comprises:

- Camera
- Location data
- Bluetooth functions
- Telephony functions
- SMS/MMS functions
- Network/ data connections

These APIs can only be accessed through the operating system. To use any protected resource on the mobile device, an application needs to define its capabilities in the manifest file. During installation of an application, the system shows the permissions asked for to the user and the user must approve them to continue with the installation [5, 24].

1.2 **Problem and Research Questions**

Android is a Linux-based mobile platform which is programmed using Java. The platform has its own security mechanisms designed for a mobile environment [5]. The unique security model of Android concentrates on putting the end user in control of the handset. Android platform makes it possible for third parties to develop applications that use mobile phone features and resources.

Google has outlined a permission-based security model in Android platform [4]. Every application is required to make permission request in order to access stored data and the features of the phone. The permissions basically specify the capability a user may grant to an Android application. In other words, a user is allowed to assess the capability of the application and decide whether or not to install the application on his device. It is therefore important that the permission-based model is properly enforced in Android smartphones owing to the crucial role it plays in running Android applications.

However, Android devices come from different manufacturers. The open nature of the platform permits proprietary extensions and changes. These extensions and changes can improve or degrade the security mechanism of the platform. Therefore, analysis of a distribution of Android is an essential step in protecting information on that system [4, 5]

The purpose of this master thesis is to analyze a variant of Android mobile platform developed by ST-Ericsson. The focus of the research is on security on the Modem Interface Services (MIS) and the interfacing Modem. The main objective is to determine weaknesses and threats (vulnerabilities) in the main Mobile Interface Services (MIS) component integrating the modem.

This study covers mismatches between android and MIS with regard to the possibility of third party applications gaining unauthorized access to resources.

Whenever appropriate, countermeasures will be proposed to mitigate the threats to the vulnerabilities that exist in the MIS component.

Figure 2 shows an overview of the platform with a focus on the MIS, which consist of several supporting blocks for applications execution. The application systems are generally known as the Host and MIS is part of it.

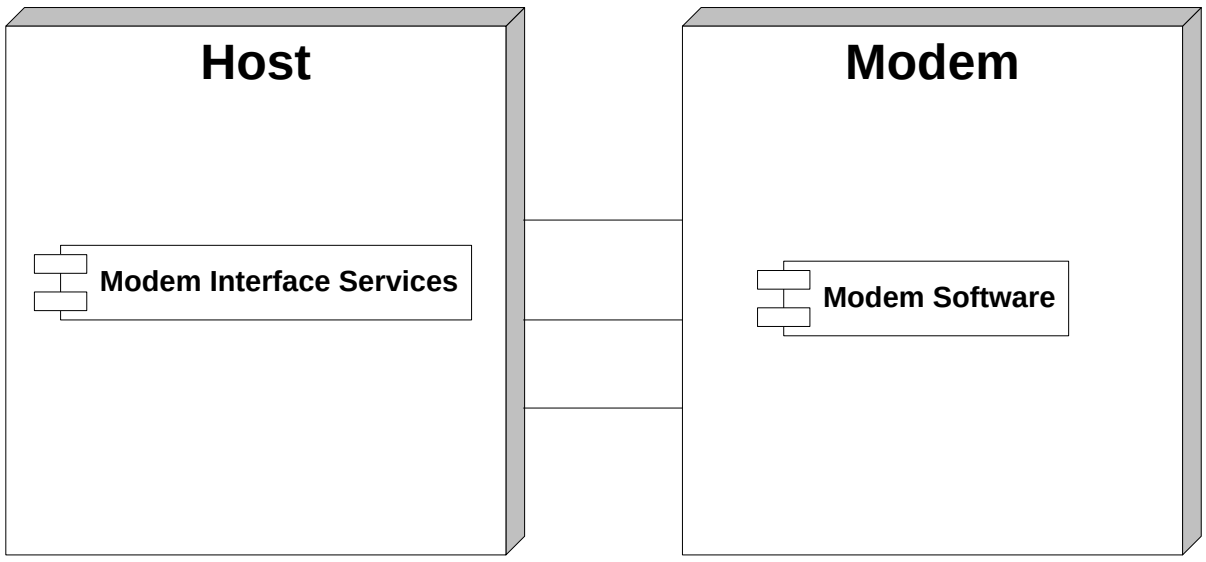

Figure 2 Modem Interface Services Overview

This figure is based on the official picture ST-Ericsson uses when showing relationship between the Host and the Modem. As shown in figure 1, MIS is an integral part of the Host.

Figure 3 depicts the position of the MIS on the Host and also shows the interface between the MIS and the Modem. The Modem consists of the hardware and software that provide the platform services to the external Host system. The MIS is the software component providing a bridge solution to the modem that is integrated to a Linux based Host system. The MIS can be considered as peripheral driver for the modem and does not contain any sensitive data. Nevertheless, if MIS components configuration files are tempered, it can lead to serious security problem. Any malicious changes in MIS components configuration files could make the whole device malfunction [6]. Hence the need to ensure that there is no vulnerability in the MIS components that can be exploited by attackers.

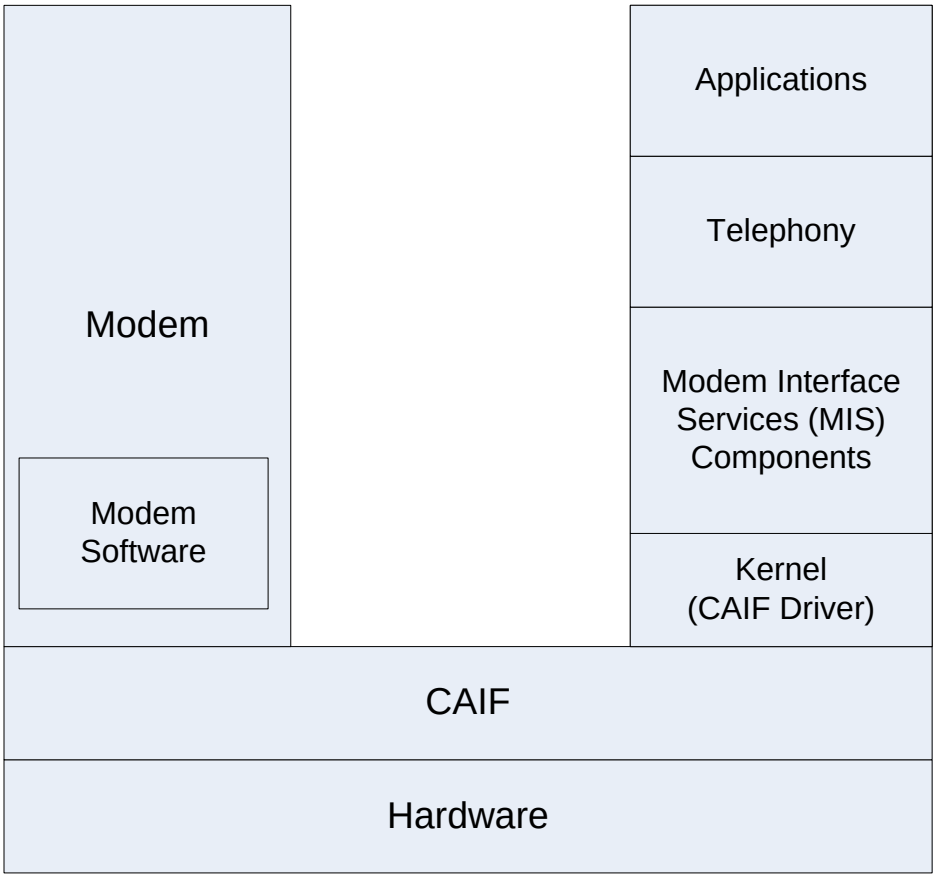

Figure 3 Platform Overview

This figure is based on the official picture ST-ERICSSON uses when showing components in the Android platform.

Research Questions

How can the Modem Interface Services be analysed to identify vulnerabilities in the Radio Interface Layer component and its environment?

Are they any mismatches between android and MIS components that will allow unauthorized access to resources?

1.3 **Literature Review**

In [8], Andreas Sjöström et al. (2010) evaluated the security of Access Linux Platform (ALP) 2.4, whiles taking into consideration security problems in the Linux operating systems and the security structure of ALP 2.4. They analysed vulnerabilities in Linux platform and demonstrated that some of them are in ALP too. In their paper [8], they presented vulnerabilities in the platform and proposed corresponding countermeasures to mitigate them.

In their paper [4], Michael Grace et al. analyzed eight popular Android smartphones and realized that the permission-based security model is not properly enforced. In this paper [4], Michael grace et al. stated that malicious application can delete user data, send SMS messages or record user conversation using this vulnerability.

L. Davi et al. Showed in their paper [16], that it is possible to mount privilege escalation attacks on Android platform. In this paper, L. Davi et al. detected transitive permission usage vulnerability in Android 2.0 and were able to escalate privileges

granted to application's sandbox. They concluded that non-privilege Android applications can escalate privileges.

In their paper, Google Android: A Comprehensive Security Assessment [29], Asaf Shabtai et al. assess the security of Android framework and identified possible threats to the platform. In this paper [number], Asaf Shabtai et al. indicate that it is possible for applications to exploit vulnerabilities in the Linux kernel or system libraries. They also mention the possibility of maliciously using permission granted to an installed application. In the same paper [29], Asaf Shabtai et al. mention that it is possible for an attacker to drain mobile device resources.

1.4 **Research Approach**

Software systems can be analysed for security vulnerabilities using different approaches. Vulnerabilities in software systems may be due to design flaws or implementation errors. In this master thesis work, three main approaches will be used to carry out security analysis of the MIS.

The first approach known as Threat Vulnerability and Risk Analysis (TVRA) technique [10], is the method that is used by ETSI standards developers in carrying out analysis of the threats, risks and vulnerabilities of telecommunications system. This method will be applied to detect vulnerabilities due to design flaws.

The second approach that will be employed in analysing the platform for security vulnerabilities is static analysis. Auditing code manually is time consuming especially when complex software systems are involved. Therefore, static code analysis method will be used to analyze the MIS component for vulnerabilities due to implementation errors.

Attack trees method will also be employed in this research work. This method will make it possible to graphically show the relations between different attacks, how they can be carried out by attackers and the impact of the attacks on the whole system. This method can detect design flaws as well as implementation errors.

In chapter two, detailed information and discussion of the three approaches is made.

1.5 **Limitations and Key Assumptions**

This research work covers security analysis of the Modem Interface Services and the interfacing Modem. The study concentrates on identifying security vulnerabilities in Radio Interface Layer (RIL) that can be exploited in order to access the Modem. In this analysis, the Modem is the asset to protect whiles the Radio Interface Layer is the asset under analysis. Additionally, this research takes into account the security policy of Android platform alongside the security mechanisms adopted by ST-ERICSSON. However, the research does not cover analysis of Android security mechanisms. Furthermore, the scope of this research has not been to find vulnerabilities in all MIS components. That would have been rather an extensive undertaking requiring more time than that allocated to this thesis work. The use of attack trees to do security analysis on systems has limitations [14]. Attack trees cannot be used to show the interaction between attacks performed on a system and the defences introduced to keep away attacks. As a result, there will be a limitation on the accuracy of analysing defensive measures. However, this method will enable me to show graphically the relations between different attacks on the RIL, how they can be performed and their impact on the system as a whole. Nevertheless, the intention is not to model every attack against the component.

The limitation on the part of static code analysis is that all static analysis tools estimate their outputs which results in output that is not very good. These tools can also give false positive or false negative results.

1.6 **Contributions to Knowledge**

This research will help to determine weaknesses in the MIS and interfacing Modem and find possible security threats that can be enacted by threat agents in order to exploit the vulnerabilities. Based on the identified vulnerabilities, countermeasures will be proposed to mitigate them. The outcome from the research will be a contribution to the efforts to analyze the Linux- based mobile platform for security vulnerabilities. The

findings of the research will create awareness of security problems in the mobile platform that will be analysed. The proposal to mitigate the vulnerabilities that will be detected will contribute to provision of secure mobile services on the mobile platform.

1.7 **Thesis Report Outline**

The rest of the thesis report is arranged as follows:

In chapter 2, the theoretical background of the master thesis work is presented. In this part of the report, detailed discussion about the three methods used for the analysis is made. It also contains exhaustive discussions on Android platform and the MIS.

In chapter 3, the approach to the master thesis work is discussed. Here, detailed discussion about the methodologies and the tools used in the research is made.

In chapter 4, the results of the analysis are presented in the form of applying the methodologies and the tools to the problem and obtaining the results.

In chapter 5, the results obtained from the analysis in chapter 4 are discussed and proposed solutions the vulnerabilities identified are presented.

In chapter 6, a brief summary of the problem, the approaches used and the results obtained is made. The contributions made and information about further work are also presented in this part of the report.

CHAPTER 2 Theoretical Background

2.1 Trust, Vulnerability and Risk Analysis (TVRA) Methodology

The TVRA approach to analysing systems involves systematic identification and prevention of unwanted incidents in the system. The technique consist of identifying the assets in the system and finding the weakness associated with each of the assets. For each asset, the threat and the threat agents that will exploit the weakness of the asset are detected. The likelihood of attacks on the system vulnerabilities and their impact on the system are modelled. This helps to determine the risk to the system.

The TVRA method is based on modeling. A TVRA model takes into account the following factors:

- Assets
- Weaknesses
- Threats
- Threat agents
- Unwanted incidents
- Countermeasures

A weakness of an asset that has corresponding threat is considered to be vulnerability; hence vulnerability is modelled as a combination of weakness that can be exploited by one or more threats. To reduce the risk in the system, countermeasures are introduced in the model to give protection against threats to vulnerabilities [10].

2.1.1 TVRA Method Process

Analysis of Telecommunication system for security vulnerabilities using TVRA method is a process. The method process consists of ten steps that are specified for analysing a system [10]. Below are the TVRA method processes:

1. Identification of Target of Evaluation (TOE). This consists of description of the main assets of the TOE and environment of the TOE. It also includes specifying the goal, purpose and scope of the TVRA.

2. Identification of security objectives. The outcome of this step is a high level statement of the security aims and issues that need to be sorted out.

3. Functional security requirements are derived from the objectives from step two.

4. Taking stock of the assets, taking into account high level asset description in step one and additional assets due to steps two and three.

5. Vulnerabilities in the system are identified and classified. The threats that can exploit the vulnerabilities and the unwanted incidents that may result from the exploitation are also identified.

6. Computing the likelihood of an attack and its impact.

7. Determination of the risks.

8. Finding the necessary countermeasures. The result of this step is a list of alternative security services and capabilities required to mitigate the risk.

9. Making countermeasure cost-benefit analysis. This is done to identify the best appropriate security services and capabilities among the alternatives from step 8.

10. Making specification of exhaustive requirements for the security services and capabilities from step 9.

Each of the steps in the method process has specific metrics outlined that will help the analyst to carry out security analysis of ICT system [10]. Specifically, repeatable metrics is used in step 6 to calculate the likelihood and impact values of attack. Cost and benefit calculation of alternative countermeasures is done in similar fashion in step 9. The repeatable metrics approach of calculating these values is necessary in that it allows for repeatability of the analysis over time.

The product of the likelihood of an attack and the value of the impact on an asset gives indication of the risk to the asset. Introduction of a countermeasure will minimise the probability of success of a threat to an asset and lessen its impact. In step eight of the

process, the analyst identifies alternative countermeasures needed to protect the assets in the system. The objective of the cost and benefit evaluation in step nine is to determine the most effective countermeasure that can mitigate the effect of an attack at a reasonable cost.

The knowledge of the security objectives of the asset and its environment is necessary to determine the type of protection needed. The identified security objectives may be categorized into one of the following five groups:

- Confidentiality
- Integrity
- Availability
- Authenticity
- Accountability

Similarly, identified threats to a system may be classified into one of the following major groups:

- Interception
- Manipulation
- Denial of service
- Repudiation of sending
- Repudiation of receiving

The methods used to enact these threats are numerous and different [10].

2.2 Static Code Analysis

In the 1970's, Stephen Johnson, then at Bell Laboratory developed a tool by name Lint to scan C source programs that had compiled devoid of errors in order to unearth bugs that had evaded detection [17].

There are many methods that can be used to discover and lessen the number of errors in a program [17-18]. Requirements stage or use case phase of product development are better stages to check for flaws. These stages are however not developed for standardization. Many security weaknesses in programs occur during coding. Therefore it may be advantageous to make corrections after design stage [17, 18]

While binary and byte code are not in human readable forms, it is possible for humans to read source code. Besides, many tools exist for analysing source codes. On account of these reasons, source code appears a fine place to start analysis [17].

Over the years, research has shown that code review is the best method to get rid of bugs in a program [18]. However, it is not always possible due to the difficulty involved in training people and organizing them to detect errors in programs. Moreover, it is time- consuming.

Most errors in programs can be classified into categories [15, 17]. Therefore automated tools can be used to detect them in programs. Static analyzer is a program developed for analysing other programs for possible flaws. These tools are used to scan the source code of a program so as to match patterns. They examine the programs for errors without running them in a procedure called static code analysis.

The success of Lint serves as pacesetter and has led to proliferation of static analysis tools, both open source and proprietary and they target different languages and operating systems.

There are different types of static code checkers. Some are used to scan the source code of programs while others can examine compiled byte code.

While each static code analyzer functions differently, most of them have some characteristics in common. They all read the program and create certain model of it. The model then serves as a sort of abstract image that can be used to match error patterns. Static analysis tools also carry out data-flow analysis. In this process, they try to deduce likely values variables could store at particular points in the program. Data-flow analysis is very important when it comes to analysing program security vulnerability. Programs receive input from users. Each time the program accept input, there is a likelihood that attacker can use that input to destabilize the system [15, 17, 18].

There are many methods and techniques in static analysis. The main methods used in code analysis comprise the following:

- Control Flow Analysis
- Data Flow Analysis
- Information Flow Analysis
- Path Function Analysis (also called semantic analysis or symbolic execution)
- Formal Verification (also called compliance analysis)

Other techniques that can be used to analyze code of a program include syntax checks, range checking and timing analysis [20].

2.2.2 Static Analysis Tools

There exist a number of tools for performing static analysis. These tools differ about the complexity of the analysis they can do. Some of them function on one or two languages. Majority of the tools run on source code. Basically, static analysis tool converts the code into a language which it can interpret. The effectiveness of these tools depends on the rule set define for each of them. The amount of rule sets also depends on the type of programming language [20].

2.3 Attack Tree Methodology

The attack trees concept was defined by Bruce Schneier in 1999, as a means of modeling threats on computer systems and the mode of using attacks to realize those threats. The basis of attack tree methodology is fault tree analysis [11, 14], which is a method used to comprehend the way systems can fail.

Attack trees are ways of thinking and describing the security of systems and sub-systems. They can also be described as a means of building an automatic database that serves as descriptions of the security of a system. The methodology can be applied in making decisions regarding how to improve security or minimise the effect of a new attack on a system [11, 25]

In creating attack trees, the attacker's objective forms the root node and is connected to sub-nodes which represent the possible attacks to accomplish the attacker's main objective. The leaf nodes correspond to attacker's actions. One attack tree represents one objective of the attacker. In any complex system, the root nodes are many and each represent a different goal. Thus in modelling attacks against a complex system, each attack objective has to get a different tree [14, 25]. Figure 4 shows a basic attack tree.

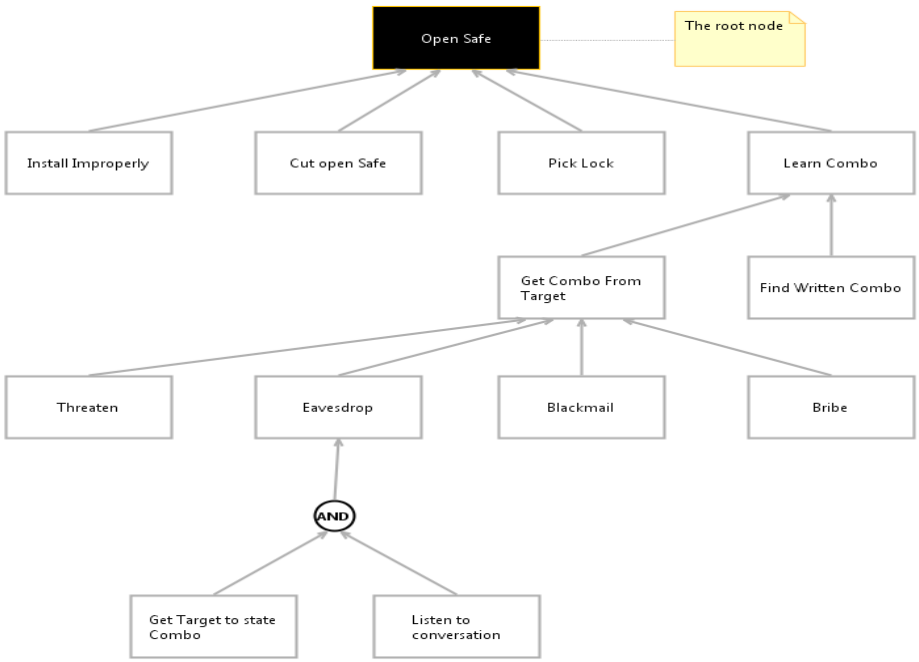

Figure 4: Basic Attack Tree

This diagram was drawn based on the official Attack tree diagram Bruce Schneier uses to illustrate attack on a safe.

The goal of the attack is to open a safe and nodes beneath the root node are the different means of achieving the goal.

The "OR" nodes represent different means to achieve the same attack goal. For instance, to learn the combo you can either get the combo from the target OR find written combo. The "AND" nodes represent different actions to realize a goal. For example, to eavesdrop a message you need to get target to state the combo AND listen to conversation.

An attack tree can be integral part of a larger attack tree. For instance, Figure 4 is part of a larger attack, whose goal is to read a document [25].

Possible attacks against a system can be modeled using attack tree. Once that is done, the tree can be used to analyze the attributes of the security of the system. A number of such attributes have been proposed by Bruce Schneier [12].

In order to use the tree to analyze the threats and attacks against a system, values are added to the leaf nodes [25]. The analysis of the system is done in two steps [12]. Firstly, the value for each leaf node is determined. Secondly, the value in non-leaf node is created from the values of its children. To determine the vulnerability of a system to an attack, the analyst examines the value of the root node to ascertain if the goal is vulnerable to attack. For instance, the presence of a possible Boolean value in the root node is an indication of the system's vulnerability. Similarly, the analyst can verify if the system is vulnerable to a specific type of attack. Furthermore, using attack tree, a security analyst will be able to obtain the assumptions on which a security system is based [12, 25].

2.4 Android Security Mechanism

Generally, android integrates a number of security mechanisms into the framework [29]. These can be classified into three main groups as:

- Linux mechanisms
- Environmental features
- Android specific mechanisms

2.4.1 Linux Mechanisms

The Linux layer in Android platform implements two security mechanisms namely, the Portable Operating System Interface (POSIX) users and file access. Users are the main components in these mechanisms. A process or a file known as object is owned by a user. Furthermore, the users are allocated to groups [29]

POSIX users

In the case of POSIX users, every Android application installed on the mobile device is given a user ID. This means that two applications' code can not run in the same process. This somewhat constructs a sandbox that precludes interference of applications with one another. Two applications need to share the same user ID in order to have the same permissions set and run in the same process. This can only be possible if the two applications explicitly declare the usage of the same sharedUserID feature. Besides, they need to be digitally signed by the same author [29].

File access

Android implements Linux ownership and permissions security mechanism [29]. Both application files and system files are exposed to these security mechanisms. Every file in Android is related to the owner's user ID and group ID. Additionally, the users and the group members can be assigned any combination of the three Read (r), Write (w), and eXecute (X) permissions.

In Adriond, the Linux kernel enforces read and write privileges on the owner while the users in the same group with the owner have write permission and the rest of the users are given execute permission [29].

Normally, in Android, user of an application owns the application file whiles either the system or root user owns system files. Ensuring file access security involves assigning various users to every application and system files together with correct permission configurations.

Besides, the Linux kernel treats various functions in the system as pseudo-files. Therefore, the use of file access mechanism makes it possible to set permissions on drivers, terminals and hardware sensors. Permissions can also be set on power state, audio, direct input readings and the like [29].

2.4.2 Environmental Features

Environmental features comprise the hardware, programming language and mobile carrier mechanisms that contribute to improve the security of Android platform. The specific technological factors that need to be considered in Android security are memory management unit, type safety and mobile carrier security features [29].

Memory management unit

Memory management unit is a hardware component that enables separation of processes into distinct address spaces. Linux, like any other operating system is required to have memory management unit to make sure that no process can read or corrupt the memory pages of another process. The use of this mechanism can prevent information disclosure and also reduce the probability of privilege escalation [29].

Type safety

This is a property of a programming language. It is required that contents of a variable maintain a certain format in order to eliminate the possibility of misuse. The absence of type safety and boundary checks can result in memory corruption and buffer-overflow attacks. These attacks can further lead to execution of arbitrary code [29].

Mobile carrier security features.

The security features of telephony systems originate from the need to identify users, monitor usage and charge subscribers appropriately. The approach to meeting these basic requirements is the implementation of authentication, authorization and accounting mechanisms into the telephony system. As a platform for smart phones, Android makes use of these security attributes mobile phone design [29].

2.4.3 Android-Specific Security Mechanisms

In a bid to provide a secure platform, Android enforces the following security mechanisms:

- Application permissions
- Component encapsulation
- Application signing

[1, 28-30].

Application permissions.

The permission mechanisms for Android applications limit applications to exact operations they can do. There are much built-in permissions in Android that control operations such as dialling the phone, taking pictures, using the Internet and the like. There is flexibility in Android that makes it possible for any application to declare additional permissions. In order to acquire permission, an application needs to explicitly ask for it [29]. Every application package has a related XML manifest file. A developer can assign permission labels to application using this manifest file. Whiles permission labels defined for an application spells out its scope of protection, giving permissions to the components that make up an application is a means of specifying the access policy to defend its resources [1].

Each permission has its related protection level. The protection levels for Android applications are designated as "normal," "dangerous," "signature," and "signature-or-system." During installation of Android application, the protection levels of the permissions that are asked for are verified. Permission whose related protection level is normal is granted all the time. Permission which has dangerous protection level can not be given to application without approval from a user. This type of permission might grant access to private data or sensitive utilities on the mobile device. A permission that is protected by signature is given to only applications that are signed using the same developer key that signed the application declaring the permission. The "signature - or - system" protection level functions in similar fashion as signature protection. Additionally, it can also be given to applications installed in the system image [28, 29].

The system gives the permission requested to applications requesting it during installation time. Granting of the permission is based on user's approval and signature checking. After installation, the application can no longer make any permission request. This means that any operation that has not been approved during installation time will not be successful at execution time [28, 29].

Component encapsulation.

Android application can prevent other applications from accessing its components. This is achieved through a process known as component encapsulation. This happens mainly by declaring the "exported" property of the component [29]. Given that this property is made "false", only application that owns the component in question and other application that share the same user ID can gain access to it. Providing the

"exported" property is set to "true", other applications can gain access to it. Developers should set the "exported" property physically since there might be a discrepancy between the default encapsulation policy and the one expected [1, 29].

Signing applications

Every Android application is put in a package named apk archive [29], meant for installation. The prerequisite of Android applications is that all the installed applications need to be digitally signed. The package is considered valid provided that its certificate is useful and acceptable. Additionally, the public key included must be able to confirm the signature. This mechanism is used to verify the origin of two or more Android applications. The approach is also very useful in SharedUserId method and permission mechanism to confirm signature and signature-or-system protection level permissions [29, 30]

2.5 Modem Interface Services (MIS)

The Modem Interface Service is a software component that runs on the host CPU in the host operating system environment. It is designed to provide a bridge solution in the Android platform. The MIS can be considered as the "peripheral driver" for the Modem and it is intended to make the modem fit into the Linux-based host system.

The application system in the platform is called the host and the modem is thought to be a "peripheral." The modem consists of hardware and software components meant to provide platform services to the external host system. It is used for configurations such as bridge, pc-cards and USB dongle [6]. Figure 5 presents detailed description of the Modem and Host relationship.

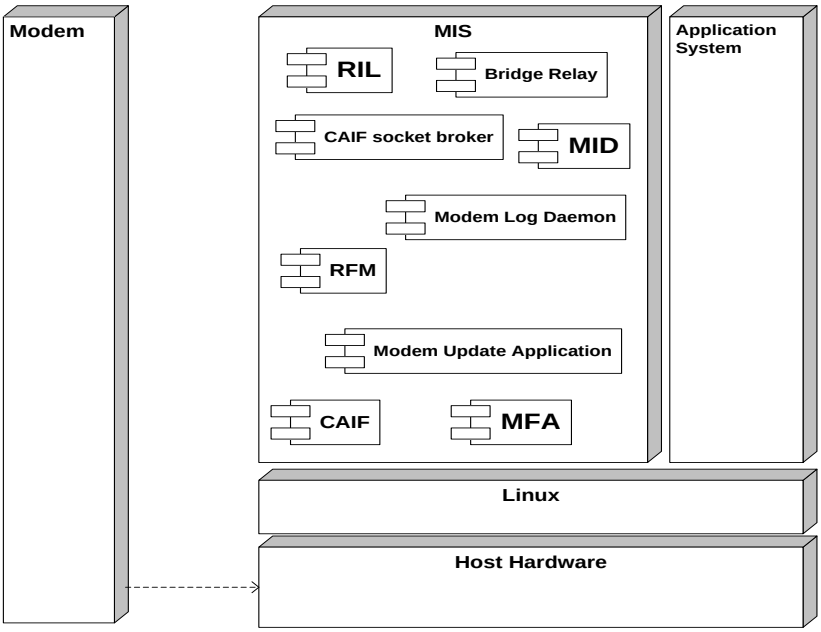

Figure 5: Host and Modem Relationship

This figure was drawn based on the official picture ST-Ericsson uses to show detailed description of the platform.

2.5.1 Modem Interface Services Modules

The design of the MIS component is in modular format. The MIS consists of a number of modules working together as a unit to implement the bridge solution function. The modules that make up the MIS are ModemInitDaemon (MID), Modem Flashing Application (MFA) and Remote File Manager (RFM). Others are Radio Interface Layer (RIL), Inter Process Communication Multiplexer (CAIF) and ModemLogDaemon. The rest are ModemUpdateApplication, BridgeRelay and CAIF Socket Broker [6].

ModemInitDaemon

The MID module in the MIS component controls the modem in that it starts and stops the modem as well as monitors it. It also coordinates all service-associated functions of the modem namely flashing and update. Additionally, the MID gathers crash reports from the modem [6].

Radio Interface Layer

The RIL is MIS component. It is the main mechanism used to integrate the modem in Android platform. The RIL enables modem - supported services in the platform. To communicate with the modem, the RIL uses the D-bus. For instance, it uses the D-bus API to perform modem initialization [6].

Modem Flashing Application

The MFA module runs loader protocols in order to flash flash-based modems. However, this function of the module is not used when flashless modem is being used. This module is also responsible for tunnelling used in the course of service and flashing from a computer [6]

Remote File Manager

The RFM is a Portable Operating System Interface (POSIX) remote file system server. A flashless modem uses this module to store non-volatile data [6].

ModemLogDaemon

This module is responsible for receiving records from the modem and writes them to the file system on the host [6].

ModemUpdateApplication

The ModemUpdateApplication gives information about new firmware to a flash –based modem. This function is not used for a flashless modem [6].

BridgeRelay

The BridgeRelay implements tunnelling functions. These functions of the module are needed in the factory and during research and development [6].

CAIF Socket Broker

This module in the component provides inter process communication multiplexer sockets to user space. Besides, it serves as an interface to Field Test Display and CIQ [6].

2.5.2 Modem Interface Services Security

The security implementation in the MIS follows the general Linux security in that each application run as a separate user. MIS does not implement enhanced security functions.

The Linux kernel on which MIS runs is either in an open or locked mode. When the kernel is in open mode, the mobile device is said to be open and when it is in locked mode the device is termed locked device.

When the device is in locked mode, software that is not operator signed cannot be loaded on the device. With this configuration, MIS security depends entirely on Linux security and is the basic means of protecting all sensitive data. Conversely, a user may alter the kernel if it is not in a locked mode to allow access to any MIS data.

The MIS contains no sensitive data. However, the MIS configuration file may be at risk if the device is open. Malicious changes in the configuration file could cause the whole device crash. Moreover, misaligning the GP10s in such away that the ones in the direction of the modem are powered when the modem is off may damage the modem.

The MIS executables run as regular users and are assigned the needed capabilities to carry out operation. However, some modules are given special capabilities in order to function. Particularly, MID, BridgeRelay and RFM start with root privileges but drop the privileges in the course of start up [6].

2.5.3 MIS Security Solutions

The security solution for the various modules in the MIS component is founded on Linux functionality. Security setting for each executable depends on the functions of the module. The following section describes security setting for the main executables:

MID executables

The MID executable starts as a root user. However in the course of the start up, it switches to become a “system” user and be a member of the radio group. It is assigned the privileges required to start and stop services. Additionally, the MID opens privileged resources at start up before changing to system user.

MID is assigned the following capabilities required for it to carry out its normal operation: CAP_NET_ADMIN, CAP_NET_RAW, CAP_SYS_BOOT and CAP_SYS_MODULE. Furthermore, the radio group and user owns /dev/h510 device [6].

MFA executables

The MID module starts MFA module. In view of the fact that MID belongs to the radio group and runs as the system user, MFA will be assigned the same user and group when MID starts it. This user and group are given the permission to access the

/dev/tty02 device. Besides, the system user and the radio group can access / data/ste /modem directory [6].

RFM executables

The remote file manager starts with root privileges and opens Dbus connection. It then drops the privileges during the start up. The module is assigned the following capabilities required to perform its usual operations: CAP_NET_ADMIN, CAP_SYS_CHROOT and CAP_CHOWM. The user and group of RFM can be changed by configuration means from “init.rc” [6].

BridgeRelay executables

The BridgeRelay starts with root privileges and drop the privileges in the course of the start up. It then switches to radio user to become a member of the radio group [6].

LifeCycleLogger executables

This module runs as radio user and belong to the Bluetooth group [6].

Chapter 3 Research Method

This chapter consists of the approaches used to analyse the MIS component for security vulnerabilities. In the following sections, the various approaches and the tools used are explained in detail including their suitability to analysing software for security vulnerabilities.

3.1 MIS Security Analysis Approach

Software vulnerabilities are errors that attackers can use to compromise software systems. These errors can be defects in the software design and implementation as well as testing and operation. However, software vulnerabilities can be classified into two [22], namely: design level flaws and implementation level errors. Therefore, security analysis of the MIS module is based on this classification. The approach used to detect implementation level vulnerabilities is source code analysis.

The existence of design error in the software can create an opportunity for an attack even if the software does not have any implementation error. For instance, lack of authentication between two interactive components is architectural flaw and can not be detected using static code analysis [22]. Therefore, TVRA method is employed in this analysis to identify architectural vulnerabilities. Architectural vulnerability analysis consists of known vulnerability analysis, ambiguity analysis and underlying platform vulnerability analysis [22]. TVRA methodology captures the three architectural vulnerability analysis approaches in its method process. Thus, it is applied to MIS security analysis.

As discussed in chapter [2.3], attack trees analysis is a systematic and well structured means of detecting possible vulnerabilities in systems. It is possible to employ this method as part of risk analysis process and it can be applied to detect security vulnerabilities at design level and development stage [34]. Hence, this method is used addition to TVRA approach and static code analysis in this research work.

3.2 Code Analysis Tools

There exist several static code analysis tools for analysing source codes written with C/C++ language. Some are designed for commercial purposes while others are open source and can be used freely. Some are also platform specific whereas others are platform independent and function on several platforms such as Linux and Windows machines.

This section focuses mainly on the tools used to perform the analysis on the MIS component. In this analysis, the following four static code analysis tools were used:

- Flawfinder 1.27-3
- RATS 2.3-1
- Cppcheck 1.54
- Yasca 2.21

These tools are selected based on the type of language used for writing MIS code. Since no static analysis tool can detect every security problem in a program, the use of four tools for the analysis will lead to better results. In the following section, detailed description of these tools is made.

3.2.1 Flawfinder

Flawfinder is an open source tool that scans source code to detect possible security vulnerabilities. The vulnerabilities detected are organized according to risk level. The tool functions on UNIX as well as Linux systems [35].

The tool has a built-in database of C/C++ functions with well-known security problems such as buffer overflow risks, format string problems, race conditions and poor random number acquisition. Thus, its capability depends on the database which comes with it.

The output of flawfinder analysis contains a list of potential security flaws called “hits” which are organized according to the risk level. In fact, not all the problems that can be detected using this tool are security vulnerabilities and it is also true that not all vulnerabilities can be detected using Flawfinder [35].

3.2.2 RATS

RATS is an acronym which stands for Rough Auditing Tool for Security. It is an open source tool developed for analysing source codes for security vulnerabilities. RATS as a static analysis tool can be used to scan source codes written in C, C++, Perl, PHP and Python. The tool generates a report at the end of a scan which highlights common security related errors. The result also gives a description of the problem and potentially

suggested remedies. The tool cannot find every error. Neither does it detect only problems that are vulnerabilities. In fact it only performs a rough analysis of source code [36].

3.2.3 Cppcheck

Cppcheck is a static source code analysis tool that is used to analyze codes written in C/C++ languages. This tool can operate on any platform and detects some bugs that cannot be detected by the C/C++ compilers. However cppcheck has its own limitation. It cannot detect all possible bugs in a code. This tool can be operated in a command line mode as well as in the GUI mode [37].

3.2.4 Yasca

Yasca is an open source static analysis tool that scans source codes for security vulnerabilities as well as check for code quality. It also checks for performance of the code and its conformance to best practices.

Yasca can scan source codes written with many languages. It actually leverages other static analysis tools when it has to scan specific file types. Yasca contains many custom scanners which are Yasca specific. The tool is used in the command line mode and it generates reports in HTML format. It has a plug-in architecture hence it can be extended [33].

3.3 Security modelling tools

In the field of science, when analysing a specific problem, it is normal to construct an ideal and slightly simpler models of the problem that will help to comprehend the phenomenon by assessing the models with respect to reality. It is therefore usual for security researchers to model systems and attacks from security perspective [31]. Modeling security is a vital aspect of software security. Particularly, it is a means of creating awareness of security situations [26]. In this research work, two modeling tools, SeaMonster and UML were used to model security in the MIS components. A brief description of these tools is made in the following section.

SeaMonster

This is a graphical security modeling tool. The base of this tool is a set of eclipse frameworks. It can be applied to model security activity graphs and attack trees. Additionally, it can be employed in modeling misuse case diagrams as well as vulnerability case graphs [26].

This tool was used to model various attacks against the platform taking into account security vulnerabilities in the MIS.

UMLsec

UML is a modelling language. There exists no modelling tool for UMLsec. Nonetheless, UML editor can be extended to bring in UMLsec [26]. In this research work, Microsoft Office Visio 2003 was used to model UML model diagrams and the block diagram.

CHAPTER 4 Research Results

4.1 Applying the TVRA Methodology

As discussed in chapter three, the TVRA methodology provides a systematic procedure for discovering possible security vulnerabilities in telecommunication systems. Applying this method to analyse security in the MIS is a means to identifying design flaws.

4.1.1 Target of Evaluation

The goal of this approach is to carry out security analysis on the MIS and the interfacing Modem. The objective of using this approach is to find vulnerabilities that exist in the system due to architectural errors. The modem and the MIS form the environment of this analysis while the Radio Interface Layer (RIL) is the asset under analysis. The analysis focuses on identifying vulnerabilities in the RIL component. As shown in Figure 1, RIL is the MIS component that interfaces Android telephony layer and links the modem to Android applications in the upper layer. Its main function is to process all communication with radio hardware and sends calls to the RIL daemon. Since RIL exists in the MIS and its functions are part of the overall functions of the MIS, any attack or malfunction on the part of RIL will certainly affect the whole system. The main focus of the analysis is then on finding those vulnerabilities that will affect the operation of the modem. Figure 6 presents the main assets in the environment of this analysis.

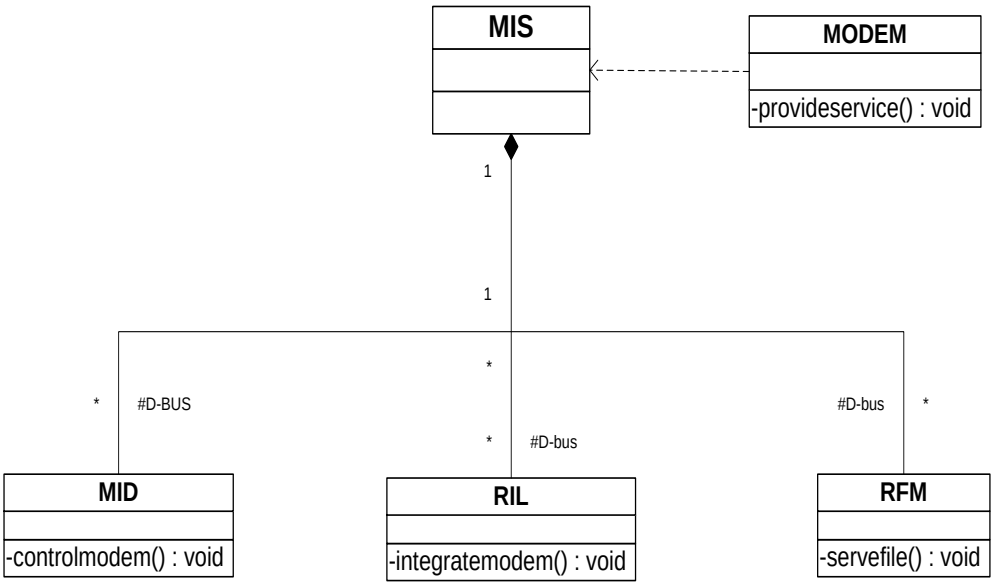

Figure 6: Target Asset and its environment

Knowing the security environment of the MIS and the modem is crucial in analysing RIL for security vulnerabilities. Table 1 in Appendix B presents the needed information on the security environment for analysing the RIL.

As discussed in chapter [2.], the MIS provides bridge solution for the modem in Android platform which runs on Linux kernel. The modem provides the platform services to the external Host system. The MID controls the modem while the RIL enables modem-supported services. The RFM acts as the file server to the modem.

The identified possible threats to MIS and the modem are:

- Loss or corruption of data
- Unauthorized access
- Denial of service
- Masquerade

4.1.2 Security Objectives

The role of RIL in the MIS makes it an important component. The RIL should ensure availability of modem supported services. Additionally, it must ensure that the services are accessible by authorized entities or applications. However, the availability of the modem itself is the responsibility of the entire MIS. Since the file system is mounted on the Host, the MIS should ensure integrity of modem-related data on the Host. Table 2 in appendix B shows information on security objectives for RIL and MIS.

4.1.3 Security Requirements

The security requirements for RIL and its environment depend on the identified objectives. The requirements consist of security functional requirements and assurance requirements. However, in this thesis work, only security functional requirement are determined. The RIL is designed to meet the following functional requirements:

- Initialise the modem to provide platform services
- Ensure secure communication between the modem and Android telephony framework.
- Relate with MID to keep track of the modem

The Android platform is designed to meet the following security requirements:

- The MIS shall ensure integrity of modem related data
- The MIS shall ensure secure communication among components and the modem
- There shall be authentication between modules and the Modem

Table 3 in appendix B depicts security functional requirements for RIL and the MIS as a whole. As discussed in the chapter 2, vulnerability in an asset is modeled as a combination of weakness and threat to that weakness. Based on the security objectives of RIL and its environment, possible threats to the system have been identified. Table 4 presents identified threats to the security objectives of the target asset and its environment.

Threat	Objective			
	Confidentiality	Integrity	Availability	Authenticity
Denial of Service			√	
Loss or corruption of data		√	√	
Unauthorized access	√	√		√
Masquerade	√	√		√

Table 4: Identified Threats to Objectives

4.1.4 Inventory of the Assets

RIL is a software component which forms part of MIS component. In Android platform, the RIL is used to integrate the Modem into the platform. The purpose of the integration is to enable telephony services (example: call, data messaging and Network

services) provided by the Modem. The RIL receives all the telephony requests from Android telephony framework and carry out mapping of the requests to the appropriate AT command. These AT commands are relayed to the Modem. In a similar fashion, the RIL translates AT commands from the Modem into a type of response Android telephony framework will understand. AT command is a means of invoking and controlling modem services [6, 27].

As mentioned in the preceding sections, RIL is the asset under analysis in this master thesis. Therefore, the impact of attack on RIL to the system must be taken into account. Table 5 shows the impact of attack on this component to the Modem.

Impact	Explanation	value
High	The Modem might be damaged using vulnerability in RIL	3

Table 5: Asset Impact

4.1.5 Identification of Vulnerabilities

To launch a successful attack on the MIS in general and the RIL in particular, an attacker must identify vulnerability in the platform. Vulnerability exists when there is a threat to a weakness in the platform. Therefore, to identify vulnerabilities in the RIL and its environment, weaknesses in the system must be identified. This analysis takes into account the flow of information in the D-Bus between the RIL and the modem as one attack interface to identify weaknesses. The other attack interface used in the analysis is the RIL to MID communication over the D-Bus. Analysis of the well-documented information on the platform reveals some design weaknesses which can be classified as follows:

- Incorrect permission assignment for critical resource
- Privilege dropping error
- Missing encryption of sensitive data

Further analysis shows that these weaknesses are exploitable using different attack methods. Using information in well- documented Common Weakness Enumeration

(CWE) paper [21], the following possible attacks can be used to exploit the identified vulnerabilities:

- Modification of data
- Assuming of Identity/ Gaining privileges
- Reading of files

An attacker launches attack on a system taking into account the type of vulnerabilities that exist in the system. Table 6 presents the weaknesses in the target asset and its environment as well as their corresponding possible attacks.

Weakness	**Possible Attacks**
Incorrect permission assignment for critical resource	1. Modification of data 2.Gaining privileges
Privilege dropping error	1. Gaining privileges/Assuming identity
Missing encryption of sensitive data	1.Reading of files 2. Modification of data

Table 6: Weaknesses and Possible Attacks

In order to exploit vulnerabilities in any system, an attacker uses the method that is suitable and easy to carry out. As explained in the preceding section, the method also depends on the type of attack and the vulnerability. In this analysis, the identified methods which can be used to exploit the vulnerabilities are listed below:

- Injection
- API abuse
- Protocol manipulation
- Executing arbitrary code

Detailed discussion of the vulnerabilities, the attacks and the attack methods is made in chapter 5.

4.1.6 Assessment of the Practicality

The presence of vulnerabilities in any system is not a guarantee that an attacker can compromise the system. Therefore, in performing security analysis on a system, attack potential associated with each vulnerability identified should be determined. To do this, certain factors must be taking into account. In this master thesis work, the factors used to perform the analysis are defined in clause B.4 of the Common Criteria Evaluation Methodology. This information is available in ETSI's paper on TVRA methodology [10].

4.1.7 Calculation of the Likelihood of the Attack and Its Impact

The factors used to perform this analysis are time, expertise, knowledge and opportunity in addition to equipment. In performing the analysis, worst case is assumed for the factors outlined to determine the attack potential for the vulnerabilities identified. Furthermore, weighted summation method is used to calculate the overall attack potential. Table 7- Table 9 in appendix B present attack potential for the various identified vulnerabilities.

The intensity of an attack on a system determines the risk related to the system. Thus, in performing security analysis, attack intensity for each attack on the asset under analysis must be determined. Moderate levels of multiple instances of attacks on RIL with a value of one are used in performing this analysis. Table 10 in appendix B presents information on attack intensity on the RIL.

Vulnerability Ratings

Vulnerability ratings are used to determine feasibility of an attack on the platform. The overall attack potential values are used to rate the vulnerabilities. Table 11- table 13 in appendix B present the rating for the identified vulnerabilities. The results of the rating show that the identified vulnerabilities have high resistant to the attacks. The vulnerability rating parameters obtained are then used to determine the likelihood of an attack on the platform using any of the vulnerabilities identified. Table 14- table 16

present the mapping of the vulnerabilities to likelihood of attack. The analysis shows that it is not likely for an attacker to exploit these vulnerabilities.

Vulnerability Rating	Likelihood	Value
High	Unlikely	1
High	Unlikely	1

Table 14: Likelihood of attack using incorrect permission assignment for critical resource

Vulnerability Rating	Likelihood	Value
High	Unlikely	1

Table 15: Likelihood of attack using privilege dropping error

Vulnerability Rating	Likelihood	Value
High	Unlikely	1
High	Unlikely	1

Table 16: Likelihood of attack using missing encryption of sensitive data

4.1.8 Establishment of Risk

Determining risk associated with each vulnerability in security analysis is a means of measuring the risk implication of an attack on an asset. In ETSI standards [10], risk is defined as the result of multiplying the likelihood of an attack and the impact the attack has on the target asset. In this analysis, the impact of the various attacks on the RIL is obtained by summing the asset impact value (From Table 5) and attack intensity values (from Table 10). According to ETSI standard [10], the impact is measured from 1 to 3. Therefore any sum more than 3 is taken to be the maximum value which is 3. Hence, the resulting impacts on the platform are 3. Table 17 in appendix B shows the calculation of the impact of attacks on the asset hence, the platform. The product of these values and that of likelihood of attacks (from Table 14 – table 16) determine the

risk values. Table 18 presents the risk values and their classifications. As shown from the analysis in Table 18, the risks associated with the vulnerabilities are major. Hence, mitigation strategies must be put in place to avert any eventuality.

Likelihood of attack	Impact on Platform	Risk Value	Risk Type
1	3	3	Major
1	3	3	Major
1	3	3	Major
1	3	3	Major

Table 18: Risk Calculation and classification

4.2 Applying the Static Code Analysis Methodology

As discussed in chapter 2, Static code analysis approach provides the means of identifying implementation errors in programs. This approach of analysing software has been applied in this research work. The four static analysis tools mentioned in chapter 3 were used in the analysis. The idea of using four tools is to obtain better and accurate results from the analysis.

The analysis was carried out using an older version of ST-Ericsson's Radio Interface Layer (RIL) component source code. The source code comprises forty-two files. Nineteen of them are C/C++ source files. The results obtained from scanning the source code are presented in the following sections.

4.2.1 Flawfinder 1.27-3 Results

Flawfinder was used on Linux (Ubuntu) platform to analyse the source code.

Running the test with this tool, 16 files were checked and 17422 lines of code were analyzed from which 151 problems were detected.

With flawfinder, errors detected are classified according to severity levels starting from zero to five. By default, one is the minimum risk level. The output of the analysis contains information about the type of errors in the code, their severity level and the possible attack description. Table 19 shows discovered implementation errors in the source code and the possible attacks against the system using these vulnerabilities. Table 20 depicts the severity levels and the number of errors at each severity level.

Error type	Possible Attacks
(Buffer) char	Program crash, Data corruption, Malicious code execution
(Integer) atoi	Program crash, Data corruption ,Arbitrary code execution
(buffer) memcpy	Program crash, arbitrary code execution, Subversion of other security services
(Misc) open	
(Buffer) strcpy	Program crash
(Buffer) read	Program crash, Data corruption
(Buffer) strncpy	Program crash, Data corruption
(Obsolete) usleep	Old function
(port) snprintf	Buffer overflow

Table 19 Vulnerabilities and Possible Attacks

Severity Level	0	1	2	3	4	5
Number of Errors	0	69	66	1	15	0

Table 20 Error Severity

4.2.2 RATS 2.3-1 Results

RATS analysis tool was also used on Linux (Ubuntu) platform to analyse RIL source code. Analysing the source code with this tool, all the 19 C/C++ source files were checked and 15904 lines of code were analyzed. From this, 43 problems were detected.

The output of the analysis contains a list of the vulnerabilities present in the code and their severity level. Table 21 shows implementation errors detected in the source code using this tool. It also contains information about possible attacks against the system using these vulnerabilities. Table 22 depicts the severity levels and the number of errors at each severity level.

Error Type	Number	Possible Attacks
fixed size local buffer	24	Buffer overflow
strcpy	14	Buffer overflow
getopt	1	Buffer overflow
sprintf	1	Buffer overflow
read	3	Buffer overflow

Table 21 Types of Errors and Possible Attacks

Severity Level	High	Medium
Number of errors	40	3

Table 22 Errors Classification

4.2.3 Cppcheck 1.54 Results

Cppcheck analysis tool was used on windows vista platform to analyse the RIL source code. The tool was used in the GUI mode and the output report generated is in xml format. Analysing the source code with this tool, all the 19 C/C++ source files were checked and from this, 43 problems were detected. The output report of the analysis lists the flaws present in the code and indicates their severity level. Cppcheck classifies implementation flaws as error or warning depending on their severity. However, the output report does not show the number of lines analyzed and attack description. Table 23 shows vulnerabilities detected in the source code using this tool. It also presents possible attacks against the system using these vulnerabilities. Table 24 shows the statistics on the classification of the vulnerabilities with respect to severity level.

Error type	Number	Possible Attacks
Null Pointer	11	Availability-Failure of the process
Memleak	1	Reliability problem-Denial of service attack
SelfAssignment	2	
Invalidscanf	3	Program crash

Table 23 Types of Errors and Possible Attacks

Severity Level	Error	Warning
Number of Problem	12	5

Table 24 vulnerabilities Classifications

4.2.4 Yasca 2.21 Results

Yasca static analysis tool was used on windows vista platform to analyse the RIL source code. The tool was used in the command line mode and the output generated in html format.

Analysing the source code with this tool, 13 of the C/C++ source files were checked and from this, 123 problems were detected. The output report of the analysis lists the flaws present in the code and indicates their severity level. Yasca classifies implementation flaws into three categories namely, critical, warning and low. The generated report also contains information relating to consequences of these vulnerabilities. But, the output report generated does not show the number of lines analyzed. Table 25 shows vulnerabilities detected in the source code using this analysis tool. It also describes possible attacks against the system using these vulnerabilities. Table 26 presents the data on the classification of the vulnerabilities about their severity level. Detail information relating to this analysis can be seen in the output report in appendix B.

Error Type	Number	Possible Attacks
Weak Credentials (password)	2	Brute-force attack
Use after free	41	Program crash,

		unexpected value, arbitrary code execution
Unsafe/banned Function	68	Memory corruption
Improper Null Termination	1	Buffer overruns
Double free	11	Memory corruption, Buffer overflow

Table 25Types of Errors and Possible Attacks

Severity Level	Critical	Warning	Low
Number of error	2	40	23

Table 26 Vulnerabilities Classification

4.3 Applying the Attack Trees Methodology

The main purpose of applying this method to analyse MIS security is to model attacks against the platform. As mentioned in chapter three, this approach helps to identify both design flaws and implementation errors.

Every attack on a system has a goal. In this thesis work, the goal is to attack the modem. The idea is to prove how possible it is to attack the modem using vulnerabilities in MIS components. To perform the attacks, use is made of well documented vulnerabilities and attacks against software systems. This information is available at various sources including National Vulnerabilities Database (NIST), SANS Institute, papers [21]. The various attacks obtained from these sources are classified according to identified security objectives of the target component.

4.3.1 Selecting Key Assets

The attack tree analysis is based on the main assets identified in the preceding section. The analysis starts by choosing some vital assets as the focal point of the analysis. These assets are selected taking into account the platform –supported services and the flow of information among the components. The key factor to consider in this case is what critical components affect the modem and what happens if they are attacked or are not functioning.

4.3.2 Modeling the Attacks

As mentioned in chapter two, the MIS consist of a number of components. However, in analysis of this nature, the asset to protect and the target asset should be identified. In this Master thesis project, the modem is the asset to protect while the RIL is the target asset.

The RIL interfaces Android components and plays very important roles in bridging the modem to Android applications in the upper layers. Therefore, it is more susceptible to attacks. Thus modeling this attack is focused on the RIL and its interaction with other MIS components.

Figure 7 presents an attack tree describing possible ways to attack the modem using vulnerabilities in MID, RIL or RFM. As there is no strict rule as to how to model attack trees [19, 34], top-bottom approach to modelling is used in this analysis. Thus the modem represents the main goal of the attack. The second level represents the main MIS components that link the modem and can be used to attack the modem. However, since this research focuses on RIL and its interaction with other components, the analysis takes into account possible attacks against this component. Thus, subsequent attacks are modelled to violate security objectives of RIL as the target asset.

The third level in the attack represents the security objectives that the attacker intends to violate. The forth level in the attack tree represents the possible mechanisms an attacker can use to launch his attack on the modem. In the following sections, sub attack trees on each of the attack mechanisms are presented.

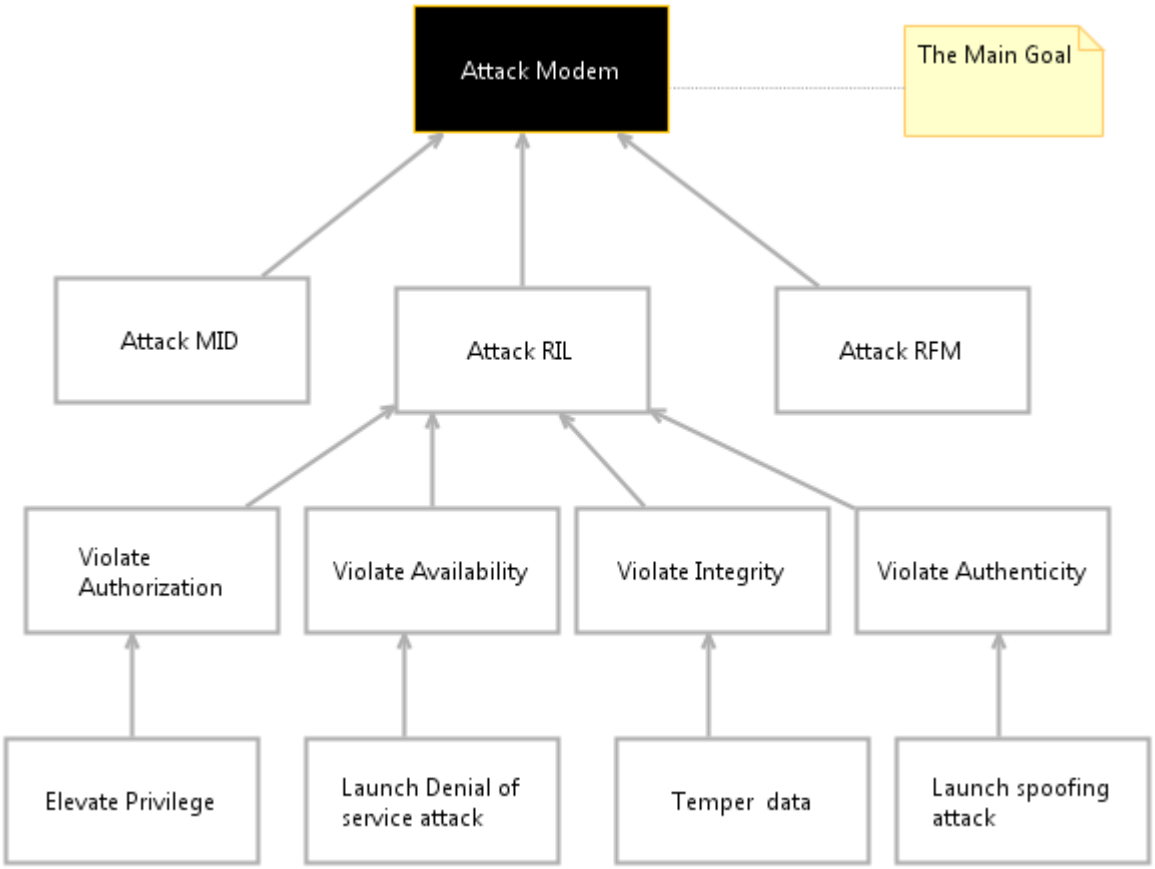

Figure 7: An attack tree showing possible attacks against the Modem

Elevation of privilege is one of the mechanisms that can be used to attack the modem. This attack is intended to violate authorization and is further developed in the following section. Figure 8 depicts the possible attacks against the RIL using elevation of privilege.

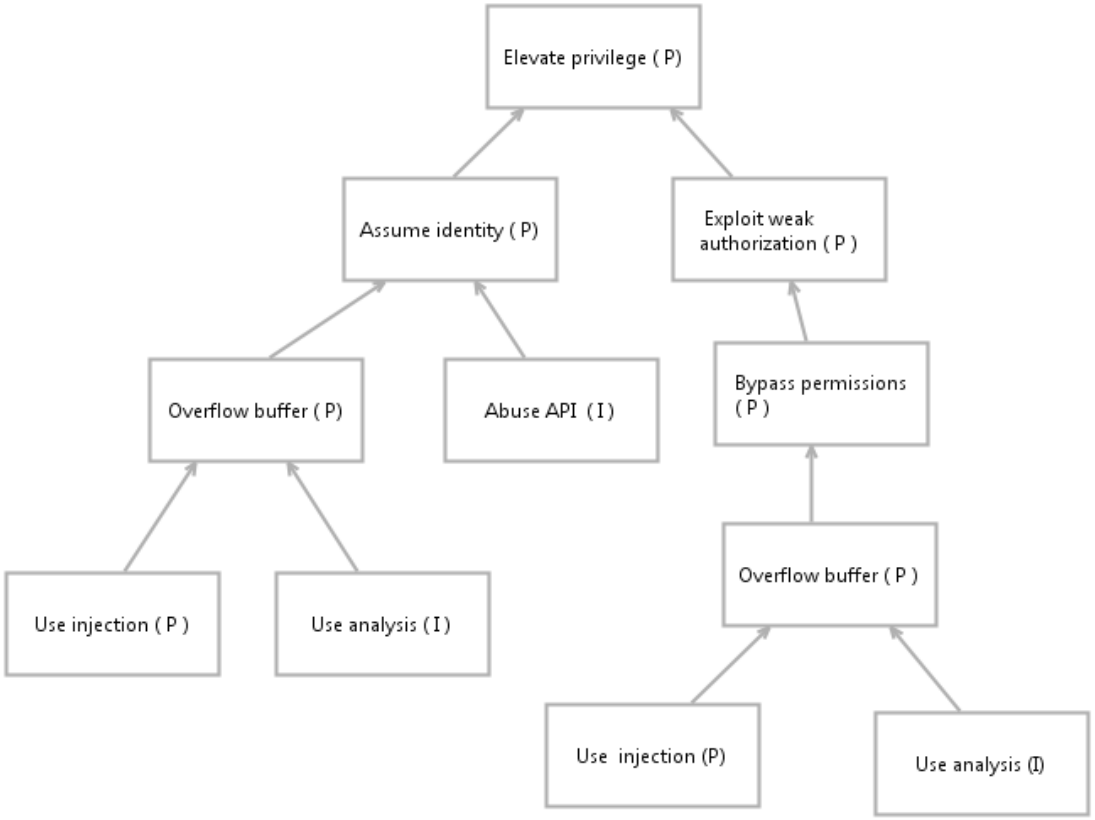

Figure 8: An attack tree showing possible attacks against RIL using elevation of privilege mechanism.

The second means by which the modem can be attacked is to use Denial of service attack. This attack is intended to violate availability objective of the RIL thus the modem. Figure 9 shows various ways to implement this attack against RIL component.

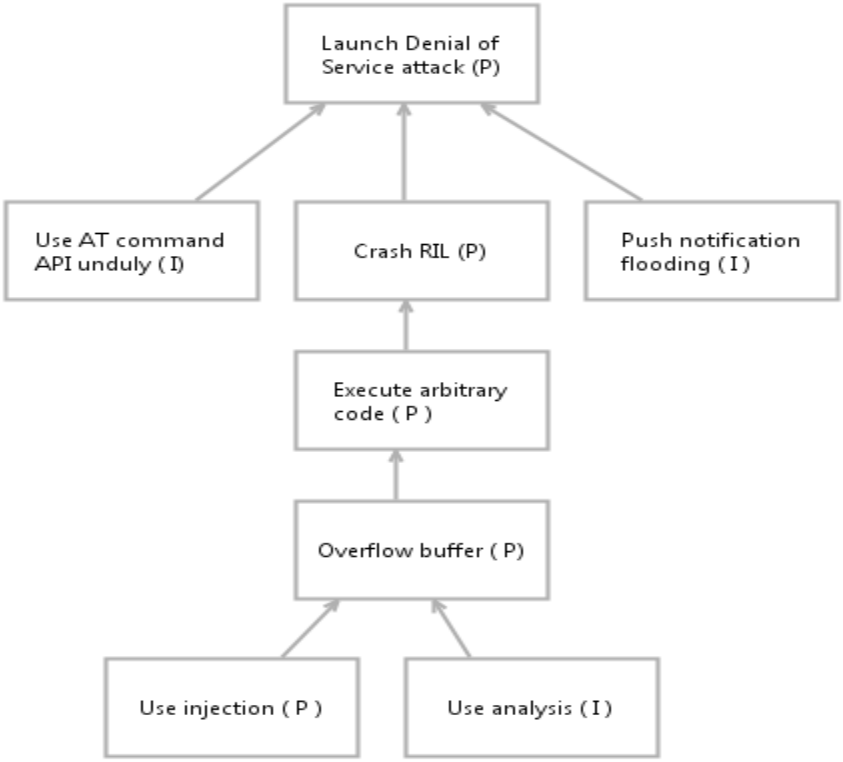

Figure 9: An attack tree showing Denial of Service attack against RIL

The third mechanism that can be used to attach the modem is to violate the integrity of the RIL. Tempering data in MIS violates the integrity objective of the target asset. Figure 10 presents possible means of carrying out this attack using vulnerabilities in the RIL component and its environment.

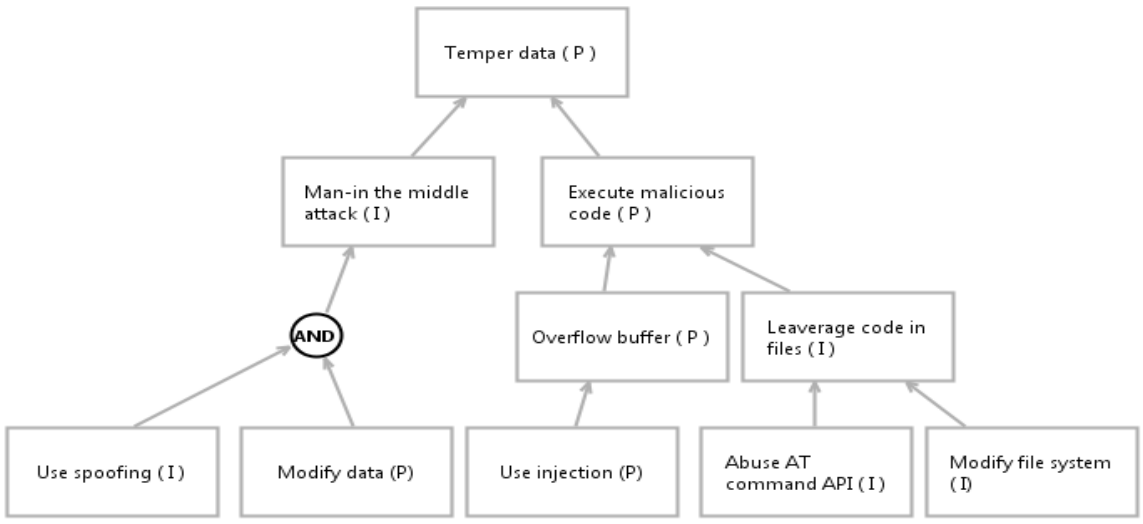

Figure 10: An attack tree depicting possible attacks against the Modem

The fourth attack against the modem intends to violate authenticity objective of the target component by using spoofing mechanism. Figure 11 shows how this attack against the system can be carried out.

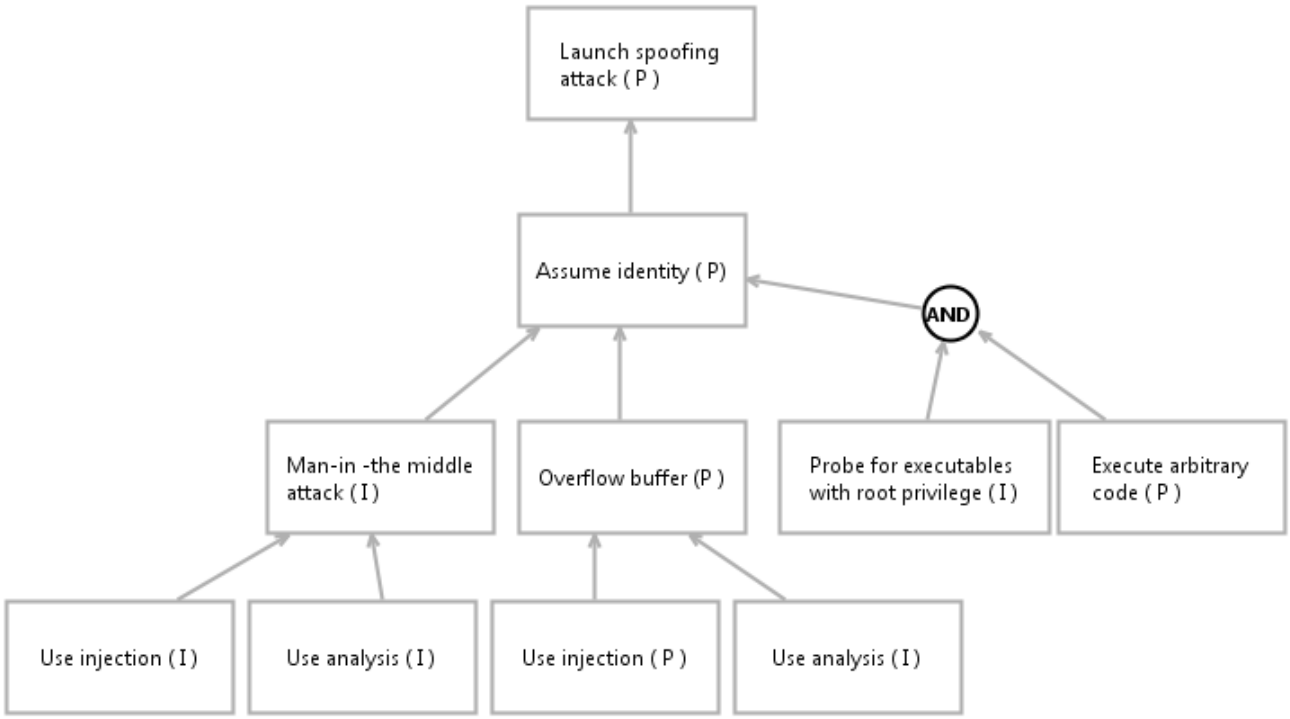

Figure 11: An attack tree showing possible attacks against the RIL using spoofing attack.

4.3.3 Identified Vulnerabilities

As mentioned in chapter [2. 3], an attack tree can be used to determine vulnerability of a system to an attack. In this analysis, Boolean values are used to determine vulnerability of RIL to a particular type of attack. The presence of possible value (P) in the attack trees indicates vulnerability while impossible value (I) shows that the component is not vulnerable to that particular attack. In this section, vulnerabilities identified from the various attack trees are presented.

A close study of the attack trees reveals the following vulnerabilities:

- An attacker can exploit buffer overflow condition in RIL in order to elevate privileges
- An attacker can use weakness in the permission system to elevate privileges
- It is possible to launch Denial of service attack on the platform using buffer overflow condition in RIL
- An attacker can overflow buffer in order to execute malicious code. This can lead to data modification.
- It is possible for an attacker to assume identity in order to violate access control.

In the following section, detailed discussion of the vulnerabilities and their implications is made.

CHAPTER 5 Discussion

In this section, vulnerabilities identified from the three analyses are discussed. Additionally, possible mitigation strategies based on the vulnerabilities are presented. The information on the vulnerabilities and their mitigation strategies are based on what is in the Common Weakness Enumeration document. Further information on these vulnerabilities is available in NIST's National vulnerability database version 2.2[21].

5.1 Vulnerabilities Identified in TVRA Analysis

The vulnerabilities identified in MIS during TVRA analysis are essentially design flaws. From the analysis, these vulnerabilities are not likely to be exploited. But, the risk associated with exploiting them is major. Therefore, mitigation strategies must be introduced to prevent any incident. In this section, discussion on the vulnerabilities and mitigation strategies is made.

Incorrect permission assignment for critical resource

This type of vulnerability exists in programs when permissions are set on critical resources in a manner that attackers can read or modify the resource. The platform under analysis is designed in a way that the file system is stored in the Host operating system file system. This means that the Host system must ensure integrity of the file. But, these files are not integrity protected. An attacker using confused deputy problem method might modify data in the files. The result of this will be data loss or code execution which will cause the modem to malfunction. To prevent this, it is advisable to ensure that the configuration file, executables and libraries are made read only. But the software's administrator can have write permission.

Missing encryption of sensitive data

Leaving sensitive data unprotected leads to security vulnerability. The calibration data software configuration data and other modem related data in the MIS are not integrity protected. An attacker might corrupt the unprotected data on the mobile platform. However, it is practically not possible to modify the data using malicious applications if the data is encrypted with the right cipher mode. Therefore, the mitigation strategy to this vulnerability is to ensure that encryption is part and parcel of the system design.

Privilege dropping error

This error occurs when a software component with root privileges fails to drop the privileges before handing out resources to other component which do not have the same privileges. MID in MIS starts as root and drops the root privileges during start up and

become a system user, However, it uses the privilege to open resources before dropping them. As discussed in chapter [4.1.1], the MID controls the modem and also relates with RIL over the D-Bus. RIL uses the open resources to relate with MID and the modem. It is possible that the privilege will spread to other components including RIL. An attacker might exploit vulnerability in RIL to gain privileges. This will result in code execution, reading of data and possibly disabling of services. Principle of separation of privileges can be used to mitigate this type of vulnerability. This mitigation strategy requires multiple conditions to be satisfied in order to access resources. It is also good to protect all communication channels that could be used to interact with MID.

5.2 Vulnerabilities Identified in Static Code Analysis

The vulnerability reports generated from analysing the source code are different. However, the vulnerabilities detected in this analysis can be classified into categories. In this section, discussion on the vulnerabilities detected in the source code is presented.

Unsafe functions

Analysis of the RIL code detected unsafe function errors in the code. These errors exist in the program because functions that are classified as potentially dangerous have been used during implementation. These functions introduce vulnerability into the programme. An attacker who intends to launch memory corruption attack can take advantage of errors of this nature. To prevent these types of vulnerabilities, a list of banned API functions should be provided and advice developers from using them. Additionally, it is advisable to automate static code analysis tools to check for these dangerous functions. Furthermore, an instruction can be given to the compiler to display utilization of these functions.

Weak credentials

The code analysis also revealed the presence of weak credentials vulnerability in the source code. This vulnerability exists because a component does not make the use of strong password requirement for users. In this case, users are permitted to use weak passwords. With this vulnerability, it is easier for an attacker to compromise user accounts. It is also possible for an attacker to exploit this vulnerability in order to gain privileges or assume identity. It is a good design to require users to use strong passwords. Absence of password complexity makes it easier for an attacker to guess password of users. To prevent occurrence of this type of vulnerability, strong password

usage should be enforced. For instance, one can design a password policy that will contain the following attributes:

- Minimum and maximum length of the password
- Requirements on mixed character sets
- No use of user name
- Introduce password expiration
- Not allowing reuse of password

Use after free

The analysis shows existence of use after free errors in the code. The errors exist in the program because memory has been referenced after it is freed back to the memory management system. This vulnerability could result in a program crash, unexpected values, or even arbitrary code execution. An attacker could exploit this condition to cause unavailability of RIL which will affect the Modem. The best strategy to avoid this problem is to use a language that will offer automatic memory management. It is also a good practice to ensure that when freeing pointers, they are sent to null. But, the use of multiple or complex data structures will render this strategy ineffective.

Double free

Double-free vulnerabilities exist in the program. The cause of this condition is that a dynamically allocated memory is freed twice. A vulnerability of this nature can lead to memory corruption and perhaps a buffer overflow condition. An attacker could use this condition to crash the RIL in an attempt to launch denial of service attack on the platform. The mitigation strategy to this problem is to ensure that each allocation if freed just once. It is also a good practice to set the pointer to null after a chuck has been freed. This will ensure that the pointer cannot be freed more that once.

Other Buffer related errors

The other buffer related errors detected in the analysis are memory copy, memory leak, null pointer error and fixed size local buffer. The existence of fixed buffer in a program may result in a serious problem. The buffer may be too small to contain data during program execution. This condition could lead to buffer overflow. An attacker could exploit this condition to cause denial of service attack, execute arbitrary code or even

cause data loss. To avoid this problem, it is good to perform input validation on any numeric input in order to ensure that it is within the correct range. It is a good practice for developer to ensure that inputs satisfy both minimum and maximum conditions for the anticipated range.

A null pointer dereference condition in the program is an error. This error can lead to program crash or exit. Usually, a number of flaws such as race condition and programming omissions can cause this error in a program. An attack can exploit this vulnerability to crash the program thus causing denial of service attack or even execute unauthorized code. This problem can be avoided by checking the results of all functions that return a value and ensure that the value is not null. Another approach to solving this problem is to use a programming language that is not vulnerable to this problem.

Memory leak problem exist in the source code analysed. This error occurs when a program uses assigned memory and does not release it well after use. The effect of this condition is the consumption of memory. The problem can be caused by error conditions and other special conditions. Generally, this type of error results in software reliability problem. Programmers who use C++ language can avoid this vulnerability by using smart pointer class specifically, std::auto_ptr. This class is defined by ISO/IEC 14882:2003[21]. Further information on this class is available in the Common Weakness Enumeration document which is obtainable from NIST's National Vulnerability database version 2.2.

Memory copy problems in software lead to buffer overflow. This condition occurs when a program copies an input buffer to an output buffer without ensuring that the size of the output buffer is more than the input buffer. An attacker can use buffer over flows to execute arbitrary code. In order to avoid this problem, a developer can run or compile programs with features that automatically offer means of protection that mitigates buffer overflow problems.

5.3 Vulnerabilities Identified in Attack Trees Analysis

This section presents discussion on the vulnerabilities identified from attack trees analysis. Analysis of the privilege elevation attack tree reveals that it is possible to escalate privilege. In this analysis, an attacker exploits weakness in android permission system to escalate privilege. The attacker using permission re-delegation can bypass MIS permission system in order to escalate privilege. To achieve this, the attacker will use injection method to exploit buffer overflow condition in RIL in order to elevate privilege. It is also possible for an attacker to assume identity using the same vulnerability in RIL. This will also result in privilege elevation.

Analysis from Denial -of -Service attack tree indicates that it is possible to crash RIL in order to launch Denial- of -service attack on the platform. An attacker can use injection method to overflow buffer and execute arbitrary code. This will result in crashing the RIL, making Denial -of -service attack possible.

The attack trees analysis also shows that it is possible to flout integrity objective of RIL. An attacker who intends to modify data can execute malicious code using buffer overrun condition in RIL. This will affect the operation of the modem.

Furthermore, this analysis indicates that an attacker can violate authenticity. To achieve this goal, the attacker uses injection method to overflow buffer in RIL and assume identity. The attacker may send any AT command to the modem or carry out any operation defined by the AT interface.

In the attack trees analysis, the possible attacks exploit two main vulnerabilities in the platform namely, buffer overflow condition in RIL and weakness in Android permission system. However, weakness in Android permission system is Android specific problem which is not investigated in this master thesis. To prevent buffer overflow attack from taking place, the following mitigation strategies can be adopted.

- Use of a compiler that automatically checks bounds
- Using secure or safe functions that will not lead to buffer overflow condition
- Using two or three static analysis tools to detect buffer overflow conditions in the software.

Chapter 6 Conclusion and Further Work.

The goal of this master thesis has been to perform security analysis on the Android mobile platform developed by ST-Ericssson. The main objectives of this analysis have been to detect security vulnerabilities in the MIS and the interfacing modem and to propose possible solutions to the identified problems to prevent unauthorized access to resources or modem damage.

The challenge was how to approach this analysis in order to detect security vulnerabilities in the MIS that will affect the normal operation of the modem.

MIS is a software component designed by ST-Ericsson to bridge their modem in Android mobile platform. The MIS consist of many components. However, the main component integrating the modem in Android platform is the Radio Interface Layer (RIL). Therefore, this analysis focused on identifying vulnerabilities in the RIL and its environment.

To analyse security in MIS and the interfacing modem, three security analysis methodologies have been used. TVRA methodology was employed to detect vulnerabilities due to design flaws in the system. Static code analysis was used to detect implementation errors that make the platform vulnerable to attacks. To graphically show possible attacks against the modem and the relations among the attacks, attack trees methodology has been used.

Performing the analysis using these methodologies made it possible to identify vulnerabilities in the RIL and its environment. Applying TVRA method to analyse MIS security revealed some design flaws in the platform. The code analysis results indicate the implementation errors in the source code. The attack trees analysis shows possible means to attack the modem using vulnerabilities in MIS.

Further analysis performed using TVRA methodology shows that the identified vulnerabilities due to design errors are not likely to be exploited. However, the risk associated with an attack on the platform using these vulnerabilities is major. Static code analysis and attack tree analysis revealed vulnerabilities that are exploitable. Based on the results obtained in the analysis, some mitigation strategies were proposed.

This analysis and the results obtained show suitability of the methods used to analyze software systems for security vulnerabilities. Particularly, TVRA method can be applied to analyze design errors while code analysis can be used to detect implementation errors. Attack trees analysis can be employed to identify both design and implementation errors.

The results obtained from the analysis and the proposal of mitigation strategies to improve on MIS security are the main contributions from this Master thesis work. Regarding further work, it is worth mentioning that the MID being the main component in the MIS that controls the modem needs to be analysed in order to establish complete security status of the modem.

References

[1] William Enck, Machigar Ongtang and Patrick McDaniel, Understanding Android Security. Pennsylvania State University [online]:

[2] Jon Oberheide. A Look at a Modern Mobile Security Model: Google's Android Platform

(March, 2009) University of Michigan [online]:

[3] Divya Muthukumaran, Anuj Sawani, Joshua Schifffman, Trent Jaeger and Brian M. Jung. Measuring Integrity on Mobile Phone Systems. Systems and Internet Infrastructure Security Laboratory Pennsylvania State University, Secure Systems Group; Sumsung Electronics Co. Ltd,Suwon-city, Gyeonggi-Do, Korea, 443-742.

[4]Michael Grace, Yajin Zhou, Zhi Wang and Xuxian Jiang. Systematic Detection of Capability Leaks in Stock Android Smartphones. North Carolina State University
890 Oval Drive, Raleigh, NC 27695

[5]Jesse Burns. iSEC PARTNERS. MOBILE APPLICATION SECURITY ON ANDROID, Context on Android security, (June, 2009) Version 0.1

[6] ST- ERICSSON platform Description, Host Design Specification.

[7] Michael Becher, Felix C. Freiling, Johannes Hoffmann, Thorsten Holz, Sebastian Uellenbeck and Christopher Wolf. (2011). Mobile Security Catching Up? Revealing the Nuts and Bolts of the Security of Mobile Devices. *IEEE Symposium on Security and Privacy*.[Online].Available:http://www.ieee
security.org/TC/SP2011/PAPERS/2011/paper007.pdf

[8] Andreas Sjöström, Kazuhide Fukushima, Shinsaku Kiyomoto, Wook Shin and Toshiaki Tanaka. (2010, May). Security Analysis of Access Linux Platform. *IJCSNS International Journal of Computer Science and Network Security*. [Online]. VOL.10 No.5. Available: http://paper.ijcsns.org/07_book/201005/20100503.pdf

[9] Lookout. (2011, August). Lookout Mobile Threat Report. Lookout Mobile Security [Online]. Available: https://www.mylookout.com/_downloads/lookout-mobile-threat-report-2011.pdf

[10] *ETSI Technical Specification for Treat, Risk and Vulnerability Analysis*, ETSI TS 102 165-1 V4.2.3 (2011-03).

[11] Anders Orsten Flaglien, Attack Trees For Risk Analysis, *Gjøvik University College*,[Online]. Available:http://www.andersof.net/wp-content/uploads/attack_trees_for_risk_and_threat_analysis08x.pdf

[12] Sjouke Mauw and Martijn Oostdijk, Foundations of Attack Trees, Eindhoven University of Technology, Radboud University Nijmegen. [Online] Available: http://www.win.tue.nl/~sjouke/publications/papers/attacktrees.pdf

[13] Bruce Schneier. (1999, December). Modeling security threats. Dr. Dobb's Journal December 1999. [Online]. Available: http://www.schneier.com/paper-attacktrees-ddj-ft.html.

[14] Barbara Kordy, Sjouke Mauw, Saša Radomirović and Patrick Schweitzer. Foundations of Attack–Defense Trees. CSC and SnT, University of Luxembourg, 6, rue Coudenhove–Kalergi,L–1359,Luxembourg.[Online].Available: http://satoss.uni.lu/members/barbara/papers/adt.pdf

[15]Brian Chess, Cary McGraw. Static Analysis for Security. IEEE Security & Privacy.(November/December, 2004) [Online]: http://buildingsecurityin.us-cert.gov/bsi/resources/414-BSI/119-**BSI**.html

[16] Lucas Davi, Alexandra Dmittrienko, Ahmad-Reza Sadeghi and Marcel Winandy. Privilege Escalation Attacks on Android. Ruhr-University Bochum, Germany, Technische University Darmstadt, Germany.

[17]Alexandru G. Barddas. Static Code Analysis. Kansas State University, United State of America.

[18]Panagiotis Louridas. Static Code Analysis. (July/August, 2006) IEEE Software

[19] Anderson Morais, Eliane Martins, Ana Cavalli and Willy Jimenez. Security Protocol Testing Using Attack Trees. (2009). State Universsity of Campinas, Telecom and Management Sud-Paris.

[20]Andy German. Software Static Code Analysis Lessons Learned. (November, 2003) Qinetiq Ltd.

[21] National Vulnerability Database Version 2.2. http://nvd.nist.gov/

[22] Dr. Natarajan Meghanathan, Risk Analysis, Mitigation and Management, Jackson State University, Jackson, MS 39217
http://www.jsums.edu/cms/tues/docs/SoftwareSecurity/Module-RiskAnalysis.pdf

[23] Vesely, W.E., Goldberg, F.F., Roberts, N., Haasl, D.: Fault Tree Handbook. Technical

Report NUREG-0492, U.S. Regulatory Commission (1981)

[24]http://www.netmite.com/andriod/mydriod/dev. Accessed on 10th May, 2012.

[25] Bruce Schneier, Attack Trees, Counterpane Systems (October, 1999). [Online]. http://web.eecs.utk.edu/~dunigan/cns04/attacktrees.pdf

[26] Egil Trygve Baadshaug, Gencer Erdogan and Per Håkon Meland. Security Modeling and tool support advantages. The Norwegian University of Science and Technology. (2010).

[27] Harsha Bagar. Integrating the Modem. Teleca white paper. (June, 2011)

[28] Machigar Ongtang, Stehpen McLaughlin, William Enck and Patrick McDaniel. Semantically Rich Application-Centric Security in Andriod. Depertment of Computer Science and Engineering, Pensylvania State University.

[29] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev and Chanan Glezer. Google Andriod: A Comprehensive Security Assessment. Ben-Gurion University of Negev, Israel.

[30] Aubrey-Derick Schmidt, Hans-Gunther Schmidt, Jan Clausen, Ahmet Camtepe and Sahin Albayrak. Technische University Berlin, Germany. Kamer Ali Yuksel and Osman Kiraz. Sabanci University Istanbul, Turkey. Enhancing Security of Linux-based Andriod Devices.

[31] Aivo Ju¨rgenson. Efficient Semantic of Parallel and Serial Models of Attack Trees. Tallinn University of Technology. (May, 2010) Dissertation.

[32] Michael V. Scovetta. “Yasca users guide.” URL: http://www.scovetta.com/yasca/yascamanual.

Pdf Accessed on 24th April 2012

[33] "The Open Web Application Security Project." URL: https://www.owasp.org/index.php/Category:OWASP_Yasca_Project Accessed on 24th April 2012
[34] Jeanne H. Espedalen. Attack Tress Describing Security in Distributed Internet-Enabled Metrology.(2007) Gjøvik University College.

[35] David A. Wheeler. "flawfinder." URL: http://www.dwheeler.com/flawfinder/ Accessed on 20th April 2012
[36] "RATS." URL: https://www.fortify.com/ssa-elements/threat-intelligence/rats.html Accessed on 21st April 2012
[37] "Cppcheck - A tool for static C/C++ code analysis." URL http://sourceforge.net/apps/mediawiki/cppcheck/index.php?title=Main_Page#Memory_leaks_.28address_not_taken.29 Accessed on 17th April 2012

Appendices

Appendix A Glossary and Abbreviations

Appendix B Tables

Table 1: MIS Security Environment

A Security Environment	
a.1 **Assumptions**	
TVRA- ID	**Summary**
a.1.1	MIS is used to provide bridge solution to ST-Ericsson Modem to be integrated to a Linux base Host system
a.1.2	MIS runs on Android Platform with a Linux Kernel
a.2 **Assets**	
a.2.1	The Modem provides platform services to the external Host system
a.2.2	MID resides in MIS and starts, stops and monitor the Modem. It also coordinates all service –related functions of the Modem.
a.2.3	RIL resides in MIS and initialises the Modem
a.2.4	RFM acts as a file server for the modem
a.3 **Threats**	
a.3.1	Loss or corruption of data
a.3.2	Unauthorized access
a.3.3	Denial of Service
a.3.4	Masquerade
a.4 **Threat Agents**	
a.4.1	Modification of resource, Injection
a.4.2	Assume identity, Gain privileges
a.4.3	Injection , Analysis
a.4.4	Spoofing

Table 2: Security objective for Target Asset and Environment

B Security Objectives	
b.1	Security objectives for RIL
b.1.1	The RIL should ensure the availability of the services provided by the modem
b.1.2	The RIL should ensure the service is accessible and usable on demand by an authorized entity or program
b.2	Security Objectives for MIS
b.2.1	The MIS should ensure availability of the Modem
b.2.2	The MIS should ensure integrity of Modem-related data and operations

Table 3: Security Requirements for Target Asset and Environment

C Security Requirements	
c.1 RIL Security Functional Requirements	
c.1.1	The RIL shall initialise the Modem to provide platform services
c.1.2	The RIL shall ensure secure communication between the Modem and Android telephony framework.
c.1.3	The RIL shall relate with MID to keep track of the Modem
c.2 MIS Security Requirements	
c.2.1	The MIS shall ensure integrity of Modem's data
c.2.2	The MIS shall ensure integrity protected link between the Modem and the Host system
c.2.3	There shall be authentication between modules and the Modem

Table 7: Attack Potential for Modification of data

Factor	**Range**	**Value**
Time	≤ 3 months	13
Expertise	Expert	5
Knowledge	Public	0
Opportunity	Unnecessary/ Unlimited access	0
Equipment	Specialised	3

Table 8: Attack Potential for Assuming Identity/Gaining Privileges

Factor	**Range**	**Value**
Time	≤ 3 months	13
Expertise	Expert	5
Knowledge	public	0
Opportunity	Unnecessary/ Unlimited access	0
Equipment	Specialised	3

Table 9: Attack Potential for Reading of files

Factor	**Range**	**Value**
Time	≤ 3 months	13
Expertise	Expert	5
Knowledge	public	0
Opportunity	Unnecessary/Unlimited Access	0
Equipment	standard	0

Table 10: Intensity Factor

Attack	Attack Intensity	Value
Modification of data	Moderate level of multiple instances	1
Assuming Identity/Gaining Privileges	Moderate level of multiple instances	1
Reading of files	Moderate level of multiple instances	1

Table 11: Vulnerability rating for incorrect permission assignment for critical resource

Attack Potential Values	**Resistant to Attacker with Potential Value of:**
21	High
21	High

Table 12: Vulnerability rating for privilege dropping error

Attack Potential Values	**Resistant to Attacker with Potential Value 0f:**
21	High

Table 13: Vulnerability rating for missing encryption of sensitive data

Attack Potential Values	**Resistant to Attacker with Potential Value of:**
18	High
21	High

Table 17: Impact on the Platform

Asset Impact	Attack Intensity	Resulting Impact
3	1	3
3	1	3
3	1	3