

LOCATION AWARE SECURITY SYSTEM FOR MOBILE DEVICES

Dinusha Nivanthaka Amerasinghe

(118201P)

Thesis submitted in partial fulfillment of the requirements for the degree Master of
Science



University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Department of Computer Science & Engineering

University of Moratuwa

Sri Lanka

May 2015

LOCATION AWARE SECURITY SYSTEM FOR MOBILE DEVICES

Dinusha Nivanthaka Amerasinghe

(118201P)

Thesis submitted in partial fulfillment of the requirements for the degree Master of



University of Moratuwa, Sri Lanka.
Science
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Department of Computer Science & Engineering

University of Moratuwa

Sri Lanka

May 2015

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

Name: D.N. Amerasinghe

Index No: 118201p



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

I certify that the declaration above by the candidate is true to the best of my knowledge and that this report is acceptable for evaluation for the CS6998 M.Sc. Research Project.

Signature of the supervisor: Date:

Name: Dr. Malaka Walpola

ABSTRACT

Smart phones and related mobile computing device usage is increasing exponentially. Prices of these devices are going down, making them attractive to a larger audience. People tend to prefer mobile computing devices since due to their inherent facilitation of mobility, advanced features, connectivity etc. These devices make the possibility of ubiquitous computing a reality. As discussed in this report there are evidence that mobile computing devices are actually a convergence of various discrete devices such as GPS Receivers, Personal Digital assistants (PDA's) etc. These devices have their own pros and cons compared with traditional computing devices. One of the main shortcomings of technology for these devices is security, due to the inherent mobile nature of these devices, the security controls that apply to PC's aren't suitable for these types of devices. Malware targeting mobile devices are very complex, they tend to exploit a number of shortcomings of these devices. The currently available technology has shortcomings on addressing these security requirements. Furthermore current malware detection technology is new and evolving. Traditional approaches such as a virus scanners doesn't work in these environments due to power, processing and other constraints.

This report analyses the approaches available to implement a location context aware security solution. An analysis of current research in this area is conducted. Several implementations of different malware detection systems, security policy management systems and location context aware systems are discussed in order to evaluate their feasibility of approach and effectiveness of the solutions. Furthermore our analysis includes the discussion of current malware behavior in the Smartphone base. Android is chosen as the platform for the implementation since it is widely deployed on many Smartphone and other mobile computing devices. Other platforms such as Apple iOS, Symbian and Blackberry features similar architectures and hence the concept discussed in this report applies to those platforms as well.

In this report we analyze the feasibility of implementing a middleware based approach to provide a location context aware security solution. In order to select the appropriate approach several malware detection methods were analyzed.

ACKNOWLEDGMENTS

I take this opportunity to express my sincere gratitude to my supervisor, Dr. Malaka Walpola, for his invaluable thoughts, encouragement, supervision and guidance throughout this research work. He offered his generous guidance every time I reached him even during his busy time schedules.

Further, I use this opportunity to thank my MSc lecturers who guided me throughout my MSc (Computer Science) course and shared their valuable knowledge with dedication.

Finally, I express my gratitude to my parents and my friends for the support and encouragement throughout my life to bring me up to this level.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

TABLE OF CONTENTS

| | |
|--|-----|
| Abstract | i |
| List of Figures | vi |
| List of Tables | vii |
| Chapter 1 - Introduction | 1 |
| 1.1 Background | 1 |
| 1.2 Problem Domain | 2 |
| 1.3 Objectives | 2 |
| 1.4 Proposed Solution | 4 |
| 1.5 Overview of the Dissertation..... | 4 |
| Chapter 2 - Literature Review | 6 |
| 2.1 Introduction | 6 |
| 2.2 Context Aware Systems | 6 |
| 2.3 Context Aware Security Systems for Mobile Devices..... | 8 |
| 2.3 Security Models of Smartphone Platforms | 11 |
| 2.4 Security Mechanisms for Android Mobile Devices | 18 |
| 2.4.1 Security Refinements of Android SDK | 25 |
| 2.5 Threat Models and Malware for Mobile Devices | 27 |
| 2.6 Malware Detection Techniques for Mobile Devices..... | 31 |
| 2.6.1 Detection in Mobile Device Only | 31 |
| 2.6.2 Detection at a Separate Server | 35 |
| Chapter 3 - Design and Implementation | 37 |
| 3.1 The Proposed Software Architecture | 37 |
| 3.1.1 The Server..... | 39 |
| 3.1.2 The Client | 39 |
| 3.2 The Implemented Solution | 40 |
| 3.3 The Client Software | 41 |
| 3.3.1 Main Activities | 41 |
| 3.3.2 Class Structure | 45 |
| 3.3.3 Data Storage..... | 46 |
| 3.3.4 The Background Service..... | 47 |
| 3.3.5 Location based Security Policy Management | 47 |

| | |
|--|----|
| 3.3.6 Application Security Audit | 48 |
| 3.3.7 Communication Method | 48 |
| 3.3.8 The Strace Utility..... | 48 |
| 3.3.9 The Client Software User Interfaces..... | 49 |
| 3.3.10 Background Services | 49 |
| 3.3.11 App Tracing | 50 |
| 3.3.12 Location Services..... | 50 |
| 3.3.13 App Security | 50 |
| 3.4 The Server Software | 51 |
| 3.4.1 Data Storage..... | 53 |
| 3.4.2 Behavior Rule format | 53 |
| CHAPTER 4 - Testing and Results | 54 |
| 4.1 Malware Behavior Rules | 54 |
| 4.2 Location Rules..... | 56 |
| 4.3 Results | 58 |
| 4.3.1 Malware Detection System..... | 58 |
| 4.3.2 Location Aware Security System | 59 |
| CHAPTER 5 - Conclusions and Future Work | 62 |
| References | 64 |



University of Moratuwa, Sri Lanka.
 Electronic Theses & Dissertations
www.lib.mrt.ac.lk

LIST OF FIGURES

| | |
|--|----|
| Figure 2.1: Location based Architecture (Source: Ardagna et al. [12])..... | 8 |
| Figure 2.2 - L4Android Architecture. (Source: M Lange [19])..... | 16 |
| Figure 2.3 - Hosted hypervisor and baremetal hypervisor. (Source: Gudeth et al. [20]) | 17 |
| Figure 2.4: Android usage control framework (Source: Bai et al. [21])..... | 19 |
| Figure 2.5 : Overhead of running ConUCON. (Source: Bai et al. [21])..... | 20 |
| Figure 2.6: Multi-level approach for performance efficient taint tracking within a common smartphone architecture. (Source: Enck et. al. [25])..... | 25 |
| Figure 2.7 - "Andromaly" framework overview. (Source: Shabtai et al. [28]) | 32 |
| Figure 2.8 - "Crowdid" Framework Overview. (Source: I Burguera and S Nadjm- Tehrani [29]) | 36 |
| Figure 3.1 - The proposed architecture of the "AndroSec" system | 38 |
| Figure 3.2 - Activity diagram for location context detection in the client..... | 42 |
| Figure 3.3 - Activity diagram for application log/trace collection in the client..... | 43 |
| Figure 3.4 - Activity diagram for application log upload to a remote server from the client..... | 44 |
| Figure 3.5 - Class diagrams of main classes | 46 |
| Figure 3.6 - Entity Relationship(ER) diagram of client database | 47 |
| Figure 3.7 - Pseudo code for location based security policy management..... | 47 |
| Figure 3.8 - Pseudo code for application security audit..... | 48 |
| Figure 3.9 - Application home screen..... | 49 |
| Figure 3.10 - Application system call trace collection | 50 |
| Figure 3.11 - Application interfaces | 51 |
| Figure 3.12 - Pseudo code for application security audit..... | 52 |
| Figure 3.13 - Server application trace processing..... | 52 |
| Figure 3.14 - Entity Relationship(ER) Diagram of Server Database..... | 53 |

LIST OF TABLES

| | |
|---|----|
| Table 2.1 : Examples of access control rules regulating access to a mobile network console. (Source: Ardagna et al. [12]) | 9 |
| Table 2.2: Smart phone permission modes. (Source: Kathy Au et al. [9]) | 14 |
| Table 2.3: Applications with duplicate permissions by market category. (Source: Vidas et al. [23]) | 22 |
| Table 2.4: Top ten duplicate permissions requested. (Source: Vidas et al. [23]) | 23 |
| Table 2.5: Malwares and their methods of infection. (Source: La Polla et al. [32]).. | 28 |
| Table 3.1: Behavior rule format..... | 53 |
| Table 4.1: Malware detection rules list..... | 55 |
| Table 4.2: Location Rules List..... | 56 |
| Table 4.3: Cumulative score for malware behaviors | 58 |
| Table 4.4: Results for real applications..... | 60 |



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

CHAPTER 1 - INTRODUCTION

1.1 Background

One of the basic features of the pervasive computing paradigm is weaving into everyday fabric of the people, so that it is indistinguishable [1]. One way to achieve this vision is to have computing devices that are small enough to be considered as mobile, and as powerful as the common computers when it comes to computation capabilities with very good connectivity to the Internet. Today mobile phones have a very high level of penetration [2] and has become a commodity. But, these devices lacked advanced features available in smart devices such as smart phones and tablets. Introduction of the Apple iPhone changed the perception of people's understanding about mobile devices [3]. Initially these devices were expensive. However, with the technological advances, the ever reducing price of Smartphones and mobile computing devices alike has made it affordable to everyone in the society. Open source mobile operating systems such as Android have reduced the cost further and due to the open source nature, encouraged third parties and mobile device manufacturers to customize them according to their will. Owing to the design of these devices it is possible to have a constant connection to the Internet, which is an essential feature of these devices. Most of the time this connectivity is complemented by Bluetooth and Wireless LAN connectivity. The popularity of these devices were fueled with the advent of the Social Media. More and more companies encourage their employees to use mobile devices in order to keep them connected with the company so that they can read emails, support customers etc., while on the move [1]. Mobile computing devices such as Smartphones possess a unique set of characteristics & challenges compared to their personal computing counterparts. A major such a characteristic is the security of mobile devices. People see their smart phones as being more personal than any other device; hence they tend to store sensitive information such as credit card details, user credentials on these devices [4].

1.2 Problem Domain

Due to the immense popularity of mobile devices, unfortunately these are becoming a favorite target of hackers and other types of attackers. Therefore, security has become a pivotal aspect in mobile computing and many are investigating the approaches of securing mobile devices. Smartphones need to adapt to varying environments. This is regarded as being context aware. Context-aware systems offer entirely new opportunities for application developers and for end users by gathering context data and adapting systems behavior accordingly. Especially in combination with mobile devices, these mechanisms are of high value and are used to increase usability tremendously [5].

1.3 Objectives

The main objectives of this research are,

1. Study and analyze the existing security mechanisms of mobile devices.
2. Propose a framework for these devices and provide a prototype

implementation.



University of Moratuwa, Sri Lanka.

Electronic Theses & Dissertations

www.lib.mrt.ac.lk

Current security mechanisms available to mobile devices resembles the traditional approaches provided by commercial vendors, these approaches does not take into consideration the context awareness in their approaches. However research in the area of security for mobile devices are continuing to increase and each approach proposed have its own pluses and minuses We will analyze each approach in the next chapter. Furthermore context-aware systems offer entirely new opportunities for application developers and for end users by gathering context data and adapting systems behavior accordingly. These mechanisms are of high value and are used to increase usability tremendously [5]. In order to explain the importance of location context aware security let us take an example scenario of an everyday user. John wakes up in the morning at his home/apartment. He picks up his Smartphone in order to read the news and taps on the screen. The agent in his smart phone check to see his location and concludes that john is at home, without further ado the agent shows john the home screen, on the background the agent scans the available Wi-Fi networks in

the vicinity. The agent match against a known list of trusted networks in order to provide suggestions to john. Meanwhile John can interact with the smart devices E.g. smart TV's at his home using Bluetooth and Wi-Fi of his smart phone. Furthermore he can purchase items using his home network. As John leaves his home and heads out, his phone immediately detects this and suggests to turn Bluetooth and Wi-Fi off, due to security reasons, and use GPRS or other connectivity methods provided only to user john. Furthermore the phone changes the wake up screen to a password protected one and reduces the sleep timeout. At a public place john searches for a free Wi-Fi. The security agent collects the details of the available networks and connects to the Internet using GPRS or other available method which is not public and search for known Wi-Fi hotspots at the current location. Based on the information received from the Internet the agent can advise to use a better Wi-Fi network. If possible the system should limit the interactions of the connected Wi-Fi, such as allowing credit card information and other information only go through secure HTTPS connections. To make things even more secure john may use the public networks to read news papers but not purchase something.

When John gets to office, the security agent identifies his location and relaxes the security restrictions described earlier. At office, he freely access office information resources, but does not want to access public Internet while he is in the lab doing sensitive data analysis. But during the coffee break at canteen, when he is not accessing sensitive office information, he may want to go online and check the sports news. Integrate some tasks like these to the above discussion. When John connects his phone with his computer/laptop, the security agent syncs the activity log of applications to the computer. This log in-turn is analyzed with the help of a security tools available in the Internet to identify malicious application behaviors and informs john about malicious behaviors and potential threats.

The above scenario summarizes the location context aware security aspects need to be considered.

1.4 Proposed Solution

The proposed solution functions as a context aware security system running on a mobile device. The functionality of the solution is greatly enhanced by the usage of a client-server solution. This is to address the limitations of power, processing capabilities and connectivity issues inherent in mobile devices. The client part of the solution runs on a mobile device. The server component runs on a remotely hosted application server. The client solution collects the application execution pattern traces and uploads into this remote server, Furthermore the client adapts the security of the device according to the current location. The remote server analyzes the uploaded application execution patterns (system call traces) and assigns a cumulative score to each application.

A prototype implementation is developed on Android. We focus on the Google Android Operating System, since it is currently the most widely deployed mobile operating system [6]. According to a Juniper networks report [7], malware for Android based mobile devices have seen a 400% increase since summer 2010. As of Android 2.3.3, there are currently 75 dangerous and normal permissions available to 3rd party developers, making Android OS the most complicated permission system [8].

1.5 Overview of the Dissertation

The content of this research is organized as chapters. A list of these chapters and a summary of their content is given below.

Chapter 2 – Literature Review

The approaches proposed by various authors with respect to context aware systems and mobile malware detection techniques are discussed in the literature review. This chapter presents an overview of the current situation with respect to security systems available for mobile systems as well as the prelude for the proposed solution.

Chapter 3 – Design and Implementation

This chapter describes the architecture of the proposed system; including authors approach, component modules and their interaction, algorithms and other considerations.

Chapter 4 - Testing and Results

Effectiveness of the prototype implementation is measured in this chapter. Authors use various testing methods and parameters in order to verify the correct and expected operation of the solution.

Chapter 5 - Conclusions and Future Work

Provides a summary of the work carried out as well as provides suggestions for further enhancements of the proposed solution.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

CHAPTER 2 - LITERATURE REVIEW

2.1 Introduction

Emerging ubiquitous or pervasive computing technologies offer ‘anytime, anywhere, anyone’ computing by decoupling users from devices. To provide adequate service for the users, applications and services should be aware of their contexts and automatically adapt to their changing contexts, which is known as context-awareness. A system is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context [8].

In order to address the proposed hypotheses the authors analyze areas that are most relevant for this specific research area. The authors try to cover as much breadth as possible. Since this research focuses primarily on Android, a number of papers discussed include Android only discussions.

2.2 Context Aware Systems

With regard to context aware systems the work by Hong et al. [8], carried out an extensive review and a classification of such systems from 2000 to 2007. Authors claim that research into context aware systems began after the year 2000. Hong et al. [8], define the term context-awareness as to provide adequate service for the users, applications and services should be aware of their contexts and automatically adapt to their changing contexts. This paper covers only journal articles. Other publication forms (The conference proceedings, unpublished working papers, master’s and doctoral dissertations, newspapers and books, etc.) were not included. Furthermore, LNCS (Lecture Notes of Computer Science) is excluded.

The classification framework developed by the authors, consists of the following five layers: concept and research layer, network layer, middleware layer, application layer and user intrastate layer. The concept and research layer involves overview, algorithm, development guideline, framework, context data management, evaluation and privacy and security categories. The network layer consists of protocol, sensing, network requirement and network implementation. Middleware layer is classified as

agent-based middleware, metadata based middleware, tuplespace based middleware, OSGI based middleware, reflective middleware, and sensor selection middleware. The user infrastructure layer is divided into interface and usability categories. Finally the application and service layer consists of smart space, tour guide, information systems, communication systems, m-commerce and web service.

Although there are various definitions for a context aware systems according to M. Baldauf [5] the most accurate definition is “*Any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves.*” proposed by Dey and Abowd [10].

M. Baldauf [5] has conducted a survey of context aware systems in 2007. Furthermore, the author discusses the sensor types available in order to detect attributes in the environment. The concepts that are of importance for our research include the design principles of context aware systems. Which are,

 **Direct Sensor access** - Ability to read Sensor Data Directly, which is the case for Mobile devices.
www.lib.mrt.ac.lk

Middleware Infrastructure - Introduces a layered architecture to context-aware systems with the intention of hiding low-level sensing details.

Context Server - This distributed approach extends the middleware based architecture by introducing an access managing remote component.

A context model is needed to define and store context data in a machine process-able form. Discussed approaches include,

Key-Value models - These models represent the simplest data structure for context modeling.

Markup scheme models - All markup based models use a hierarchical data structure consisting of markup tags with attributes and content.

Graphical models - The Unified Modeling Language (UML) is also suitable for modeling context.

Object oriented models - Modeling context by using object-oriented techniques offers to use the full power of object orientation (e.g., encapsulation, reusability, inheritance).

Logic based models - Logic-based models have a high degree of formality. Typically, facts, expressions and rules are used to define a context model.

Ontology based models - Ontologies represent a description of the concepts and relationships.

Although above concepts are generic in nature, they are applicable to mobile devices as well.

2.3 Context Aware Security Systems for Mobile Devices

Snekkenes [11] proposed a generalized access control model using the usual credentials and current location. The proposed solution considers confidence and timeout as parameters of policy specification. Fragments of a language intended for formulating personal privacy policies has been presented. Several examples illustrating the use of the language has been given.

Regarding location context aware security, Ardagna et al. [12] proposed an Access Control Policy Framework which supports location. This paper discusses an extensive approach to location based Access Control Systems. This research work addresses the limitations of the current technology used to ascertain the location of the requester. The authors describes location verification as a service using two service level agreement parameters, confidence and timeout. Figure 2.1 shows an overview of the functionality of the proposed concept.



Figure 2.1: Location based Architecture (Source: Ardagna et al. [12])

The proposed LBAC architecture involves the following three entities.

Requestor

Access Control Engine (ACE)

Location Service (LS)

Location-based conditions are expressed as,

$$\text{predicate(parameters)} \rightarrow [\text{range, accuracy, timeout}]$$

Boolean queries are expressed as,

$$\text{predicate(parameters, value)} \rightarrow [\text{bool value, confidence, timeout}]$$

Examples of access control rules regulating access to a Mobile Network Console are shown in Table 2.1.

Table 2.1 : Examples of access control rules regulating access to a mobile network console. (Source: Ardagna et al. [12])

| | subject | | action | object |
|---|--|---|-----------------|--------|
| | generic conditions | location conditions | | |
| 1 | $\text{user.Role=Admin} \wedge \text{Valid}(\text{user.Username}, \text{user.Password})$ | $\text{inarea}(\text{sim}, \text{Server Room}) \wedge \text{density}(\text{Server Room}, 1, 1) \wedge \text{velocity}(\text{sim}, 0, 3)$ | Configure | MNC |
| 2 | $\text{user.Role=Admin} \wedge \text{Valid}(\text{user.Username}, \text{user.Password})$ | $\text{inarea}(\text{sim}, \text{Inf. System Dept.}) \wedge \text{velocity}(\text{sim}, 0, 3) \wedge \text{local_density}(\text{sim}, \text{Close By}, 1, 1) \wedge$ | Read_Data | MNC |
| 3 | $\text{user.Role=CEO} \wedge \text{Valid}(\text{user.Username}, \text{user.Password})$ | $\text{local_density}(\text{sim}, \text{Close By}, 1, 1) \wedge \text{inarea}(\text{sim}, \text{Corporate Main Office}) \wedge \text{velocity}(\text{sim}, 0, 3)$ | Read_Data | MNC |
| 4 | $\text{user.Role=CEO} \wedge \text{Valid}(\text{user.Username}, \text{user.Password})$ | $\text{local_density}(\text{sim}, \text{Close By}, 1, 1) \wedge \text{disjoint}(\text{sim}, \text{Competitor Location})$ | Read_Statistics | MNC |
| 5 | $\text{user.Role=Guest} \wedge \text{Valid}(\text{user.Username}, \text{user.Password})$ | $\text{local_density}(\text{sim}, \text{Close By}, 1, 1) \wedge \text{inarea}(\text{sim}, \text{Corporate Location})$ | Read_Statistics | MNC |

The concepts discussed above provided the necessary consideration that needs to be taken into account when designing context aware services. Since we consider Android as our implementation platform we looked at several papers discussing the current security situation.

Since Snekkenes [11] concepts are not complete for this research the author decided to use the concepts proposed by both Snekkenes [11] and CA Ardagna et al. [12] When designing the proposed context-aware solution.

Context-Aware systems for other platforms

Work by Covington et al. [14] focuses developing context aware security systems for emerging applications. Their work considers a substantial area of Internet of Things. They envision a synergistic system called "Aware Home" where security policies imposed covers a breadth of devices including household appliances. The authors proposes a comprehensive security mechanism in which there is a policy specification language, a security management service, an authorization service, an environment role activation service, an authentication service and a context management service. Policy specification language extends role based access control system to include contextual information. Each of the services gather contextual information using available sensors and executes security policies defined for that specific context and user roles.

The authors work also proposes a GUI tool to define policy rules. This GUI allows a security administrator to associate permissions with various combinations of roles. For example, a child can be denied access to a category of resources that is classified using a single role, dangerous appliance, during certain environmental conditions (e.g., during a parent's working hours). This policy rule can be defined as shown in figure 2.1.

```
<GRBAC_TABLES>
  <POLICY>
    <SROLE> Child </SROLE>
    <OROLE> Dangerous Appliance </OROLE>
    <ACTION> ALL </ACTION>
    <EROLE> Working Hours </EROLE>
    <PERMS> Deny </PERMS>
  </POLICY>
</GRBAC_TABLES>
```

Figure 2.1 - Policy rules format of "Aware Home" system. (Source: Covington et al.

[14])

A similar concept is proposed by Al-Muhtadi et al. [16]. The authors discuss the implementation of a context aware security scheme for smart spaces. Although smart spaces consist of a vast number of integrated devices connected together, it has concepts that are still relevant to this research. The authors introduce a project called Gaia. Gaia provides the infrastructure for constructing smart spaces. In this Gaia project, there is a core component called "Cerberus" which handles core services in Gaia that integrates identification, authentication, context awareness, and reasoning. Cerberus enhances the security of ubiquitous applications that are built using Gaia. In the Gaia Context Infrastructure, contexts are defined as first order predicates. There are two kinds of policies used in Cerberus. One set of policies is used by the authentication server at the time of logon or authentication. These policies determine the confidence level of authentication. The other set contains access control policies, which determine whether a principal is allowed access to a particular resource. Context information from various sensors is provided to context synthesizers. Context synthesizers are components that get sensed contexts from various context providers, derive higher-level or abstract contexts from these simple sensed contexts and provide these inferred contexts to applications.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

2.3 Security Models of Smartphone Platforms

2.3.1 Mobile Security Models

In order to analyze the security models available for mobile devices, the author chose a simple taxonomy proposed by Jon Oberheide and Farnam Jahanian [15] as a reference model. We have further sub-divided the proposed taxonomy, in order to indicate the mechanisms we considered for our research. This extended reference taxonomy is shown below;

- Application Delivery
- Trust Levels - Permission Based security
- System Isolation - Virtualization/Sandboxing Based Security

This section discusses the generic approaches available to all the Mobile platforms, while the next section will discuss the Android OS specific security mechanisms.

Application delivery

Barrera and Van Oorschot [17] had carried out a survey on application delivery of mobile devices. The main application delivery model for mobile platforms is through the app stores. While some of these are closed/controlled app stores, there are other app stores which provide unrestricted access to the developers. Different mobile platforms use varying techniques to control the possibility of malicious applications entering their app store.

iOS

In case of apple devices that use iOS the applications are only delivered through the iTunes app store. The developers have to obtain a paid registration to submit applications to the app store. Furthermore, the company performs a combination of automated and manual verification tests to verify the authenticity and trustworthiness of the applications submitted to the app store by the developers. In this context, the iTunes is a very much a controlled app store with company verified security.

Android

Applications for Android can be downloaded through the “Android Market” (Google’s controlled app market) [17], or obtained directly through a developer’s site or third party app market (also known as "Sideloaded"). Google has minimal involvement when applications are uploaded to the Android market and no involvement when applications are distributed from a third party developer site.

Blackberry

Blackberry models can access a repository of RIM-approved applications called Blackberry "AppWorld" through an on-device application. Applications submitted to AppWorld are tested by RIM to check whether they interfere with the core

functionality of smartphones, before publishing them. Even though RIM must approve each submitted application for inclusion in AppWorld, developers are free to host their applications on other servers. However when compared with iOS and Android, application control level falls between iOS and Android, where control is not strict as iOS and stricter compared to Android.

Symbian

Symbian mandates that all applications be digitally signed, but not all signatures have to be issued by the Symbian foundation. Developers can self-sign their applications, allowing them to access “user capabilities”, which include making phone calls, initiating network connections, and accessing device location data. Applications that need to modify system settings or access core OS files (also known as “system capabilities”) must be submitted to the Symbian "Signed4" program for approval. Users can configure Symbian phones to check an online server for the validity of a certificate. At present development of Symbian and the Symbian App store is frozen.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Windows

Applications for windows platform are distributed only through Windows store. Similar to iOS Windows store is moderated and apps must be certified for compatibility and content.

Trust Levels

Permission Based security models

Kathy Au et al. [9] Surveys the Security models found on popular smart phone platforms. While praising the fact that most smart phone platforms employ sandboxing of 3rd party applications, the article points out that there are insufficient permission systems among the platforms. The authors highlight the over declaration of permission by application developers where the applications ask for more permissions than they actually need.

The permission systems are analyzed considering three main categories,

Control - Control indicates how much control the permission system gives the user over applications.

Information - The amount of information conveyed to the user is another factor considered in categorizing the permission systems.

Interactivity - Interactivity represents the amount of burden the permission system enforces on the user by considering the amount of interaction required for the setting up of permissions.

Table 2.2: Smart phone permission modes. (Source: Kathy Au et al. [9])

| OS | Initial Release Date | Number of Permissions | Control | Information | Interactivity |
|-----------------|----------------------|-----------------------|---------|-------------|---------------|
| Android | 2008/09/23 | 75 | Medium | High | Low |
| Windows Phone 7 | 2010/10/11 | 15 | Medium | Medium | Low |
| Apple iOS | 2007/06/29 | 1 | Low | Low | Low |
| WebOS | 2009/06/09 | 1 | Low | Low | Low |
| BlackBerry OS | 2006 Q3 | 24 | High | High | High |
| Maemo | 2005/11/1 | 0 | None | None | None |

Table 2.2 highlights the results of the survey. According to the discussion Windows and Android share the similar permission levels. Apple iOS has only one permission level and instead depend on the security provided by their App Store. Blackberry provides a great deal of interactivity when installing and using applications. Nokia's Maemo platform does not have permission levels.

The proposed security model [9] tries to eliminate the over declaration problem by a combination of lowering the cost of determining the correct set of permissions, making the cost of over declaration explicit to the user and making it easy for the developers to be aware of the permission used by the different components of the application.

Virtualizing/Sandboxing Based Security Models

Lange [19] discusses a popular approach to enhancing security of mobile devices using virtualization, this is already a proven technology available for desktops and servers. The authors highlight the features of new mobile devices such as Near Field Communication (NFC) which could make the device vulnerable. The authors reject the idea of relaying on security patches for mobile OS's, instead they propose to use a specially designed thin OS that act as a virtual memory manager, which provides the environment to run mobile operating systems such as Android. Authors use Android as the operating system of the proof of concept model. Also the authors claim the monolithic nature of the Linux kernel is not suitable for security sensitive mobile platforms, instead they propose to use a microkernel based approach.

The proposed [19] L4Android framework supports the following features,

Software Smartcards - Secure implementation of smartcard functionality in software

Unified Corporate and Private Phone - Private and a business phone can be unified on one device in a secure manner

Mobile Rootkit Detection - Rootkits are a class of malware that is known to hide inside the kernel. L4Android provides a virtual machine to run Android.

Hardware Abstraction - Implements a device specific drivers in a layer below Android

The Architecture of the proposed [19] solution is shown in Figure 2.2.

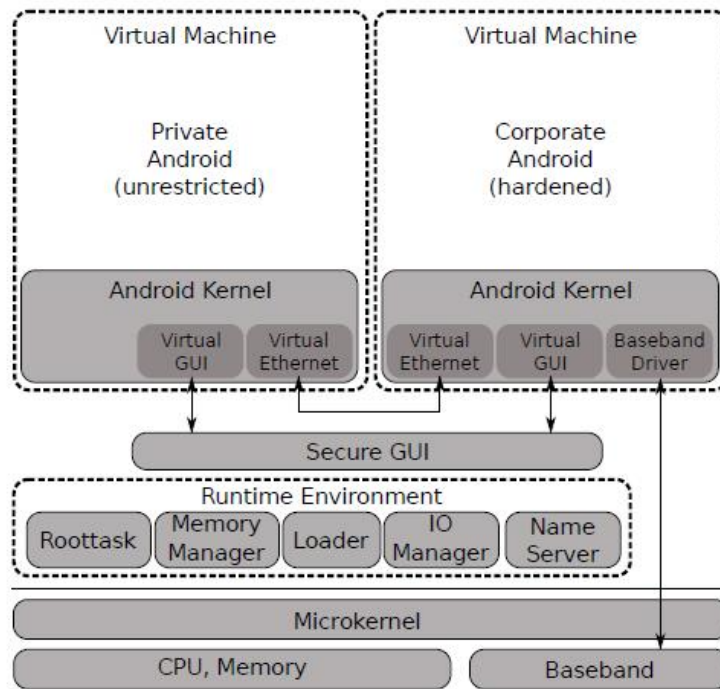


Figure 2.2 - L4Android Architecture. (Source: M Lange [19])



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Kernel of the system is significantly customized for this application. To establish isolation between components the kernel provides protection domains, called tasks. Execution is provided by threads while communication between processes is achieved by using Inter Process Communication (IPC). This kernel implements virtualization containers, thereby creating virtual machines on top of the kernel. Resource access controls are customized. Resources can be either memory pages or objects with an operation called map. Sender of a mapping can only grant access rights to others that it possesses itself, with the same or lower rights.

Gudeth et al. [20] discusses an attempt to move virtualization technologies available for desktops and servers into the mobile markets in hope for providing better security. The authors argue that having a trusted operating system as a part of their Trusted Computing Base (TCB) could be compromised and as such there should be other means of protection. The authors propose a bare metal hypervisor which mediates and providing an environment for an operating system to be run on top as a

guest OS. As shown in Figure 2.3, the bare metal hypervisor provides the enough resources that need to run a guest operating system or a trusted operating system on top. The objective of this research is to prevent or minimize the damages caused by internal threats and external threats.

Principles of the delivering secure applications on commercial mobile devices are,

- Minimize the trusted computing base
- Isolate trusted applications
- Reuse trusted software
- Be operating system agnostic
- Do not rely on technical competency of end users
- Minimize performance degradation
- Keep changes to commercial off the shelf (COTS) devices to a minimum
- Enable portability to new hardware
- Uniformly enforce system policy

Figure 2.3 shows the software architecture of the bare metal hypervisor-based implementation. Furthermore the concept design discusses the features such as static policy enforcement and virtualizing sensitive drivers.

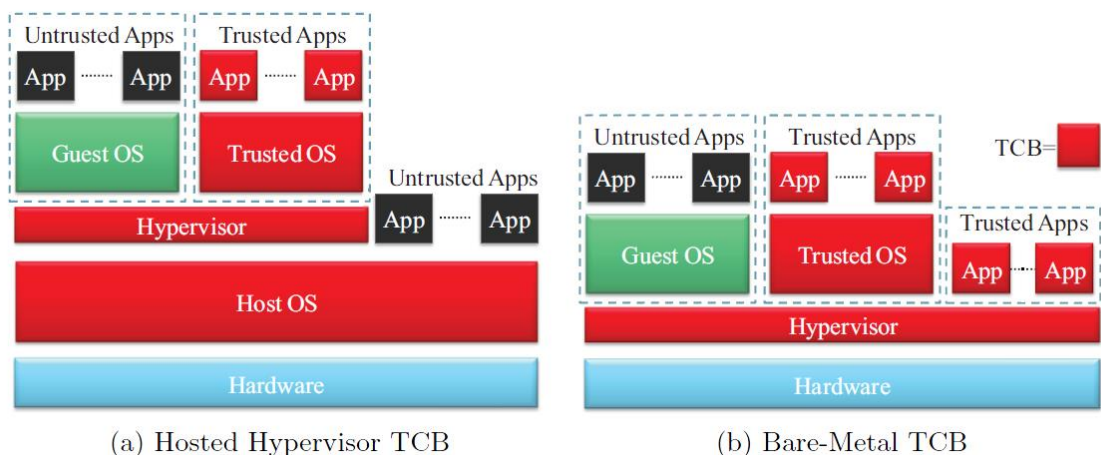


Figure 2.3 - Hosted hypervisor and baremetal hypervisor. (Source: Gudeth et al. [20])

2.4 Security Mechanisms for Android Mobile Devices

Current breed of smartphone manufacturers are locking the core of the operating system so as to prevent abuse of the cellular hardware by both users and malware. This is due to the inherent requirements of cellular networks. Locking of these devices makes customizations by users impossible. Husted et al. [10] regarded this as a bad practice as it will compel users to “Root” or “Jail Break” their devices, which in-turn makes the devices more vulnerable to malwares.

According to Barrera, and Van Oorschot [17] the Open Handset Alliance’s Android platform (mainly backed by Google) is an open source Linux-based middleware that runs on top of a Linux kernel. Android powers a variety of devices (over 60 smartphones models, tablets and netbooks as of July 2010 produced by a large number of manufacturers.

Applications for Android are written in Java and run in a custom virtual machine called Dalvik. Process and file system isolation is primarily provided by making each application run as its own user (standard UNIX UID). While Dalvik provides some isolation as well, Android makes no security claims or assumptions that the VM itself provides security. This is because application developers can create and invoke libraries written in C/C++, which run natively, beyond VM boundaries. A feature that makes Android unique in the smartphone space is that the OS allows applications to interact and use system resources based on a list of permissions labels.

Bai et al. [21] discusses context-aware usage control model "ConUCON", which leverages the context information to enhance data protection and resource usage control on a mobile platform. The Lack of fine grained access control for applications in Android is also addressed in this paper.

Bai et al. [21] argue that the granularity of application security provided by Android suffers major setbacks, these includes;

- Permission model of Android is coarse-grained and incomplete.
- The user cannot revoke or change the permissions of an application once he grants the permissions, unless the application is re-installed.

- It cannot provide data protection and resource usage constraints in a fine-grained manner.
- There is no mechanism for the user to enforce context-aware constraints on data and resources on Android.

Since the authors [21] propose to use Android as the host operating system in order to test the prototype system, the above-mentioned drawbacks need to be addressed. The proposed ConUCON model complements the UCON model with context awareness. The UCON model consists of eight components,

- Subjects
- Subject attributes
- Objects
- Object attributes
- Rights
- Authorizations
- Obligations
- Conditions



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.etd.lmu.ac.lk

The permission granularity problem in Android is also addressed in this framework.

Figure 2.4 shows the outline of the framework.

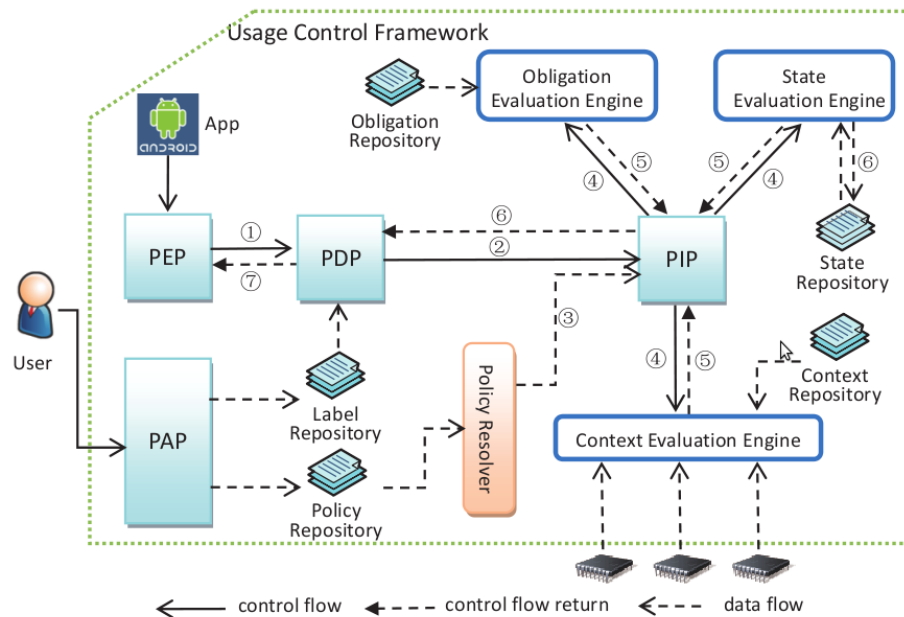


Figure 2.4: Android usage control framework (Source: Bai et al. [21])

Since the proposed framework acquires a considerable amount of overhead when run on a computationally limited mobile device, the authors [21] had carried out a performance evaluation. The measurements are carried out using the Android emulator. The results show a promising achievement of the implementation. The output delays starting various services are shown in Figure 2.5.

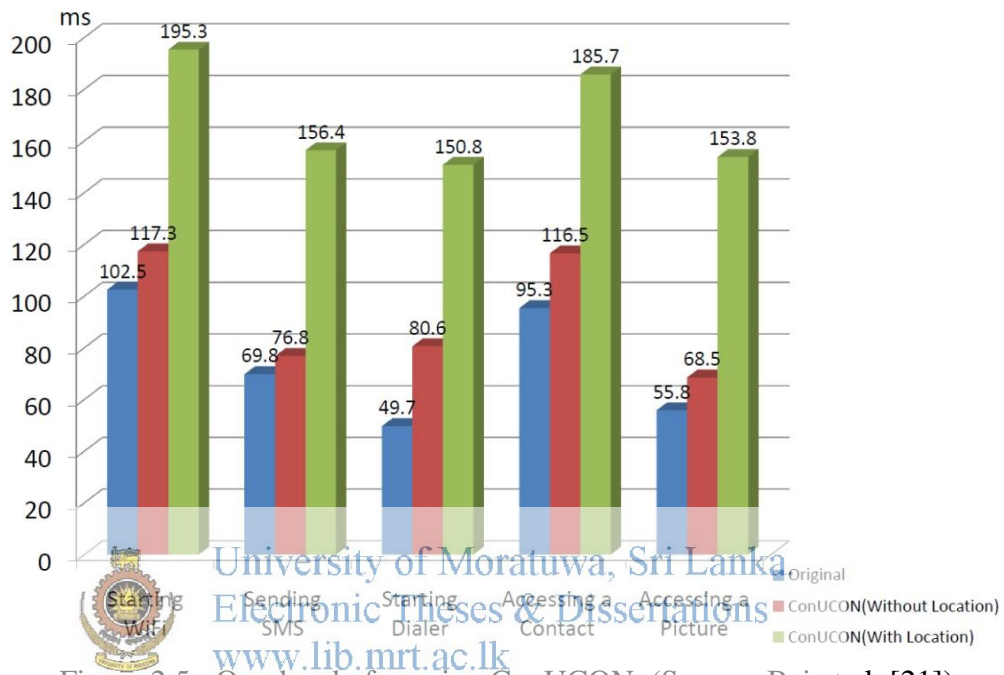


Figure 2.5 : Overhead of running ConUCON. (Source: Bai et al. [21])

Enck et al. [22] surveys the security model of Google Android platform. Authors use two in-house developed application in order to explore the security features. The first application discussed is a friend tracker and friend viewer application. The friend tracker application tracks friends while the Friend Viewer application shows the friends locations on a map. The tool is called “Kirin”, which is capable of extracting applications security policy from its manifest file and displays it to the user, who in-turn can determine whether or not to install the application.

Android’s main selling point is the seamless integration of Google’s Gmail, Calendar and Contacts. Android uses a simple permission label assignment model to restrict access to resources and other applications, but for reasons of necessity and convenience, its designers have added several potentially confusing refinements as

the system has evolved. Each android application consists of a collection of components. These components can be divided into the following categories,

Activity - These components define an application's user interface.

Service - These components perform background processing.

Content Provider - These components store and share data using a relational database interface.

Broadcast receiver - These components act as mailboxes for messages from other applications.

The developer specifies components and their security policies using a XML manifest file that accompanies every application package. Components interact with each other and the system using 'intents'. Which is simply a message object containing a destination component address and data.

Android follows a Mandatory Access Control (MAC) policy. Furthermore security enforcement occurs in two places: each application executes as its own user identity, allowing the underlying Linux system to provide system-level isolation; and the Android middle-ware contains a reference monitor that mediates the establishment of Inter-Component Communication (ICC). All ICC occurs via an I/O control command on a special device node, "/dev/binder". Android middle-ware mediates all ICC establishment by reasoning about labels assigned to applications and components. If the label isn't in the collection, establishment is denied even if the components are in the same application. For E.g. Component A's ability to access components B and C is determined by comparing the access permission labels on B and C to the collection of labels assigned to the parent application. However Android's permission label model only restricts access to components and doesn't currently provide information flow guarantees.

The permission over declaration problem in Android is discussed further in the paper by Vidas et al. [23]. The Authors propose a developer tool which helps developers in deciding the required permissions for a developed application using source code analysis. The authors had implemented their approach as an Eclipse IDE-Plugin.

Table 2.3 shows the number of applications that request duplicate permissions by market category, demonstrating that even reference libraries and medical applications contain some duplicates. Table 2.4 shows the permissions most often duplicated.

Table 2.3: Applications with duplicate permissions by market category. (Source: Vidas et al. [23])

| Market Category | Total Apps | With Duplicates |
|------------------------|-------------------|------------------------|
| Arcade and Action | 1344 | 17 |
| Books and Puzzle | 1452 | 24 |
| Brain and Puzzle | 1352 | 14 |
| Business | 1092 | 38 |
| Cards and Casino | 842 | 85 |
| Casual | 966 | 4 |
| Comics | 838 | 11 |
| Communication | 1311 | 77 |
| Education | 1305 | 21 |
| Entertainment | 1522 | 40 |
| Finance | 1354 | 44 |
| Health and Fitness | 1258 | 36 |
| Libraries and Demo | 1156 | 21 |
| Lifestyles | 1489 | 48 |
| Live Wallpaper | 537 | 14 |
| Media and Video | 1360 | 49 |
| Medical | 527 | 2 |
| Music and Audio | 1124 | 89 |
| News and Magazine | 1419 | 63 |
| Personalization | 1342 | 54 |
| Photography | 1165 | 283 |
| Productivity | 1319 | 54 |
| Racing | 216 | 76 |
| Shopping | 1155 | 46 |
| Social | 1296 | 41 |
| Sports | 1433 | 23 |
| Sports Games | 365 | 71 |
| Tools | 690 | 27 |
| Transportation | 454 | 8 |
| Travel and Local | 1473 | 35 |
| Weather | 342 | 4 |
| Widgets | 1395 | 64 |

Table 2.4: Top ten duplicate permissions requested. (Source: Vidas et al. [23])

| Permission | Count |
|------------------------|-------|
| INTERNET | 620 |
| ACCESS_NETWORK_STATE | 438 |
| READ_PHONE_STATE | 153 |
| RECEIVE_BOOT_COMPLETED | 147 |
| WRITE_EXTERNAL_STORAGE | 59 |
| READ_CONTACTS | 49 |
| ACCESS_FINE_LOCATION | 48 |

The authors had created a permission-API database with one-to-many permission-API mappings by manually parsing the API documentation and creating a database of functions and permissions upon which they depend. Since software developers rarely make the source code available, it is very difficult or impossible to measure permission over declaration in applications by users other than the developer.

Vidas et al. [24] had carried out a survey on current Android attacks. The Android security model creates several new security sensitive concepts such as Android's application permission system and the un-moderated Android market. The authors try to provide a taxonomy of mobile platform attack classes with specific, concrete example as each class applies to the Android environment. The authors also argue that the slow patch cycle for android is making the devices more vulnerable. The authors discuss the absence of clear permission models. The authors discuss Android security issues such as,

- Application model
- Patch cycles
- Trusted USB connections
- Recovery mode and boot process
- Uniform privilege separation

Threats for Android is classified according to the following classes,

- No physical access
- Physical access with ADB enabled

- Physical access without ADB enabled
- Physical access on unobstructed device

Under mitigations the authors discuss,

Reduce the Patch Cycle Length - Reducing the patch cycle length would mitigate these threats with greater effectiveness. Zero-day exploits would still be possible, however the common lingering threats will be reduced.

Privileged Applications - With a market model split into trusted and untrusted applications, Google could provide enhanced security with minimal administrative overhead and minimal reduction in the openness of the platform.

Leveraging Existing Security Technologies - Generally, operating system level software modifications such as adding a firewall or SELinux to Android involve porting existing technology to the Android kernel and creating an application to facilitate administration.

Authenticated Downloads - To ensure downloads are made only by the user, the market should require authentication before every transaction, similar to the model currently used by the iPhone.

Authenticated ADB - With ADB authentication, the attacker no longer has a backdoor to bypass the lock mechanism's authentication process, mitigating the ADB attack against obstructed devices.

Trusted Platform Module - Using a Trusted Platform Module (TPM) provides a ground truth on which device security could be built, providing authentication of device state.

The "TaintDroid" approach proposed by Enck et al. [25] provides real-time analysis by leveraging Android's virtualized execution environment. "TaintDroid" incurs only 14% performance overhead on a CPU-bound micro-benchmark and imposes negligible overhead on interactive third-party applications. The primary goals are to detect when sensitive data leaves the system via un-trusted applications and to facilitate analysis of applications by phone users or external security services. When tainted data are transmitted over the network, or otherwise leave the system,

"TaintDroid" logs the data's labels, the application responsible for transmitting the data, and the data's destination. TaintDroid leverages Android's virtualized architecture to integrate four granularities of taint propagation: variable-level, method-level, message-level, and file-level. Figure 2.6 shows the above discussed approach.

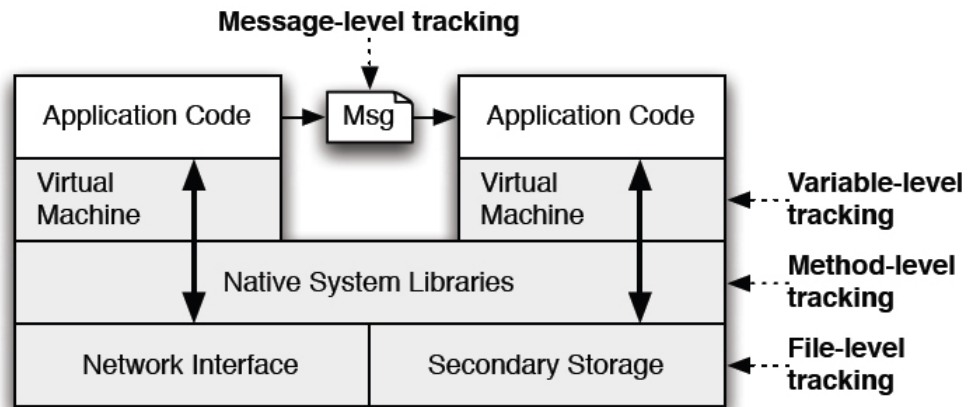


Figure 2.6: Multi-level approach for performance efficient taint tracking within a common smartphone architecture. (Source: Enck et. al. [25])

This approach extensively modifies the internal working of the Android OS, therefore unfortunately this requires a custom built Android OS for each device since the usual Android applications cannot gain the capabilities discussed in this paper. This is currently impossible to obtain.

2.4.1 Security Refinements of Android SDK

In Android SDK release 1.0r1 following additional security refinements were introduced,

Public vs. Private Components

By making a component private, the developer doesn't need to worry which permission label to assign it or how another application might acquire that label. If a public component doesn't explicitly have an access permission listed in its manifest definition, Android permits any application to access it.

Broadcast Intent Permissions

Unprotected intent broadcasts can unintentionally leak information to explicitly listening attackers.

Content Provider Permissions

Security-aware developers should define separate read and write permissions, even if the distinction isn't immediately apparent.

Service Hooks

Lets developers arbitrarily extend the reference monitor with a more restrictive policy.

Protected API's

By protecting sensitive API's such as hardware access, Android forces an application developer to declare the desire to interface with the system in a specific way.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Permission Protection Levels

3 Permission levels are introduced,

Signature and System - Permissions are granted only to applications signed by the same developer key as the package defining the permission.

Normal - Permissions act like the old application permissions and are granted to any application that requests them in its manifest.

Dangerous - Permissions are granted only after user confirmation.

Pending Intents

The authors discuss the delegation of intents to another. Features that are shared are discussed below.

URI Permissions

Enforces permissions for content providers which provides services using URI's.

In the ending discussion the authors note that although Android security policy revealed that it begins with a relatively easy to understand MAC enforcement model, but the number and subtlety of refinements make it difficult for someone to discover an application's policy simply by looking at it. The validity of assigning permissions using text strings and the lack of more expressive declarations are questioned. Furthermore by using their tool 'Kirin' authors claim that they have discovered some vulnerability in the base applications distributed with android.

2.5 Threat Models and Malware for Mobile Devices

La Polla et al. [32] has conducted an extensive survey on mobile malware for the period of 2004-2011. The authors point out that from 2009 to 2010 there was a 42% percent increase in mobile malware. According to the authors threat models for mobile devices include,

- Wireless Break-in based - Exploiting the wireless capabilities of the device, includes Wi-Fi and Bluetooth
- Infrastructure based - Exploiting the vulnerabilities of GSM Infrastructure, e.g. Breaking the encryption
- Worm based - Designing of software which propagates from device to device
- Botnet based - Taking control of mobile devices remotely
- User based - Exploiting the security vulnerabilities using physical means. E.g. Direct hardware access etc.

Authors refers to various sources on regarding the evolution of mobile malware, From various survey papers on the same subject and reports from reputed antivirus companies. Examples for mobile malware are shown in Table 2.5. Authors discuss the current and future trends of mobile malware. A comparison between Mobile security vs personal security is also discussed, with an emphasis on the inherent features of mobile devices such as mobility, strong connectivity and technology convergence and reduced capabilities.

Table 2.5: Malwares and their methods of infection. (Source: La Polla et al. [32])

| Name | Time | Type | Method of Infection | Effects | OS |
|------------------|------|----------------------|--|--|-----------------------|
| Liberty Crack | 2000 | Trojan | Pretend to be a hack | Remove third-party software | Palm OS |
| Cabir | 2004 | Worm | Bluetooth connection and copies itself | Continuous scan of Bluetooth, drain phone's battery | Symbian OS |
| Dust | 2004 | Virus | File Infector | Infect all executables in root DIR | Windows Mobile |
| Brador | 2004 | Trojan | Copy itself in to the startup folder | Open a backdoor | Windows Mobile |
| Mosquitos | 2004 | Trojan | Embedded in a game | Send SMS to premium-rate numbers | Symbian OS |
| Skulls | 2004 | Trojan | Vulnerability in overwriting system files | DoS | Symbian OS |
| MetalGear | 2004 | Trojan | Vulnerability in overwriting system files | Disable virus scanner | Symbian OS |
| CommWarrior | 2005 | Worm | Replicates via Bluetooth and MMS | MMS charging | Symbian OS |
| Doomboot | 2005 | Trojan horse | Doom 2 video game | Prevents booting and installs Cabir and CommWarrior | Symbian OS |
| Lasco | 2005 | Virus | File infection | Add itself to install packages | Symbian OS |
| Locknut | 2005 | Trojan | Vulnerability in OS | Create entries for a new application | Symbian OS |
| Feakk | 2005 | Worm | SMS message | Send SMS to all contacts | Symbian OS |
| Cardblock | 2005 | Virus | Fake SIS application | Encrypt memory card with a random password | Symbian OS |
| CardTrap | 2005 | Cross-Platform Virus | Auto-start of removable storage | Copy Wukill on the phone | Symbian/Windows OS |
| Blankfont | 2005 | Trojan | Replace font files | Fonts not displayed | Symbian OS |
| Crossover | 2006 | Cross-Platform Virus | CIL vulnerabilities | Copy to/from mobile/PC | Windows/Mobile OS |
| Letum | 2006 | Worm | E-Mail spreading | Infect registry | Windows Mobile |
| Fontal | 2006 | Trojan | Vulnerability in overwriting system files | Device not restart after reboot | Symbian OS |
| Mobler | 2006 | Cross-Platform Worm | Dropping Mechanisms | Disable antivirus and infect removable storage | Symbian/Windows OS |
| Redbrowser | 2006 | Trojan | Fake Browser | Send SMS continuously | OS-Independent (J2ME) |
| Wesber | 2006 | Trojan | Fake Browser | Send SMS to premium-rate numbers (Russia only) | OS-Independent (J2ME) |
| Acallno | 2006 | Spyware | Fake Commercial Software | Gather and send information about user's activities | Symbian OS |
| Lasco | 2007 | Worm | A worm that spreads over Bluetooth networks | Searching and infecting other phones | Symbian OS |
| Feak | 2007 | Worm | Proof-of-concept worm | Sending SMS to contact list with URL | Symbian OS |
| Flocker | 2007 | Trojan | It claims to be an ICQ application to trick the user | Sending SMS to a hard coded phone number | Symbian OS |
| Beselo | 2008 | Worm | Via MMS and Bluetooth fake application | MMS charging | Symbian OS |
| InfoJack | 2008 | Trojan | Attach itself to installation packages | Disable security settings | Windows Mobile |
| Pncryptic | 2008 | Worm | Memory card spreading | Dialing premium-rate numbers | Windows Mobile |
| Yxe | 2009 | Worm | SMS containing malicious URL | Send contact lists to external server | Symbian OS |
| Yxes | 2009 | Worm/Botnet | SMS containing malicious URL | Send contact lists to external server | Symbian OS |
| Ikeek | 2009 | Worm | Scanning a IP ranges and SSH | Alter wallpaper | iPhone |
| FlexiSpy | 2009 | Spyware | Fake Application | Tracking/log of device's usage | Symbian |
| Curse of Silence | 2009 | SMS Exploit | Vulnerabilities in e-mail parsing | Disable SMS functionalities | Symbian OS |
| ZeUS MitMo | 2010 | Worm | Fake SMS | Steal bank account information | Cross-Platform |
| iSAM | 2011 | Multifarious malware | Scanning IP and connecting to SSH | Collect private information, send malicious SMS, DoS | iPhone |



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Security solutions for mobile devices are also discussed. Security solutions for mobile devices are also categorized. These include Intrusion Detection Systems (IDS) and Trusted Mobile Solutions. Authors partition existing IDS solutions into detection principles, architecture, reaction, collected data and Operating Systems (OS). The authors present the idea of detecting malicious Applications by using a behavior based anomaly detection techniques employing methods such as system call, function call and network operations monitoring. Implemented examples of such systems are also presented.

The work by Felt [26] looks at incentives behind 46 pieces of iOS, Android and Symbian malware that spread in the wild from January 2009 and June 2011. The collected 46 pieces of malware distributed among the mobile platforms, 4 for iOS, 24 for Symbian and 18 for Android. The authors had found that 61% of the malware collect user information, 52% send premium-rate SMS messages, in addition to

malware that was written for novelty or amusement, credential theft, SMS spam, search engine optimization fraud, and ransom.

Threat Models of Applications

The authors discuss three types of threat posed by third party Smartphone applications. Which are,

Malwares - Malware gains access to a device for the purpose of stealing data, damaging the device, or annoying the user, etc.

Personal Spyware - Spyware collects personal information such as location or text message history over a period of time.

Grayware - Some legitimate applications collect user data for the purpose of marketing or user profiling.

Security Measures for Applications

Markets - Smartphone users are encouraged to download and purchase applications from centralized application markets. Apple, Google, and Nokia promote the use of centralized markets with decreasing strictness.

Permissions - Smartphone operating systems may also protect users by requiring user consent before an application can access sensitive information or dangerous capabilities.

According to the authors “Jailbreaking” or “Rooting” of devices are actually making the devices more vulnerable. 46 malware are categorized by their behavior. Furthermore incentives for writing malware are also discussed.

The authors had categorized incentives as current incentives and future incentives.

Current incentives

- Novelty and Amusement
- Selling User Information
- Stealing User Credentials

- Premium-Rate Calls And SMS
- SMS Spam
- Search Engine Optimization
- Ransom

Future Incentives

- Advertising Click Fraud
- Invasive Advertising
- In-Application Billing Fraud
- Governments
- E-Mail Spam
- Distributed Denial of Service
- NFC and Credit Cards

The Authors note that most of the malwares could be automatically detected according to their behavior. These malwares asks permissions such as sending SMS, reading the IMEI number of the device etc.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Authors discuss the presence of “jailbreaks” and “rooting” of devices, The authors argue that that the current “locked” model used by phone vendors is detrimental to the security of end users because it aligns the incentives of attackers and smartphone users. According to estimates, 15% to 20% of Android phones are rooted, and 6% of iPhones are "Jailbroken".

The report published by United States computer emergency response team (US-CERT) [27] provides suggestions for keeping mobile devices safe. The report discusses the importance of mobile device security and the rate at which the exploitations increase and the inherent short comings of the mobile platforms.

We now turn our attention to containment of malware threats using novel approaches. This includes providing the best security and stability possible while minimizing the performance overhead.

2.6 Malware Detection Techniques for Mobile Devices

2.6.1 Detection in Mobile Device Only

With the increase of malware attacks on mobile devices, there had been an extensive research into the development of malware detection techniques. Ramu [33] surveys the available malware detection methods for mobile devices.

Static Analysis

Static Analysis is a fast and simple approach. Chandramohan et al. [33] summarized the static analysis techniques suggested in various papers. There are three types of static analysis. System call based, Static taint analysis, Source code analysis.

System call based: The mobile application is first dissembled using software tools. This is to extract the System calls made by the application and then passed to a processing engine to perform anomaly detection and classify applications based on the malicious activities.

Static taint analysis: Egele et al. [33] analyzed static taint analysis on iOS application binaries. The study was carried out by developing an automated tool named PiOS that was capable of verifying privacy breaches. The PiOS tool uses Static Analysis to check if the application accesses sensitive information and transmit it over the network.

Source code analysis - involves identifying vulnerabilities in control flow, data flow, structural and semantic analysis. The application is first decompiled in order to analyze.

Dynamic Analysis

An interesting concept is proposed in "Andromaly" framework by Shabtai et al. [28], which focuses on creating a generic and modular framework for detecting malware on Android mobile devices using its behavior. The detection process consists of real-time, monitoring, collection, pre-processing and analysis of various system metrics,

such as CPU consumption, number of packets sent through the Wi-Fi, number of running processes and battery usage of applications. Under related work the authors discuss malware detection techniques such as static and dynamic analysis. Static and dynamic analysis solutions are primarily implemented using two methods: signature-based and heuristic-based. Signature-based is a common method used by antivirus vendors and it relies on the identification of unique signatures that define the malware. The heuristic-based methods are based on rules which are either determined by experts or by machine learning techniques that define a malicious or a benign behavior, in order to detect unknown malware. While the methods discussed under related work focus on offline application behavior analysis this paper attempts to understand the feasibility of applying the detection on the device. Figure 2.7 shows the architecture of the framework.

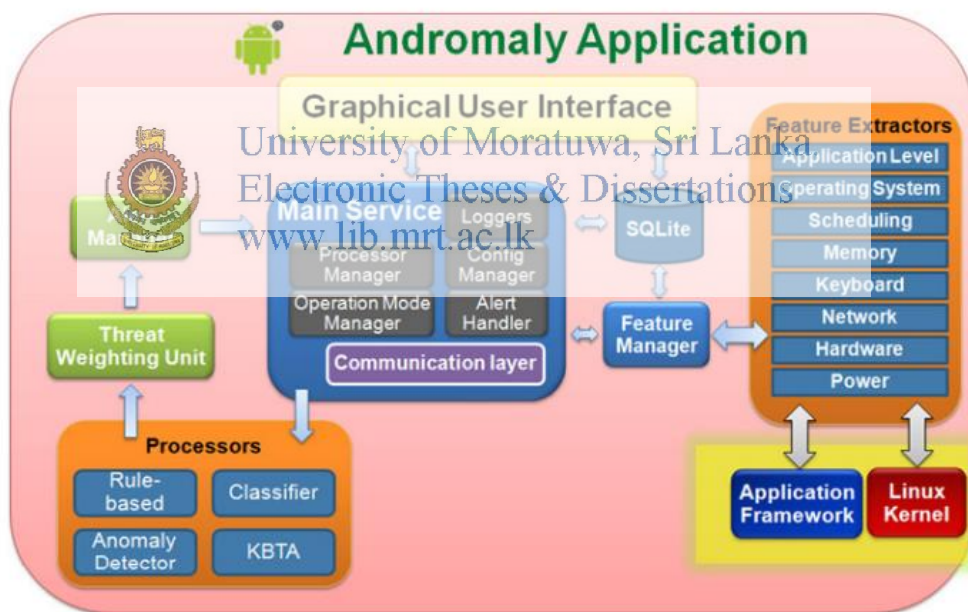


Figure 2.7 - "Andromaly" framework overview. (Source: Shabtai et al. [28])

Detection Method

The proposed framework employs machine learning classification techniques for realizing a malware detection system. Authors states that based on previous

experience and after weighing the resource consumption issue they selected the following classifiers.

- k-Means
- Logistic Regression
- Histograms
- Decision Tree
- Bayesian Networks
- Naïve Bayes

Evaluation of machine learning classifiers is typically split into two subsequent phases: training and testing.

The authors evaluation tries to address the following questions.


- Is it possible to detect unknown malicious applications on Android devices using the "Andromaly" framework?
- Is it possible to learn the behavior of applications on a set of Android devices and perform the detection on other devices?
- Which Classification algorithm is most accurate in detecting malware on Android devices: DT, NB, BN, k-Means, Histogram or Logistic Regression?
- Which number of extracted features and feature selection method yield the most accurate detection results: 10, 20 or 50 top-features selected using Chi-Square, Fisher Score or InfoGain?
- Which specific features yield maximum detection accuracy?

Feature selection

In Machine Learning applications, a large number of extracted features, some of which redundant or irrelevant, present several problems such as - misleading the learning algorithm, over-fitting, reducing generality, and increasing model complexity and run-time [28]. These adverse effects are even more crucial when applying Machine Learning methods on mobile devices, since they are often restricted by processing and storage-capabilities, as well as battery power.

A filter based approach was used by authors [28] for feature selection. Three feature selection methods were applied to the datasets: Information Gain (IG), Chi-Square (CS) and Fisher Score (FS). These feature selection methods follow the Feature Ranking approach and, using a specific metric, compute and return a score for each feature individually. Chi-Square measures the lack of independence between a feature f and a class C . The Fisher Score expresses the difference between two classes relative to a specific feature taking into account the mean and standard deviation of the feature's values in different classes. Information Gain determines the amount of information which a feature provides about a class by measuring how well it separates the training examples according to their target classification.

In order to avoid any bias by selecting an arbitrary number of features, the authors used, for each feature selection algorithm, three different configurations: 10, 20 and 50 features that were ranked the highest out of the 88 featured ranked by the feature selection algorithms.

Evaluation  University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

The authors consider several parameters when evaluation the outcome, accuracy, false positive rate (FPR), area-under-curve (AUC), true positive rate (TPR). Evaluation is split into 2 experiments for each of the parameters mentioned above.

Experiment 1

The purpose of this experiment is to evaluate the ability of each combination of detection algorithm, feature selection method, and number of top features to differentiate between game and tool applications when training set includes all game/tool applications. The training set contained 80% of the feature vectors of both the game and tool applications. The feature vectors were assigned in a random fashion. This experiment was repeated for each device 20 times, with different allocations of the training and testing set which results in a total of 5,400 runs.

Experiment 2

The purpose of this experiment is to evaluate the ability of each combination of detection algorithm, feature selection method, and number of top features to differentiate between game and tool applications not included in the training set. The configuration of this experiment resembles the first one. However, unlike the first experiment the training set contained feature vectors clusters for 80% of all games and 80% of all tools. The testing set contained feature vectors clusters of the rest of the 20% games and 20% tools that were not included in the training set on the same device. This examined the ability of the different algorithms to detect unknown applications. This experiment was repeated for each device 20 times, with different allocations of the training and testing set which results in a total of 5,400 runs

As the authors note in that the memory consumption of the application was steady in the interval $16,780 \text{ Kb} \pm 32$ (which is approximately 8.5% of the device's RAM) and the CPU consumption was in the interval $5.52\% \pm 2.11$. Furthermore the authors advise that in-order to minimize battery usage the system can be scaled down so that detection can be performed using only 10 features and a smaller number of classifiers.



University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Although there is no one single algorithm which is best, there are two algorithms which stands out from the rest. When implementing a malware detection system for Android it is recommended by the authors [28] to have a meshed approach of the following two methods: Classification and Anomaly detection and Misuse-based detectors (e.g., rule-based, knowledge-based). When they operate in a synergy. they can yield an optimal recall rate and low false alarms.

2.6.2 Detection at a Separate Server

The approach proposed by I Burguera and S Nadjm-Tehrani [29] proposes the use of a crowd-sourcing system to obtain the traces of applications' behavior, which helps researchers to collect different samples of application execution traces. The whole analysis process is on a dedicated remote server. The server will exclusively collect

information and detect malicious and suspicious applications in the Android platform by analyzing OS system calls. Furthermore a dataset of behavior data will be created for every application used. Finally, they cluster each dataset using a partitional clustering algorithm. Partitional clustering is simply a division of the set of data objects into non overlapping sub sets (clusters) such that each data object is in exactly one subset. The clustering algorithm used is K-Means. Figure 2.8 shows the overall architecture of the system.

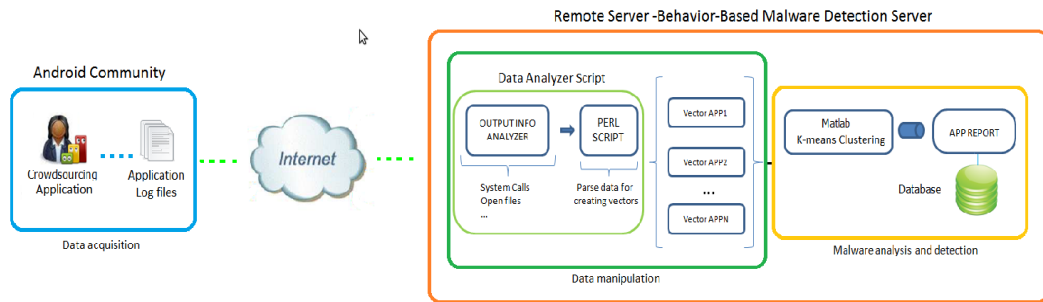


Figure 2.8 - "Crowdid" Framework Overview. (Source: I Burguera and S Nadjm-Tehrani [29])

The authors had used custom written malwares as well as a few set of available ones to test their concept, since none of the malwares available at that time showed features that they expected. The authors claim that the results were promising since the custom written malwares were detected with 100% accuracy.

Dynamic Analysis vs. Detection at a separate server

Dynamic Analysis is expensive to run on the device. Since there is no one particular malware detection algorithm, using a collection of algorithms will drain the mobile devices' limited resources. The "Andromaly" framework discussed above suggest to use a low number of features. However it does not mention the accuracy of the system after this modification. The authors tested the malware detection only with a handful of applications. There would be false positives if the classification algorithm is not trained properly. Offloading the processing to a server is a viable option in this regard. The results shown by Burguera and Nadjm-Tehrani [29] offers promising results. In fact their detection system takes malware detection into a whole new level by using a crowd sourced implementation rather than operating on a single device. Furthermore by using this approach, the chances for false positives decreases.

CHAPTER 3 - DESIGN AND IMPLEMENTATION

3.1 The Proposed Software Architecture

The research carried out by Burguera et al. [29] discusses the benefits of a client server architecture, their discussion emphasizes the offloading of processing into a remote server and having the mobile device act as a thin client. This divided architecture allows the application to be responsive as well as robust. The client extract the features from applications and these extracted features are pushed into a server, where it would process these features to detect malware behaviors. As mentioned earlier the limitations of mobile devices can be overcome by using such an architecture. The client can offload heavy processing required for the malware detection into a remote server, while it can perform the application tracing (feature extraction) on the device itself since it's not resource intensive as discussed in previous research ideas. However, using this architecture also has some drawbacks such as connectivity problems, security of the connection etc. In order to overcome the connectivity problem, the client part of the software is made to work without any Internet connections and connect only to the server when there is an active Internet connection. This enables the users to work with the software without major impacts. The security of the connection is not addressed in this research.

The client component is run as a background service, in order to provide active status changes. This service includes two separate monitoring systems, the location change monitoring component (LCMC) and application launch monitoring component (ALMC). The components are developed as loosely coupled components in-order to provide easy maintenance. The location change monitoring component listens to location context changes occurring with the device and applies security rules for that particular location context. The user can create security rules for locations using geo coordinates or use Wi-Fi access point name (SSID) as a location beacon. The research by Ardagna et al. [13] discusses location based security rule systems. The system described here uses a hierarchical rule system according to location

determined by location beacons. In this proposed architecture we use a simple flat Rule System which relies upon Geo-Coordinates or Wi-Fi access point ID (SSID). This prototype idea can be extend to support various security rule systems. The user can setup Wi-Fi, Bluetooth, Screen timeout settings for a particular location.

As discussed in the paper by Burgera at al. [29], monitoring of system calls are an effective method to detect malwares on mobile devices. Hence in order to identify application execution patterns, the proposed software architecture utilizes this method. Only a minimalistic software change is needed to run the software. Instead of a clustered approach used by the above mentioned authors, the proposed software uses a simplified score assigning method where rule patterns are given a score. This part of the software runs on the server. For e.g. If an application uses open system call, the system would assign a fixed score, the application would get a final cumulative score according to its system call pattern. The rules are defined as system calls, URL and text strings. The inspiration for these ideas were also drawn from the work by Bugeira et al. [29].



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Due to the client server architecture and the reuse of existing technologies makes the solution robust. The client can be any platform, Android, Apple, Nokia etc. and the server can cater to all of these platforms without the need of separate instances. Figure 3.1 shows the architecture of the proposed system("AndroSec").

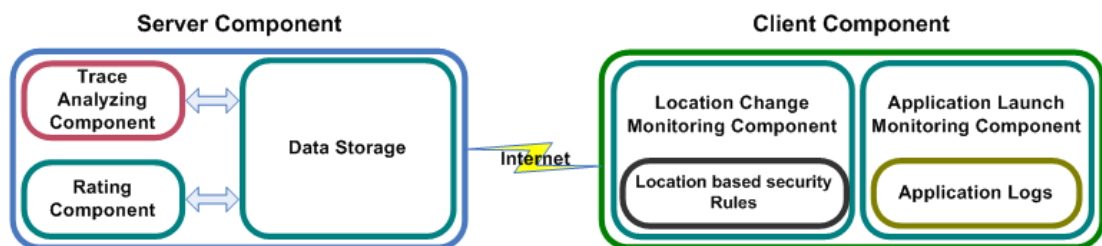


Figure 3.1 - The proposed architecture of the "AndroSec" system

3.1.1 The Server

The server is a passive entity where application traces are uploaded from clients. The client component is configured to connect to a particular server using the network or the Internet. It is responsible for the processing of uploaded application traces and to assign a cumulative score. The features required in the server were broken down into two components, Application trace analyzing and rating components. These two components complement each other. Each one can be improved without significant problems to other one. The server would have the capability to track application trace uploads and collect metadata such as phone ID, date and time of the upload, package name and application name furthermore it would process the uploaded application traces and report back the results to connected clients.

The server would use a database to store application trace metadata storage. The final cumulative score is also maintained in this database.

Clients would connect to the server using the HTTP protocol. This is done to ease the development efforts to manageable level since HTTP already has the required features, such as posting data and uploading files. In order to handle the HTTP requests Apache web server is used, because of its wide availability and popularity. PHP web application language is used as a programming language, due to its ease of deployment, lightweight and speed. There would be several scripts to be run by the client application, these scripts would perform certain actions on the server. These would include uploading application traces and downloading the final report on applications. The server would be a passive entity in this setting where the clients would connect and disconnect.

3.1.2 The Client

The client software runs on the device. This part of the software need to be custom written for any platform e.g. Android, Apple, Windows etc. The proof of concept component is written for Android. The client runs as a background service doing

minimal processing. This is to minimized battery and CPU usage since as mentioned previously these are very limited in mobile devices. Although the processing is offloaded to a remote server the client software itself does not handle the battery and CPU usage management. The design decisions were influenced by several papers discussed previously. The papers evaluated in this research have used similar functionality.

The client contains two subcomponents, location policy management and application auditing and reporting components. The client is an active entity in the proposed solution where it would monitor the location changes occurring on the device and apply policy rules according to these anticipated changes. Furthermore it is responsible to collect application behavior traces and upload in to the server.

3.2 The Implemented Solution

A proof of concept was developed in support of the above proposed software architecture. Each of the features discussed above is implemented in the proof of concept development. The client section was implemented on Android. While the server section was implemented using PHP, MySQL as the database and Apache was used as the web server.

We used a version of android Operating System (Version 2.3) which had a higher penetration than others [30]. The development of the software was carried out on a PC, while the testing was done on the Android emulator software.

The client software uses the location sensor data available on Android. The Android system provides action listeners for various events that occur when using the device. Our system uses these services to adapt or change the security settings ie. Levels. However Android operating system doesn't differentiate between a privileged application from a non privileged one, all the applications run inside a sandbox environment. This prevented our application to work as expected; hence the authors

used a method called “rooting” to provide the required privilege requirements that are not available by default.

3.3 The Client Software

The client software runs on the device. Its duty is to monitor location changes, apply security policies according to location changes, collection of application traces and communicating with the server software. A Linux utility called "strace" should be installed on the device.

The device software contains several sub-components. These include an agent which monitors the changes on the device such as location changes and applies security rules according to the location and monitors application launches and collects system call traces.

3.3.1 Main Activities

The client component carries out several functionalities as mentioned earlier. These functionalities would be listening to location context changes on the device, collecting application logs and connecting to remote server to upload data and download reports.

Once the client service is started, it would listen to location context changes occurring on the device. The software would acquire the current location coordinates or the Wi-Fi SSID as a location identifier. If there exists a location rule defined for that particular location the software would apply the settings of the rule. If the coordinates are not defined as a rule, the software would apply a general rule if its defined. Figure 3.2 elaborates this functionality of the software.

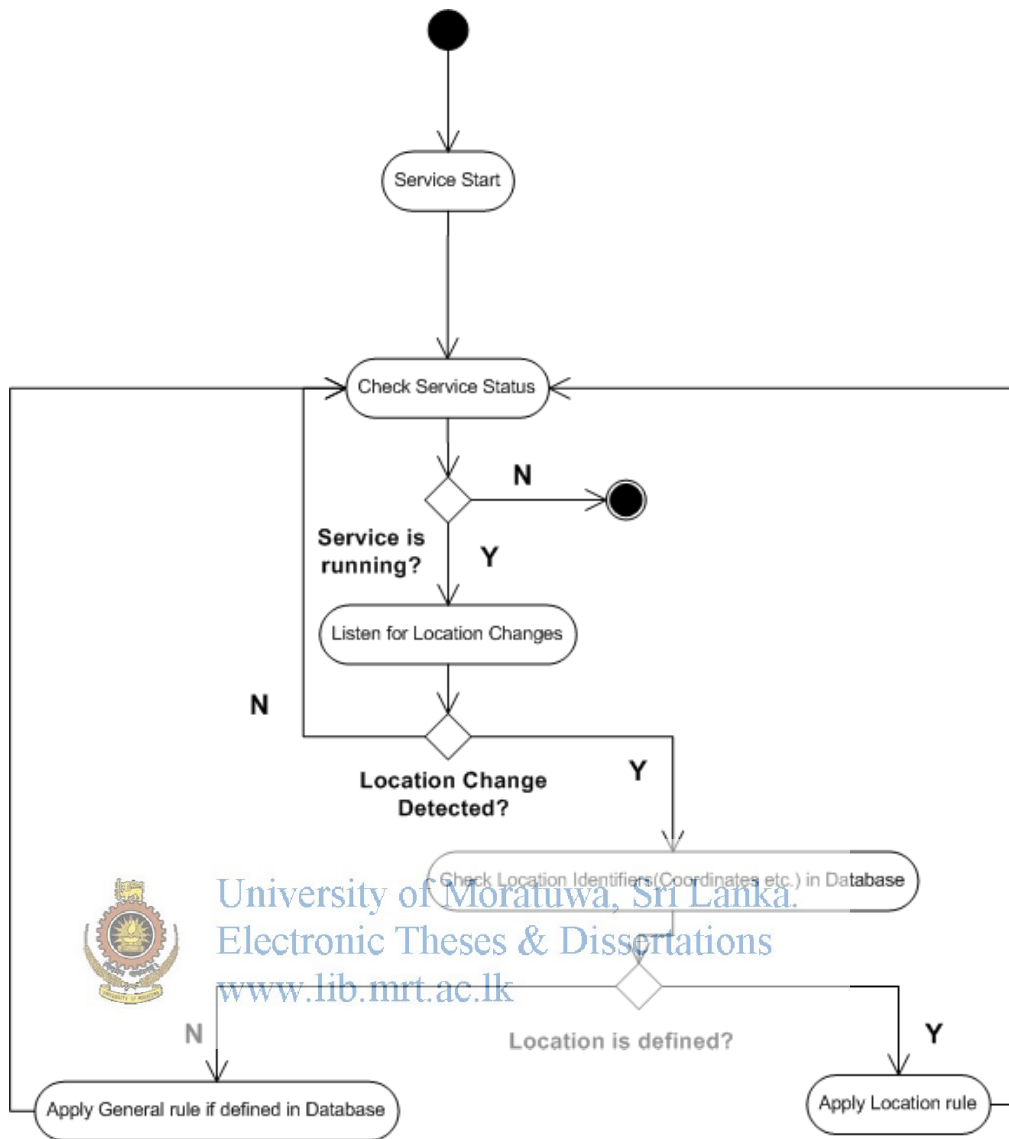


Figure 3.2 - Activity diagram for location context detection in the client

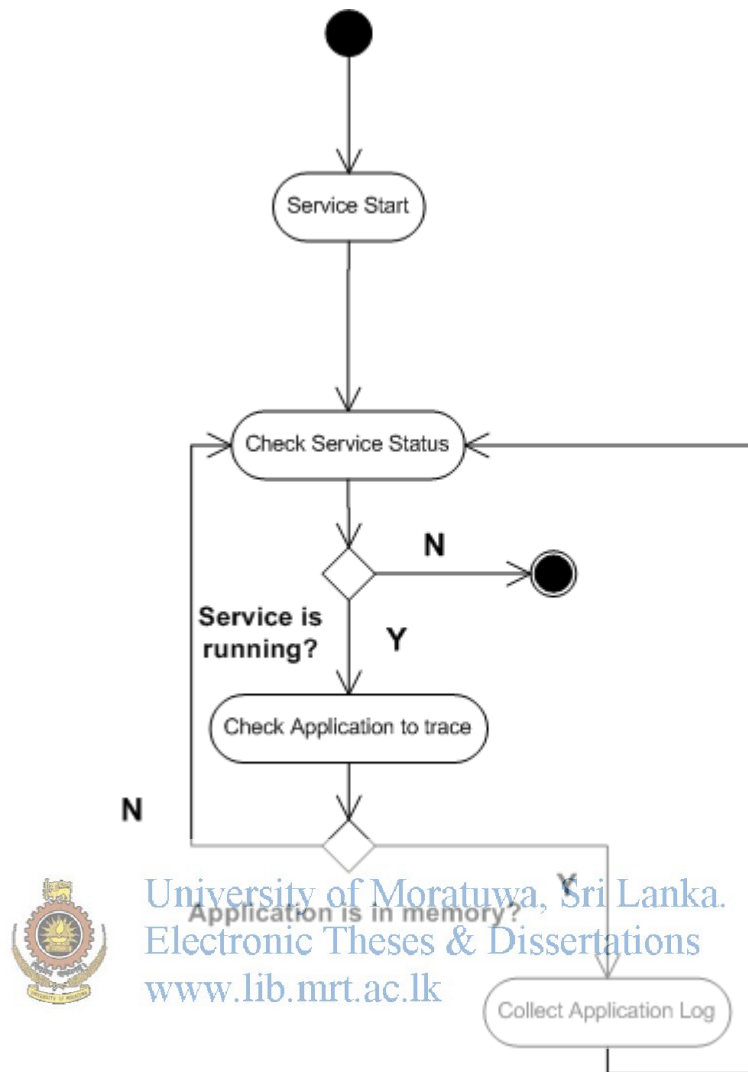
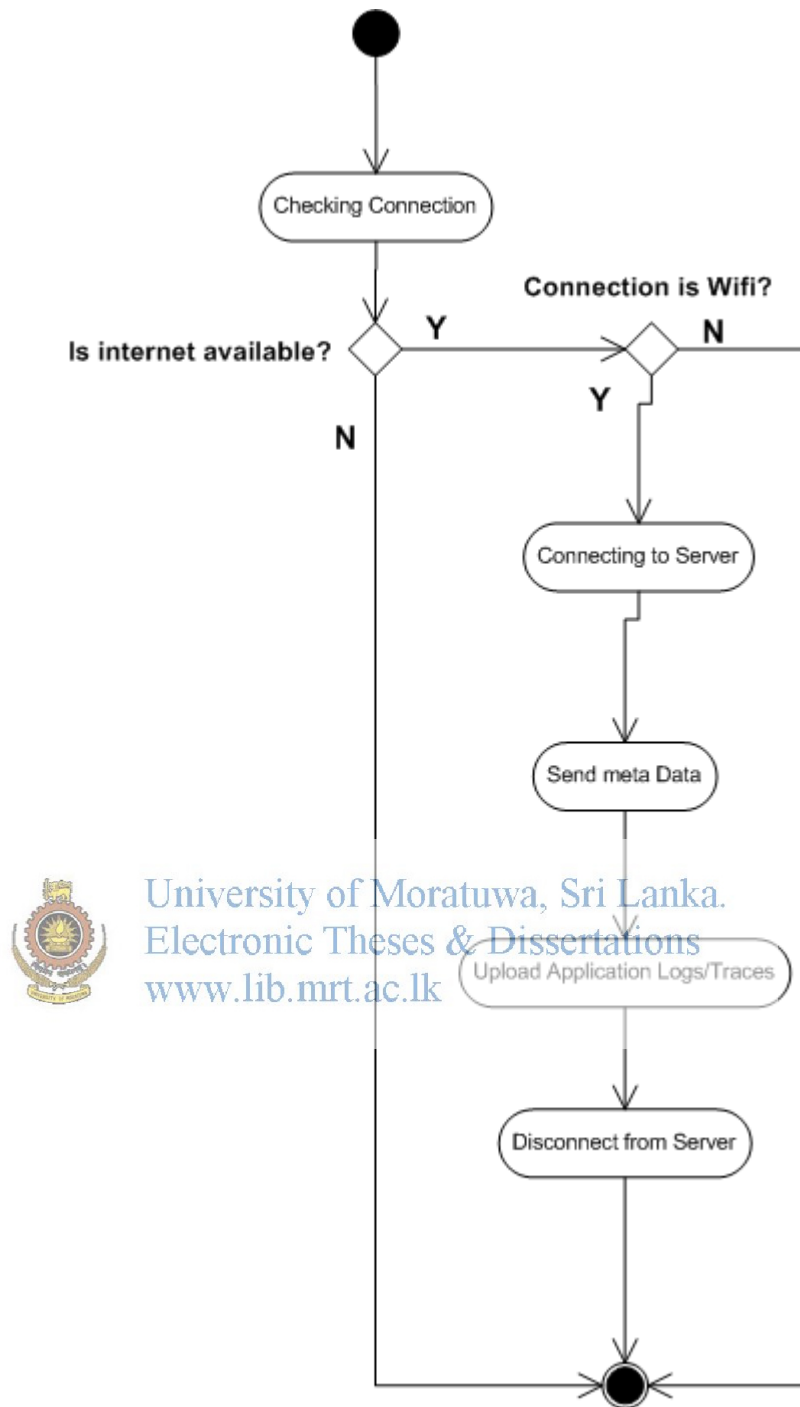


Figure 3.3 - Activity diagram for application log/trace collection in the client

Figure 3.3 elaborates the application log/trace collection happening on the device. The user can select an application to monitor from a list. Once an application is selected, the service component of the application would monitor application launches and would immediately come into action when a designated application is running.



University of Moratuwa, Sri Lanka.
 Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Figure 3.4 - Activity diagram for application log upload to a remote server from the client

Figure 3.4 Elaborates the application log upload activity when it is executed by the user. The client can be configured only to upload the application logs when the device is connected to Wi-Fi. The client software sends a unique identifier and then

application log identification data known as metadata and then uploads application logs.

3.3.2 Class Structure

Client software contains 5 main classes which is designated to carry out different functionalities of the software. These classes are,

BackgroundService Class

ServiceDataSource Class

LocationServicesActivity Class

App TracerControlActivity Class

AppReportActivity Class

BackgroundService Class

This is the main service class of the software. It performs actions such as listening location context changes, logging of applications.

ServiceDataSource Class

This is the data abstraction layer of the software. It provides data retrieval and storage/deleting facilities for upper level classes.

LocationServicesActivity Class

This is the controller class for the entering of location rules. This class provides capability to add/delete location rules.

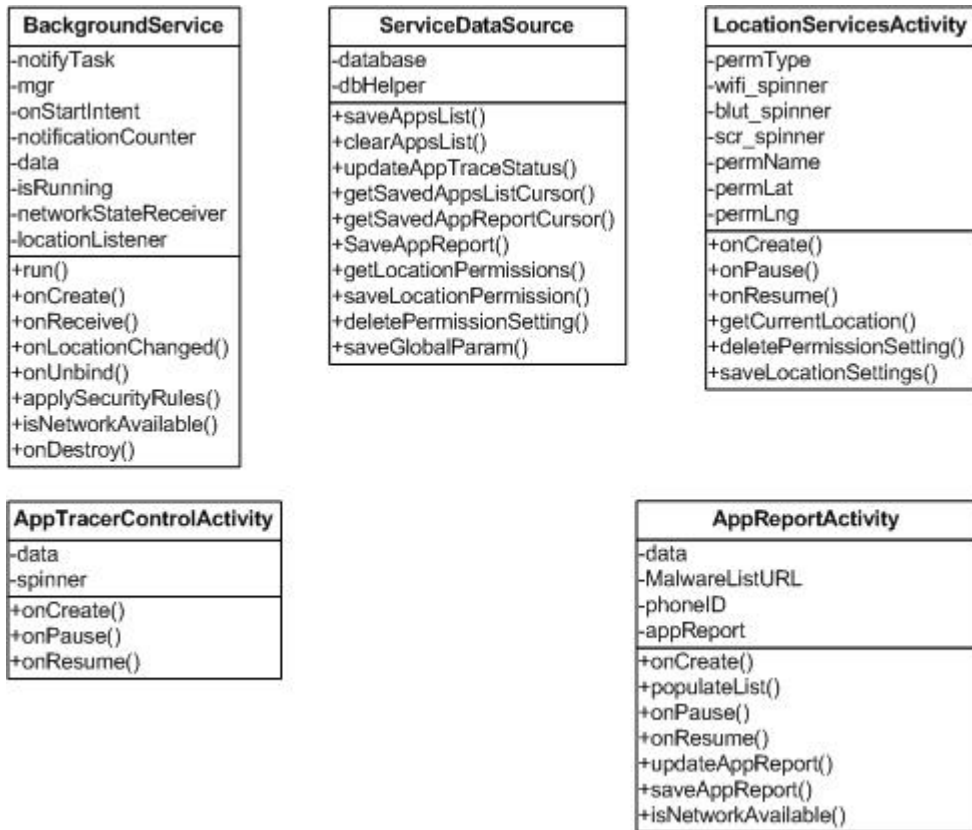
AppTracerControlActivity Class

This is the controller class for the selection of applications for logging.

AppReportActivity Class

This is the controller class for connecting to the server and downloading the application report.

Figure 3.5 shows the class diagrams for above mentioned classes.



3.3.3 Data Storage

The Security System stores information on the SQLite database which comes with the Android Operating System. There are 4 tables used by the client software as shown in the Figure 3.6.

- installed_apps** - Stores currently installed app details.
- location_permissions** - Stores current location permissions
- app_report** - Stores the result of App Report sent from server.
- app_settings** - Stores Global Settings of AndroSec app.

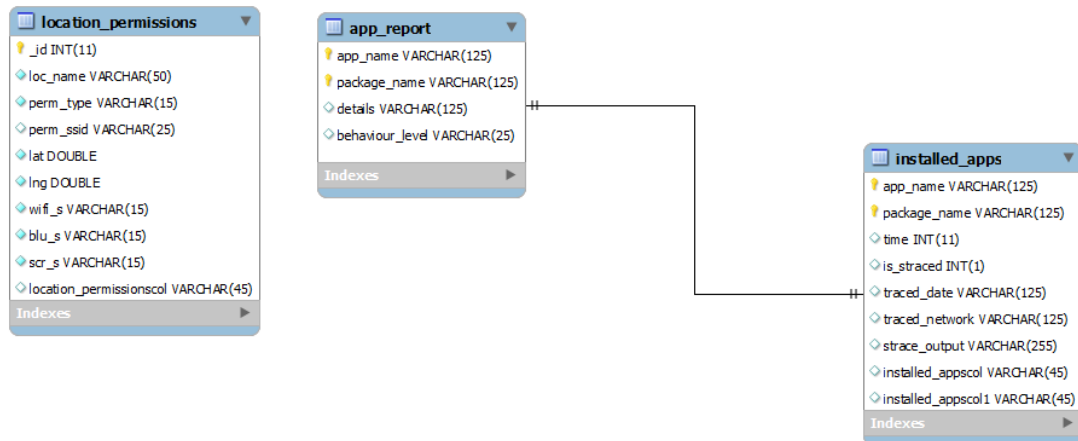


Figure 3.6 - Entity Relationship(ER) diagram of client database

3.3.4 The Background Service

As mentioned previously the background service carries out several background tasks at once. This background service extends the service class of Android. On start of the service several parameters are initialized. These include getting the current location, assigning location update listeners and starting the timers.

3.3.5 Location based Security Policy Management

Once the background service is started, the application listens for location changes ("onLocationChanged" event). The software selects the best sensor out of Network, WiFi and GPS to receive location updates. Location rules are stored in a Sqlite database table. When a new location change event is fired, the system would search the database for any rules that are specific to that location, with a range of 300M. There can be a special catch all rule, denoted by "*", This rule would apply a default set of rules.

The pseudo code of how this works is shown below

```

if Location Change is detected
    Retrieve location rule for the new location
    if Location rule is present
        Apply new location rule
    else
        Apply the default security rule
  
```

Figure 3.7 - Pseudo code for location based security policy management

3.3.6 Application Security Audit

This component, attaches to running processes and collects syscall traces. This is achieved using the "strace" program.

The pseudo code of how this works is shown below

```
if user flagged App is in memory
    Attach to App and intercept system calls
    Log System calls
```

Figure 3.8 - Pseudo code for application security audit

It maintains its own tables that store various information related to application tracing. The application trace is saved as a text file. This application trace contains every system call made by the application.

The collected application traces are then uploaded into a remote server along with Meta-Data related to trace collection. The traced application are downloaded into users device on demand.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

3.3.7 Communication Method

The client uses a push-pull method (active client). The collected app traces are uploaded into a remote server, where it would get processed. The client software running on the mobile device uses HTTP POST requests to upload the files. The server would create folders according to a phone ID also sent by the device. The client would also download the malwares list from server.

The server component stores each devices ID, when a connection is made on the first time. The app traces are stored according to this phone ID.

3.3.8 The Strace Utility

The "Strace" utility is used to gather runtime information about each application. The "Strace" utility used in the software is the generic version available to the ARM

architecture. This utility is available for download at <http://benno.id.au/blog/2007/11/18/android-runtime-strace> [28].

3.3.9 The Client Software User Interfaces

Once the application is launched the following user interface is shown (Figure 3.9), From this interface the user can control the behavior of the application.

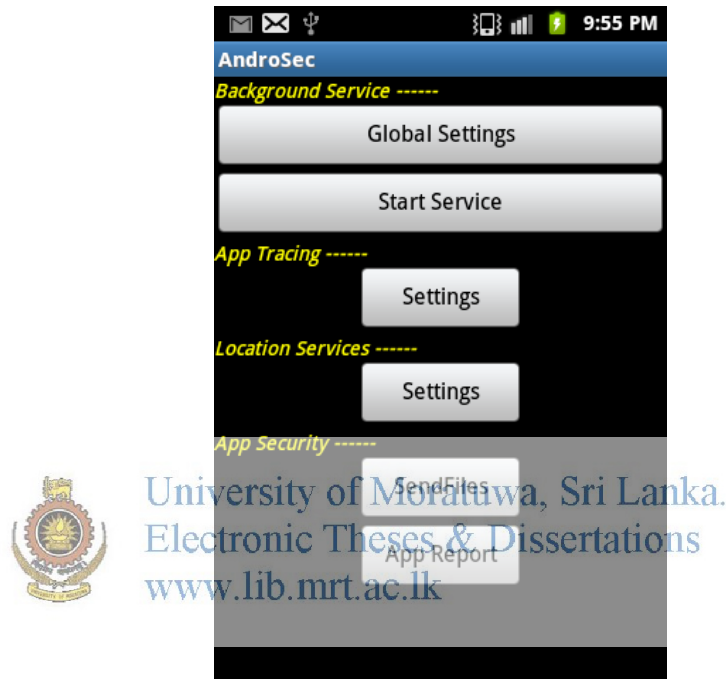


Figure 3.9 - Application home screen

The application has 4 subcomponent categorizations; they are background service, application tracing, location services and application security.

3.3.10 Background Service

Background service component contains two operations, global settings and the service control. This interface allows to input service trace upload path and malware information list download path as well as to read the installed applications list into the system and to clear the read data. The user can star/stop the background service using the controls provided in the interface. Once the service is started it will continue to run even after the user exits the application.

3.3.11 App Tracing

App tracing component allows the user to set the application to be scanned for system calls. Furthermore the screen shows the application which already has scan reports.

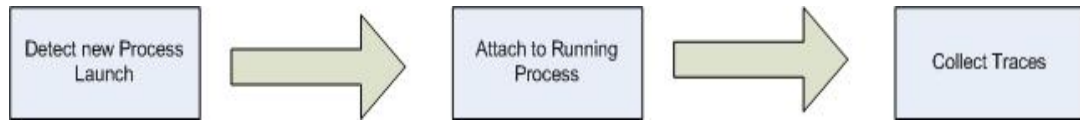


Figure 3.10 - Application system call trace collection

3.3.12 Location Services

This interface allows the user to manage the location based security settings using the form on this interface the user can provide a security rule name; its location coordinates (Longitude, Latitude). The user can get the coordinates automatically or can type in the details. Finally the user can set the status of Wifi, Bluetooth and Screen timeout timings for that particular location. The users can use this interface to view the current location based rules that are already defined, and also to delete any rule.



University of Moratuwa, Sri Lanka
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

3.3.13 Application (App) Security

This category has two actions, one button to upload the application system call (syscall) traces into a remote server and to view application report. The user sends application traces to a remote server with metadata such as the application name, package name, traced date and network information using this interface; furthermore the phone ID given by Android is also sent. Using this interface the user can view the application score assigned by the server according to its behavior. Updating of the report is also possible using this interface. Higher the application score, higher the tendency of the application becoming a malware.

Interfaces corresponding to above mentioned features are shown below,

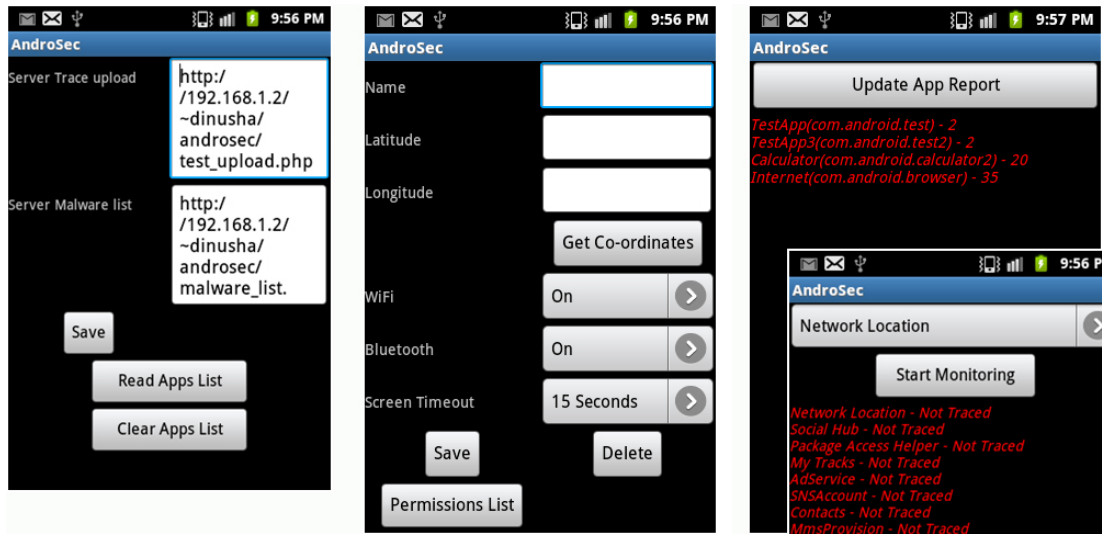


Figure 3.11 - Application interfaces

3.4 The Server Software

Server side software operates as a passive endpoint for client connections. The clients connects and uploads application traces to the server using its unique device ID. The server then stores the application traces as well as stores other details (Metadata) in a database server.

The uploaded application traces are scanned for malicious activity by using a string processing engine which accepts several types of rules. Each application is assigned a cumulative score depending upon the behavior, this is then recorded in a database. The server software is written using PHP and MySQL. The client (device) software uploads application trace files and their metadata into a known location. Meta-data is saved on the MySQL database. The same database contains malware patterns with a scoring.

A PHP program parses the trace files searching for matching malware patterns. When a certain pattern is found a score is assigned to the scanned application, and this score is summed up. This cumulative score is reflected on the database. The

client software downloads the results into the device and presents it to the user, where he/she can decide which applications are malicious according to the score it received.

Each of the behavior rules would contain the type, pattern name, pattern string (if required) and a pattern score. When the parser reads application traces the rule patterns, if present are identified. The parser counts the number of times a particular rule was detected and assigns a cumulative score for each rule which had matched.

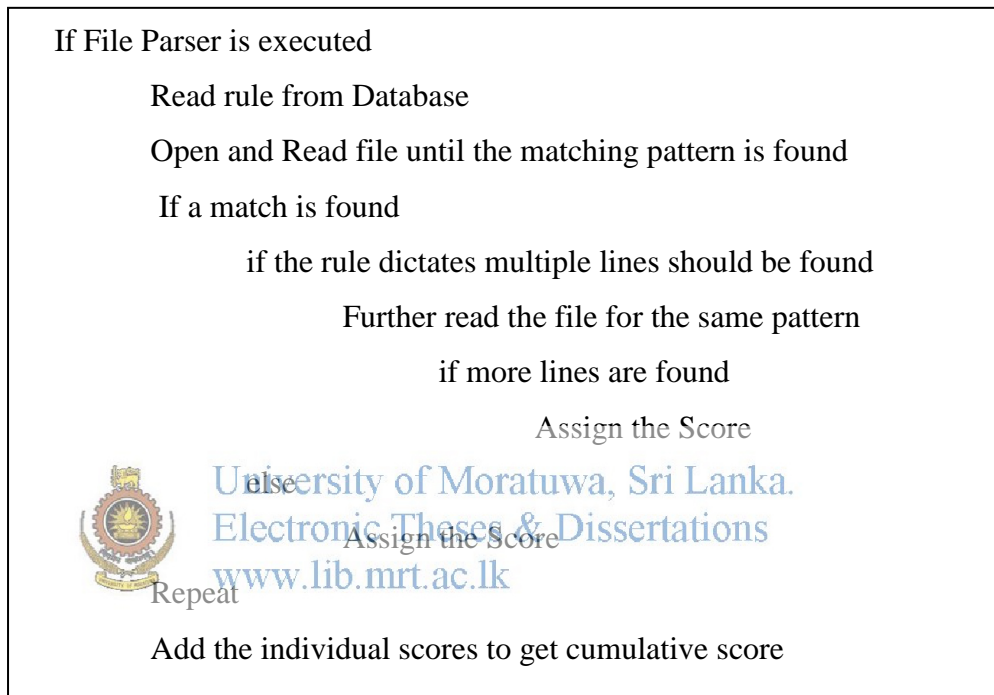


Figure 3.12 - Pseudo code for application security audit

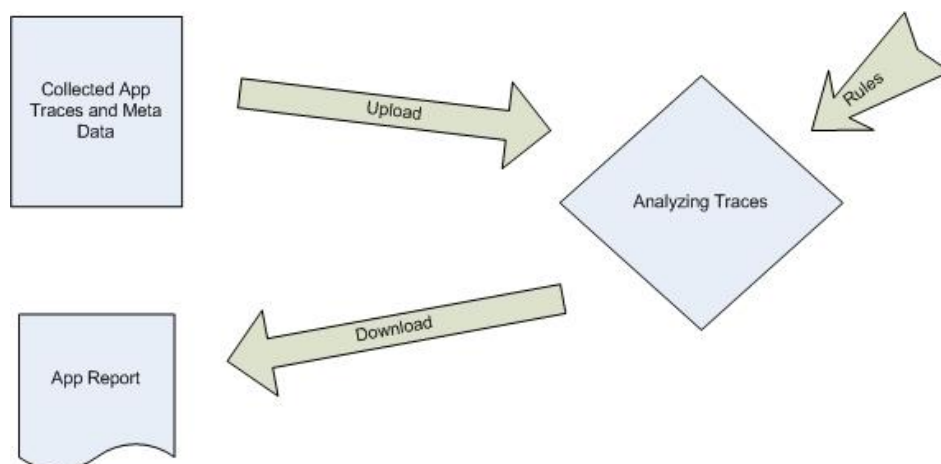
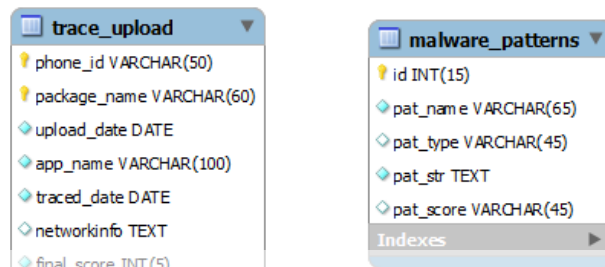


Figure 3.13 - Server application trace processing

3.4.1 Data Storage

When the application traces are uploaded, meta-data regarding each trace collected is also sent to the server, which in turn stores it in the "trace_upload" table. The application behavior is rated using a set of rules each with a score. These application behavior rules are added into the system by an administrative user of the system. These application behavior patterns has attributes such as a name which can be used to refer to the rule, a type field which specifies if the rule is of a syscall , URL or text. Next field is the pattern identification. Finally the score field specifies the score to be assigned.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mru.ac.lk

Figure 3.14 - Entity-Relationship(ER) Diagram of Server Database

3.4.2 Behavior Rule format

Behavior rules needs to have a proper type and format, this is shown below.

Table 3.1: Behavior rule format

| Type | Pattern name | Pattern Type | Pattern String | Pattern Score |
|-------------|--------------|--------------|-----------------------------------|--------------------|
| System call | ANY | syscall | ANY:[Minimum Number] | ANY e.g. 10, 15 |
| URL | ANY | url | URL e.g.:http://www.google.com | ANY e.g. 10, 15 |
| Text | ANY | text | ANY e.g.: contacts list | ANY e.g. 10, 15 |

CHAPTER 4 - TESTING AND RESULTS

After the development of the proof of concept application on the developer environment, Testing of implementation was carried out on several real Android devices. Testing of the developed system were done using custom developed malware since the proper rules for detecting malicious behavior were not available. Security companies censor certain malware features that would be useful for rule generation. Hence in order to test our system we used a combination of rules from research papers and our own intuition. Following sections provide details on the tests and their outcomes.

The test plan included testing the prototype with both malware behaviors and benign behaviors as well as location based security policy management feature. The prototype was evaluated to prove that the system correctly identifies malware behavior from benign behavior as well as the correct application of location based security rules.

The system was populated with two types of rules, location based rules and malware behavior categorization rules. The author categorized detection rules into three groups and had populated the developed system with rules from 2 categories.

As mentioned previously the application traces are uploaded to a remote server for analysis. The server component analyses the app traces and assigns a cumulative score to each application.

4.1 Malware Behavior Rules

The system was initially populated with rules that were found to be among malwares. It is noted that open(), read(), access(), chmod() and chown() are the most used system calls (syscalls) by malware [26]. Furthermore, a list of URL's that were notorious for spreading malwares were also added into the system. It is noted in the paper by Bugera et al. [26] that trojanized apps have a different execution pattern than a benign App. The following tables lists the malware rules that were incorporated into the system.

Table 4.1: Malware detection rules list

| Rule | Rule Type | Rule pattern | Pattern iteration(Min) | Rule Score |
|----------------------------------|------------------|---------------------|-------------------------------|-------------------|
| Connect Syscall | System call | connect | 1 | 10 |
| Socket Syscall | System call | socket | 1 | 10 |
| Getsockname Syscall | System call | getsockbyname | 1 | 10 |
| Bind Syscall | System call | bind | 1 | 10 |
| Read Syscall | System call | read | 1 | 10 |
| Open Syscall | System call | open | 1 | 10 |
| Recv Syscall | System call | recv | 1 | 10 |
| Access Syscall | System call | access | 1 | 10 |
| Chmod Syscall | System call | chmod | 1 | 20 |
| Write Syscall | System call | write | 1 | 20 |
| Close Syscall | System call | close | 1 | 20 |
| Reading "contacts2.db" | String Pattern | contacts2.db | 1 | 35 |
| Executing the "ls" shell command | String pattern | ls | 1 | 35 |

These System calls are infrequently made by common applications, Instead they function as special purpose calls. The systems calls were assigned with arbitrary scores. The cumulative score is not normalized hence it could add up passing 100.

Behavior Rule Scores

After careful consideration it was decided to use a nominal score for system calls that were deemed to be used by every application. Work by Bugera et al. [26] found that there were system calls which were abundant in every application whether it was malware or not. System call patterns which were unique to malware were assigned a considerably higher score. Although having just one or two of these system calls would not identify a harmless application as malware. Applications which tend to act as malware would utilize several pattern behaviors which in turn would generate a higher score.

4.2 Location Rules

A set of sample location rules was added into the system. System was tested using simulated location coordinates. As mentioned previously the system had the accuracy of up to 300M when differentiating a location. The following table shows the location rules that were added into the system.

Table 4.2: Location Rules List

| Rule | Location - Latitude | Location - Longitude | Wifi Status | Bluetooth Status | Screen Timeout |
|--------|---------------------|----------------------|-------------|------------------|----------------|
| Test 1 | 6.85 | 79.87 | ON | ON | 1 Minute |
| Test 2 | 13.32 | 13.32 | ON | OFF | 30 Seconds |
| Test 3 | 23.32 | 13.32 | OFF | OFF | 15 Seconds |
| * | 0 | 0 | OFF | OFF | 15 Seconds |

As shown in the Table 4.2 location rules were entered into the application, the 4th rule(*) is a special rule which speculates a universal rule where if other rules does not apply to a particular location, the this rule can be applied. It is termed as a general location rule.

Android Location Simulation App

A location simulation application was used to mimic location context change. The user can input manually the location co-ordinates and fire a location change event. This new location event can invoke the listening services and these services can take appropriate action.

Android Malware App - Self Written

The custom written malware application features 4 different malware behaviors and 2 benign behaviors. These behaviors can be activated on demand by the user. The list of malware behaviors are explained below,

Malware Behavior 1 - Connect to a Remote Server

This behavior make the application connect to a remote server and push/upload sample data. This behavior is used by malware to send personal data to third parties, unknown to the user. Our application connects to a known server setup by ourselves and uploads/sends sample data.

The calls made by this behavior include connect, socket, getsockname, bind, read, open, recv, access sys calls.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Malware Behavior 2 - Create a File on the Device and Write some Data

Most malwares uses file operations to manipulate/change ownership or read data from the device [26]. Our application tries to mimic the save behavior by creating a file with the ownership to the current application and writing some data into it. We try to show that if the detection works for this behavior, it is possible to have rules for actual malware behavior.

The calls made by this behavior include read, open, write, close sys calls.

Malware Behavior 3 - Reading the Phone contacts List

This also a prominent feature in mobile malware, reading the contacts list of the user and sending the contact numbers and other details to a remote server (contacts list harvesting). These details then can be used for advertising and other purposes. Our app mimics this behavior by reading the contacts list. We aim to prove the security rules populated would be able to detect this behavior.

The calls made by this behavior include read, open, recv system calls and reads the "contacts.db" file.

Malware Behavior 4 - Executing a Shell Command

This is also a common behavior in malware; malwares execute various commands on the device. Our application also mimics this behavior. The application executes a simple command on the device.

The calls made by this behavior include read, open, recv, access sys calls.

Benign Behavior 1 - Displaying text

This behavior when run, displays/changes a text field in the interface. By using this type of behavior we aim to differentiate the characteristics of benign applications and malware apps.

Benign Behavior 2

This application adds two numbers and shows the result in a text field. As mentioned earlier we use this behavior as a base case for benign apps.

4.3 Results

4.3.1 Malware Detection System



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

Table 4.3: Cumulative score for malware behaviors

| Behavior | Cumulative Score |
|--------------------|-------------------------|
| Malware Behavior 1 | 120 |
| Malware Behavior 2 | 90 |
| Malware Behavior 3 | 95 |
| Malware Behavior 4 | 85 |
| Benign Behavior 1 | 50 |
| Benign Behavior 2 | 50 |

After the initial runs the resulting cumulative score is shown in Table 4.3. The above results confirm that by having proper detection rules malware applications can be differentiated from benign applications.

4.3.2 Location Aware Security System

Due the difficulty of testing each rule without actually being on the given location, It was decided to use a GPS mockup application which would generate mock location changes on the device just as a real GPS Device. The developed security application would then intercept these location changes and applies the corresponding security rules for that location.

The application successfully detected the location changes and applied the security rules with 100% accuracy.

In order to test our system furthermore we have run tests for applications that are developed by 3rd parties. The rule set which was used on previous tests were used without any modifications. Some of these applications were already installed on the mobile device by default and others had to be downloaded using the Google App store called Play.

Testing with Real Apps

Since the results obtained from custom written malware behaviors were successful. The system was tested with real applications. Through not malware, it was anticipated that the system was able to identify malware like behaviors from benign behaviors. As mentioned earlier malware rules that were populated in the system were able to detect behaviors such as connecting to external servers, changing and writing to files on the device. Therefore it is anticipated that the system would assign higher scores to applications which show these types of behaviors. We have analyzed the following applications, which were selected due to availability and popularity.

- Calendar
- Clock
- Angry Birds
- TED

Calendar Application

This is the default calendar application which comes with Android. The Calendar application provides users with the ability to add events and view past events along with the calendar for each month. It comes installed on the devices by default.

Clock Application

This is the default clock application which comes with Android. It is already installed on the devices. It does not have any special features apart from displaying time.

Angry Birds Application

This is a very popular gaming application. Once the game is launched it takes user through different levels with varying difficulty levels. When a certain level is successfully ended, a score is assigned to the player. During the game play advertisements are shown. This application needs to be downloaded from the Google App store (Play Store) and is freely available without a cost.

TED Application

This is the official TED application for viewing Videos of TED website. TED website gathers research level content from scientists around the world. The application for Android which is downloadable from the Google App Store (Play Store) and can stream videos into devices. It connects to a remote server for video streaming and to fetch information.

Application Trace Analysis

After collecting the application traces on the device it was uploaded to a remote server setup. Each of the application traces were then analyzed by the server software. Each application trace were analyzed thrice (3) and the resulting scores were averaged.

Results

After collecting the above discussed application traces and scanning through the system call patterns the following results were obtained.

Table 4.4: Results for real applications

| App | Cumulative Score |
|-------------|-------------------------|
| Calendar | 60 |
| Clock | 70 |
| Angry Birds | 120 |
| TED | 90 |

Both Calendar and Clock applications gained a score of 60 and 70 respectively. While Angry Birds and the TED App gained scores of 120 and 90 respectively. Both Calendar and Clock applications had read, open write and close system calls in their traces while Clock App had an extra access syscall.

Angry Birds and TED applications had shared read, open write and close system calls in their traces as well as had other system calls, since the two apps connected to remote servers. These system calls include chmod, connect, socket, bind, recv, access, getsockname.

Although the tested applications were not malwares the system identified according to the system calls made, applications which connects to external servers and assigned them a higher cumulative score. In this case if we assigned a boundary score of 70, then applications which connects to external servers and does file system changes would be getting a higher cumulative score than 70 and hence could be classified as suspicious.



University of Moratuwa, Sri Lanka.
Electronic Theses & Dissertations
www.lib.mrt.ac.lk

CHAPTER 5 - CONCLUSIONS AND FUTURE WORK

The outcome of this thesis is to provide a new dimension into malware detection using their behavior as well as to have a security system which adapts to the change in location. Various researchers have shown that for mobile devices the stealing of personal data is far more damaging than anything else since people tend to see their mobile device as a personal tool and hence put sensitive data on the device.

The two main obstacles which prevents the implementation of a comprehensive security system is the limited battery power of the mobile device as well as the limited computing ability of majority of mobile devices, hence the proof of concept was designed as a client - server solution. The client component reside on the mobile device capturing the behavior of apps and managing the changing Location based security, while the server component does the processing of the app behavior and assigns a score. The client connects to the server to upload the app traces and to download the score for each traced app.



University of Moratuwa, Sri Lanka.

Electronic Theses & Dissertations

www.lib.mrt.ac.lk

The client component was successfully developed and deployed on a real mobile device running Android OS version 2.3 (Gingerbread). The server component was hosted using a public hosting server which was accessible using the Internet. The server component was written using PHP with MySQL database and the client component was written using java targeted for Android.

The adaptations of security according to the devices location were successfully demonstrated. This security feature also helped to save the power consumption by switching off communication features of the device like Bluetooth and Wi-Fi when in a public area. The malware detection mechanism utilizes a behavior identification method and assigning a score to each app according to its behavior when run. This makes the malware detection efficient since unlike traditional malware detection techniques using signature patterns; this technique uses a set of generalized rules to detect malwares according to their behavior.

The developed malware detection scheme addressed several shortcomings of mobile security systems. The test results showed 100% accuracy, however in order to test for real malware a comprehensive study on the low level behaviors of malware apps needs to be carried out separately. This is due to the fact that there is no study done to analyze the low level system calls of each behavior e.g. sending SMS, reading the contacts list etc. In order to test the developed system we developed a malware app with different malware behaviors, furthermore the app contained several benign behaviors to show that it can differentiate between the two types of behavior. All the behaviors were activated manually. The server was populated with known behavior rules.

The proof of concept application that was developed in this report is far from a consumer level application. This software can be made much more user friendly and attractive. The concepts discussed can be refined more. There isn't a proper survey done to investigate the system call patterns of most malware. This would have helped immensely in populating proper malware behavior rules. Currently the user has to select which applications to scan and calculate the scores. This process can be automated where the system would make a list of applications to be scanned and the traces would be collected automatically once an application in the list starts to run.



REFERENCES

- [1] I-Lung Kao, Global Strategist, IBM Security Services, “Securing mobile devices in the business environment,”[Online]. Available: www-935.ibm.com/services/uk/en/attachments/pdf/Securing_mobile_devices_in_the_business_environment.pdf [Accessed: 19-Feb-2012]
- [2] ITU, “The World in 2009,”[Online]. Available: www.itu.int/ITU-D/ict/material/Telecom09_flyer.pdf [Accessed: 19-Feb-2012]
- [3] P Silva, “Secure iPhone Access to Corporate Web Applications,”[Online]. Available: <http://www.f5.com/pdf/white-papers/secure-iphone-access-tb.pdf> [Accessed: 19-Feb-2012]
- [4] “Symantec Consumer Guide to Wireless Device Security.”[Online]. Available: <http://spotlight.getnetwise.org/wireless/wirelessguide.pdf> [Accessed: 19-Feb-2012]
- [5] M. Baldaun, S. Dustdar, and F. Rosenberg, “A survey on context-aware systems,” *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, Jan. 2007.
- [6] Google, “Platform Versions.”[Online]. Available: <http://developer.android.com/resources/dashboard/platform-versions.html>. [Accessed: 19-Feb-2012]
- [7] “Juniper Networks Inc. Malicious mobile threats report 2010/2011.” Technical report, Juniper Networks, Inc., 2011.
- [8] J. Hong, E. Suh, and S. Kim, Context-aware systems: A literature review and classification. 2009.

- [9] K. W. Y. Au, Y. F. Zhou, Z. Huang, P. Gill, and D. Lie, "Short paper: a look at smartphone permission models," New York, NY, USA, 2011, pp. 63–68.
- [10] N. Husted, H. Saïdi, and A. Gehani, "Smartphone security limitations: conflicting traditions," New York, NY, USA, 2011, pp. 5–12.
- [11] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," in Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, London, UK, 1999, pp. 304–307.
- [12] E. Snekkenes, "Concepts for personal location privacy policies," New York, NY, USA, 2001, pp. 48–57.
- [13] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati, "Supporting location-based conditions in access control policies," New York, NY, USA, 2006, pp. 212–222.
- [14] M. J. Covington, P. Fogla, Z. Zhan, and M. Ahamad, "A context-aware security architecture for emerging applications," in Computer Security Applications Conference, 2002. Proceedings. 18th Annual, 2002, pp. 249–258.
- [15] J. Oberheide and F. Jahanian, "When mobile is harder than fixed (and vice versa): demystifying security challenges in mobile environments," in Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications, New York, NY, USA, 2010, pp. 43–48.
- [16] J. Al-Muhtadi, A. Ranganathan, R. Campbell, and M. D. Mickunas, "Cerberus: a context-aware security scheme for smart spaces," in Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003), 2003, pp. 489–496.

- [17] D. Barrera and P. Van Oorschot, "Secure Software Installation on Smartphones," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 42–48, Jun. 2011.
- [18] "Google Inc. Android market." [Online]. Available: <https://market.android.com/>. [Accessed: 02-Feb-2012]
- [19] M. Lange, S. Liebergeld, A. Lackorzynski, A. Warg, and M. Peter, "L4Android: a generic operating system framework for secure smartphones," New York, NY, USA, 2011, pp. 39–50.
- [20] K. Gudeth, M. Pirretti, K. Hoeper, and R. Buskey, "Delivering secure applications on commercial mobile devices: the case for bare metal hypervisors," New York, NY, USA, 2011, pp. 33–38.
- [21] G. Bai, L. Gu, T. Feng, Y. Guo, and X. Chen, "Context-Aware Usage Control for Android," in *Security and Privacy in Communication Networks*, vol. 50, S. Jajodia and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 326–343.
- [22] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security," *IEEE Security & Privacy*, vol. 7, no. 1, pp. 50–57, Feb. 2009.
- [23] Timothy Vidas, Nicolas Christin, and Lorrie Cranor, "Curbing Android Permission Creep," In *Proceedings of the 2011 Web 2.0 Security and Privacy Workshop (W2SP 2011)*. Oakland, CA. May 2011.
- [24] Timothy Vidas, Daniel Votipka, and Nicolas Christin, "All Your Droid Are Belong to Us: A Survey of Current Android Attacks.," In *Proceedings of the 5th USENIX Workshop on Offensive Technologies (WOOT '11)*. San Francisco, CA. August 2011.

- [25] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," in Proceedings of the 9th USENIX conference on Operating systems design and implementation, Berkeley, CA, USA, 2010, pp. 1–6.
- [26] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," New York, NY, USA, 2011, pp. 3–14.
- [27] US-CERT, "Cyber threats to mobile Phones." [Online]. Available: http://www.us-cert.gov/reading_room/cyber_threats_to_mobile_phones.pdf. [Accessed: 12-Feb-2012]
- [28] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "'Andromaly': a behavioral malware detection framework for android devices," Journal of Intelligent Information Systems, Jan. 2011.
- [29] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for Android," New York, NY, USA, 2011, pp. 15–26.
- [30] "Gingerbread most popular Android flavor at 56 percent market share," CNET. [Online]. Available: http://news.cnet.com/8301-1035_3-57524230-94/gingerbread-most-popular-android-flavor-at-56-percent-market-share/. [Accessed: 2-Nov-2012].
- [31] "Strace for Android," Android - strace runtime, 05-Apr-2012. [Online]. Available: <http://benno.id.au/blog/2007/11/18/android-runtime-strace>. [Accessed: 05-Apr-2012].
- [32] M. La Polla, F. Martinelli, and D. Sgandurra, "A Survey on Security for Mobile Devices," IEEE Communications Surveys Tutorials, vol. 15, no. 1, pp. 446–471, First 2013.
- [33] Ramu, S. "Mobile Malware Evolution, Detection and Defence", April 2012.