Acknowledgement

In our endeavour to achieve the success in completing project report on **“PRIVACY PRESERVATION IN ANDROID SMARTPHONES”** in the Final Year Engineering in Information Technology. We take this opportunity to express our deep sense of gratitude to our guide respected **Prof. N.R.Jain**, for his valuable guidance and kind co-operation throughout the period of work has undertaken which has been instrumental in success of project. I especially thankful to our project in charge **Prof.D.S.Hirolikar**, she have also guided us much for preparation of each and every project work. We also very thankful to our respected H.O.D. **Prof. N.R.Jain**, for providing us with adequate facilities, ways and means by which I was able to complete this project report. We would also like to thank our respected Principal **Prof. D.O Patil Sir** who creates a healthy environment for all of us to learn in best possible way. We express our thankfulness to all teachers and staff of computer department for timely help in course of project report. Finally, special thanks to my friends, family members and all others those unseen people across the internet for maintaining sources on the internet that helped us in the successful completion of this work.

GANESH S. KAREWAD PRIYA N. PHADKE SAGAR A. MANDAVE

Abstract

In 21st century Mobile became one of our primary needs. Everyone uses smartphones nowadays. Security is one of main concerns of Smartphone users today. Apps can theft user’s data. Smartphones are becoming some of our most trusted computing devices. People use them to store highly sensitive information including email, passwords, financial accounts, and medical records. These properties make smartphones an essential platform for privacy-preserving applications. To date, this area remains largely unexplored mainly because privacy-preserving computation protocols were thought to be too heavy weight for practical applications. Users don’t know how to deal with that kind of apps. We are suggesting some changes that can enhance user’s privacy preservation. A user simply relies on phones OS for protection. We studied few Smartphone OS and observed that mobile OS don’t provide much security for private data like contacts and log info. Android is mostly used Smartphone OS. We are suggesting a system that will monitor applications running on the system or device. Permissions give power to apps to misuse apps. Our proposed system will study the permission structure and try to identify threats to privacy. We have studied 100 android apps and recovered their source code to study behavior of apps. We found that app can misuse information only if it has regarding permission. Real time monitoring helps to understand if app uses its permission for valid reason.

This app will assist user for securing his/her data from being theft by users. The app will notify user when an app is asking for excess permission. The app also should monitor for activities of running apps. It should notify user if an app is trying to access sensitive data files or trying to send data files to any other device.

Chapter 1

1. Introduction

Android has the biggest market share among all Smartphone operating system. Security is one of the main concerns for Smartphone users today. People use them to store highly sensitive information including email, passwords, financial accounts, and medical records. These properties make smartphones an essential platform for privacy-preserving applications. We are suggesting a Monitoring App. This app will assist user for securing data from being theft by Apps. The app will notify user when an app is asking for excess permission. The app also should monitor for activities of running apps. It should notify user if an app is trying to access sensitive data files or trying to send data files to any other device. The limitations of smartphones provide a number of challenges for building such applications.

We introduce the issues that make smartphones a unique platform for secure computation, identify some interesting potential applications, and describe our initial experiences creating privacy-preserving applications on Android devices.

1.1 Overview

Some features and benefits offered by this application are:

1. Application is compatible on any android version.
2. To help user to protect private data.
3. Applications is providing guide to the user.
4. This application is very easy to use.

1.2 Brief Description

Android is a modern mobile platform that is designed to be truly open source. Android applications can use advanced level of hardware and software, as well as local and server data, exposed through the platform to bring innovation and value to consumers. Android platform must have security mechanism to ensure security of user data, information, application and network.

1. Open source platform needs strong and rigorous security architecture to provide security. Android is designed with multi-layered security that provides flexibility needed for an open platform, whereas providing protection for all users of the platform designed to a software stack, android includes an operating system, middleware and core application as a complete.
2. Android powers hundreds of millions of mobile devices in more than 190 countries around the world. Android architecture is designed with keep ease of development ability for developers. Security controls have designed to minimize the load on developers. Developers have to simply work on versatile security controls. Developers are not familiar with securities that apply by defaults on application.
   1. Android is not secure as it appear, even when such robust security measures. There are several security problems faced by the android, some of them are mentioned below.
3. Android has no security scan over the apps being uploaded on its market.
4. There are some apps which can exploit the services of another app without permission request.
5. Android’s permission security model provides power to user to make a decision whether an app should be trusted or not. This human power introduces a lot of risk in Android system.
6. The Open Source is available to legitimate developers as well as hackers too. Thus the Android framework cannot be trusted when it comes to develop critical systems.
7. The Android operating system developers clearly state that they are not responsible for the security of external storage.
8. Any app on the android platform will access device data just like the GSM and SIM marketer Ids while not the permission of the user.

1.3 Applying software engineering approach

Incremental Model:

The product is decomposed into a number of components, each of which are designed and built separately (termed as builds). Each component is delivered to the client when it is complete. This allows partial utilization of product and avoids a long development time. It also creates a large initial capital outlay with the subsequent long wait avoided. This model of development also helps ease the traumatic effect of introducing completely new system all at once. There are, overall, few problems with this model. The incremental build model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping. The product is decomposed into a number of components, each of which are designed and built separately (termed as builds). Each component is delivered to the client when it is complete. The Allows partial utilization of product and avoids a long development time. It also creates a large initial capital outlay with the subsequent long wait avoided. This model of development also helps ease the traumatic effect of introducing completely new system all at once. There are, overall, few problems with this model.

Advantages:

Any faulty piece of software can be identified easily as very few changes are done after every iteration.

It is easier to test and debug as testing and debugging can be performed after each iteration.

This model does not affect anyone’s business values because they provide core of the software which customer needs, which will indeed help that person to keep run his business.

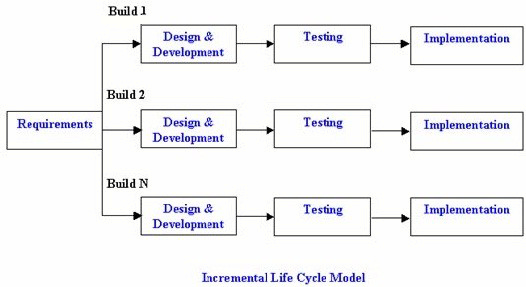


Figure 1.1: Incremental model

Disadvantages:

1. If the requirements initially were thought to be stable but at later stages are realized to be unstable then the increments have to be withdrawn and have to be reworked.
2. Resulting cost may exceed the cost of the organization. Problems may arise related to system architecture.

Chapter 2

2. LITERATURE SURVEY

### Privacy-Preserving Applications on Smartphones

#### Authors: Yan Huang, Peter Chapman, David Evans.

#### Summary:

Smartphones are becoming some of our most trusted computing devices. People use them to store highly sensitive information including email, passwords, financial accounts, and medical records. These properties make smartphones an essential platform for privacy-preserving applications. In this paper, they introduce the issues that make smartphones a unique platform for secure computation, identify some interesting potential applications, and describe our initial experiences creating privacy-preserving applications on Android devices.

This paper explores possibilities of using smartphones for privacy preservation. Despite of low computation power of device publisher of this ppr thinks that it is possible to develop an application for privacy preservation. Smartphones are considered as unique platform for secure computation. Paper includes comments on publisher’s initial experience on creating privacy preserving application on android devices.

#### Disadvantages:

#### Need good hardware support.

#### Their code uses many BigInteger objects, which is an immutable datatype provided by the Java API, so they observe substantial footprint of the JVM’s garbage collection thread.

### Review on Android and Smartphone Security

#### Authors: Tiwari Mohini, Srivastava Ashish Kumar, Gupta Nitesh.

#### Summary:

Android has the biggest market share among all Smartphone operating system. Security is one of the main concerns for Smartphone users today. As the power and features of Smartphone’s increase, so has their vulnerability for attacks by viruses etc. Perhaps android is more secured operating system than any other Smartphone operating system today. Android has very few restrictions for developer, increases the security risk for end users. In this paper we have reviewed android security model, application level security and security issues in the Android based Smartphone.

This paper lights security flaws of android security system. According to this paper android is most secure OS. Reason for threats is that android developers have very few restrictions resulting unethical backdoor exploitation of users data. Users don’t pay much attention to these things resulting violation of privacy.

#### Disadvantages-

1. They have not given any solution for this project. As they should have given some solutions.
2. Android platform provides all security features, but there will always be a risk if the user will install suspicious apps or allow permission to an app without paying attention.
   1. Privacy control in smart phones using semantically rich reasoning and context modelling.

#### Authors: Dibyajyoti Ghosh, Anupam Joshi, Tim Finin and Pramod Jagtap.

#### Summary:

They present our on-going work on user data and contextual privacy preservation in mobile devices through semantic reasoning. Recent advances in context modelling, tracking and collaborative localization has led to the emergence of a new class of smartphone applications that can access and share embedded sensor data. Unfortunately, this also means significant amount of user context information is now accessible to applications and potentially others, creating serious privacy and security concerns. Mobile OS frameworks like Android lack mechanisms for dynamic privacy control. They show how data flow among applications can be successfully filtered at a much more granular level using semantic web driven technologies that model device location, surroundings, application roles as well as context-dependent information sharing policies.

In this paper publisher wrote about lack of security mechanism in smartphones for securing user data from malicious apps. Normally mobile os should have mechanism that prevents malicious apps from unauthorized actions. Paper discusses how data flow among applications can be successfully filtered at a much more granular level using semantic web driven technologies that model device location, surroundings, application roles as well as context-dependent information sharing policies.

* 1. Improving Efficiency and Security Based Data Sharing in Large Scale Network.

Authors: B. SakthiSaravanan, R.Dheenadayalu, A.Vijayaraj.

#### Summary:

The Key Generation Center (KGC) could decrypt any messages addressed to specific users by generating their private keys. This is not suitable for data sharing scenarios where the data owner would like to make their private data only accessible to designated users key. To overcome this problem we propose escrow problem which means a written agreement delivered to a third party and Attribute-Based Encryption (ABE). Attribute-based encryption is a promising cryptographic approach, is a fine-grained data access control which provides a way of defining access policies based on different attributes of the requester, environment and the data object. The KGC can decrypt every cipher text addressed to specific users by generating their attribute keys. This could be a potential threat to the data confidentiality or privacy in the data sharing systems.

This paper discusses about different security mechanisms for data transmission over network. Encryption and decryption of data is discussed. Data should be safe while transmitting over network. Android explicitly doesn’t use any encryption for data. Data security resides on applications resulting vulnerabilities.

#### Disadvantages-

1.  There can be potential threat to the data confidentiality or privacy in the data sharing systems.

2.5 Generating random regions in Spatial cloaking algorithm for location privacy preservation.

Authors: Mrs. Suchita R. Shastry, Prof Dr.P.K.Deshmukh, Prof.Dr.A.B.Bagwan.

#### Summary:

Location based Servers (LBS) include services to identify a location. Network users expect to access services relevant to their locations, and preserve their privacy without disclosing their exact location. The spatial cloaking method provides the solution where exact location of user gets hazy. In peer to peer network (P2P), communication between the peers becomes time consuming and communication overhead. In this paper we have proposed the method where instead of communicating with peers, user directly communicates with LBS. we have presented two algorithms where first algorithm which where the LBS provide the direct list of in ascending order. The second algorithm for query processing generates the region of different shapes which minimizes the chances of getting the user disclosed to adversary.

Location is one of the very private things for user. Apps can access user’s location. Misuse of this information can be done. In this paper concept of cloaking is discussed for extending some privacy for user. Creating some cloaks gives privacy to user. A region can be decided as per need. These kinds of methods are used by militaries to give low access on gis data to public. Different apps which serve transportation facility use this information. So these apps should be excluded from this mechanism so that user can have uninterrupted service.

#### Disadvantages-

#### The Algorithm needs a high End Device.

#### Does not work on each and every device.

#### 2.6 A Large-Scale Study of Mobile Web App Security

#### Author: Patrick Mutchler, Adam Doup´ey, John Mitchell, Chris Kruegelz and Giovanni Vignaz

#### Summary:

Mobile apps that use an embedded web browser, or mobile web apps, make up 85% of the free apps on the Google Play store. The security concerns for developing mobile web apps go beyond just those for developing traditional web apps or mobile apps. In this paper we develop scalable analyses for finding several classes of vulnerabilities in mobile web apps and analyse a large dataset of 998,286 mobile web apps, representing a complete snapshot of all of the free mobile web apps on the Google Play store as of June 2014. We find that 28% of the studied apps have at least one vulnerability. We explore the severity of these vulnerabilities and identify trends in the vulnerable apps. We find that severe vulnerabilities are present across the entire Android app ecosystem, even in popular apps and libraries. Finally, we offer several changes to the Android APIs to mitigate these vulnerabilities.

2.7 Application Security framework for Mobile App Development in Enterprise setup

Authors: D. P. Acharjya, Sugata Sanyal.

Summary:

Enterprise Mobility has been increasing the reach over the years. Initially Mobile devices were adopted as consumer devices. However, the enterprises world over have rightly taken the leap and started using the ubiquitous technology for managing its employees as well as to reach out to the customers. While the Mobile ecosystem has been evolving over the years, the increased exposure of mobility in Enterprise framework have caused major focus on the security aspects of it. While a significant focus have been put on network security, this paper discusses on the approach that can be taken at Mobile application layer, which would reduce the risk to the enterprises.

2.8 IccTA: Detecting Inter-Component Privacy Leaks in Android Apps

Authors: Li Li, Alexandre Bartel, Tegawende. Bissyande, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, Patrick McDaniel

Summary:

Shake Them All is popular “Wallpaper” application exceeding millions of downloads on Google Play. At installation, this application is given permission to access the Internet (for updating wallpapers) and use the device microphone (to change background following noise changes).

With these permissions, the application could silently record user conversations and upload them remotely. To give more confidence about how Shake Them All actually processes what it records, it is necessary to build a precise analysis tool that tracks the flow of any sensitive data from its source point to any sink, especially if those are in different components.

Since Android applications may leak private data carelessly or maliciously, we propose IccTA, a static taint analyser to detect privacy leaks among components in Android applications. IccTA goes beyond state-of-the-art approaches by supporting inter component detection. By propagating context information among components, IccTA improves the precision of the analysis. IccTA outperforms existing tools on two benchmarks for ICC-leak detectors: Droid Bench and ICC-Bench. Moreover, our approach detects 534 ICC leaks in 108 apps from MalGenome and 2,395 ICC leaks in 337 apps in a set of 15,000 Google Play apps.

Chapter 3

1. Requirement and Analysis

3.1 PROBLEM STATEMENT

Android Operating System doesn’t provide any special security architecture for data privacy. Malicious apps can access user’s private data. Apps can misuse permissions to gain access to privileged data. A new system should be developed that can help user to keep data private and secure mobile phone from malicious apps.

3.2 REQUIREMENT ANALYSIS

3.2.1 Introduction

Following Section describes the detailed software requirement specification.

3.2.2 Purpose and Scope Documentation

* + - 1. Purpose

#### The Purpose of the software requirement specification was done to help user to protect private data by:

#### Preventing Apps from misuse of permission.

#### Providing Guide on data privacy for user.

#### Project Scope

#### Nowadays smartphones are being used by everyone.

#### 70 % smartphone are Android phones.

#### There are thousands of malicious.

#### 3.2.3 Usage Scenario

The main purpose of the use case diagram is to make the user understand the basic functionality of particular project.

The use case describes how the user uses the application to monitor the Applications.

3.3 Use Cases

Use case diagrams are a set of use cases, actors and their relationships. They represent the use case view of a system.

A use case represents a particular functionality of a system.

A single diagram can be used to depict all the use cases in the system. List of actors and their details

There are three main actors in this system which will play an important role in Feticide monitoring using android phone.

3.3.1 Use Case View

#### 

Figure 3.1: Use Case

3.3.2 Use Case Description

The main purpose of the use case diagram is to make the user understand the basic functionality of particular project. The use case describes how the user uses the application to monitor the server.

#### Preconditions

1. The user should have android compatible Smartphone.
2. The application should be install by user.

|  |  |  |
| --- | --- | --- |
| Use Case |  | Description |
| Guide |  | The Guide has various permissions and Data privacy operations. |
| Permission |  | The permission can check the permissions used for the application. |
| Data Privacy |  | The Data privacy is used for the putting up privacy for data. |
| Permission Manager |  | The Permission Manager checks for 2 methods as App by permission and permission by app. |
| App by permission |  | The App by permission checks for the only required permissions for this app. |
| Permission by app |  | The Permission by app checks for the permissions of the application. |
| App classifier |  | The App classifier has 3 stages as they are Dangerous for system, can cause money and NO harm. |
| Dangerous for system |  | The Dangerous for system can cause harm to the device. |
| Can cause Money |  | The Can cause money can minus the balance from your particular accout. |
| No harm |  | The No harm app is the safe app which is not dangerous for system or cannot cause money. |

#### 3.3.3 Data Flow Diagram:

#### 

Figure 3.2: 0 Level DFD

3.3.4 State Diagram

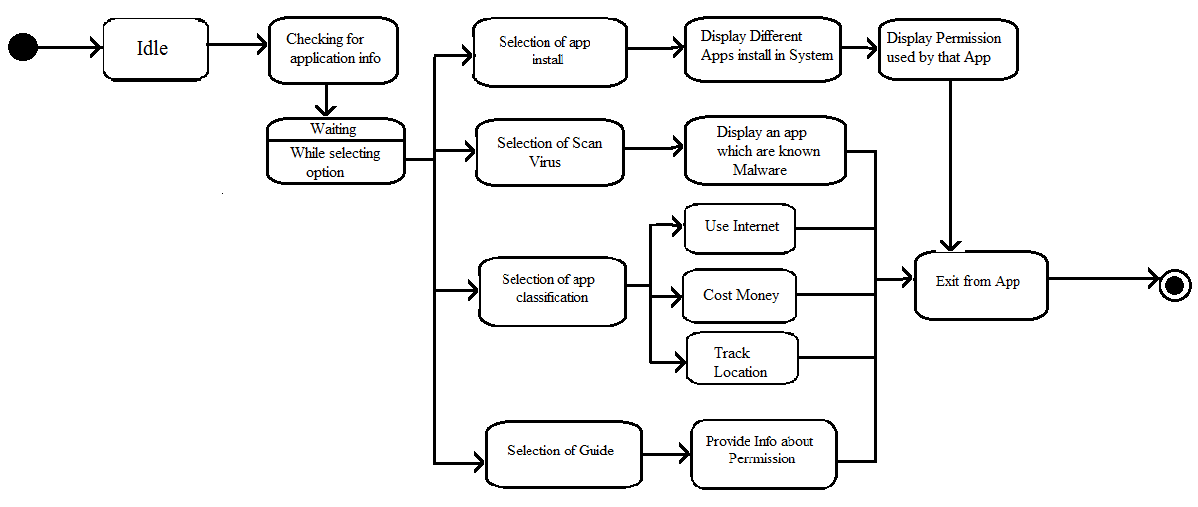


Figure: State Diagram

1. State diagram for designing the dynamic aspect of software system.
2. The state diagram depicts implementation of each specific element involved in the software system.
3. The state diagram comprises of three fundamental element: Event, State, Transition.

3.4 Activity Diagram

1. Activity diagram describes the flow of control in a system. So it consists of activities and links.

The flow can be sequential, concurrent or branched.

1. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.
2. Activity diagrams are used to visualize the flow of controls in a system.

This is prepared to have an idea of how the system will work when executed.

#### 

Figure 3.3: Activity Diagram

|  |  |  |
| --- | --- | --- |
| Use Case |  | Description |
| Application Interaction |  | The Application interaction has 3 steps as Guide, Permission manager, App classifier. |
| Guide |  | The Guide informs user about various permissions and Data privacy operations. |
| Data Privacy |  | The Data privacy is used for the putting up privacy for data. |
| Permission Manager |  | The Permission Manager checks for 2 methods as App by permission and permission by app. |
| App by permission |  | The App by permission lists for the applications which use that particular permission. |
| Permission by app |  | The Permission by app checks for the permissions of the application. |
| App classifier |  | The App classifier has 3 stages as they are Dangerous for system, can cause money and NO harm. |
| Dangerous for system |  | The Dangerous for system can cause harm to the device. |
| Can cause Money |  | The Can cause money can minus the balance from your particular accout. |
| No harm |  | The No harm app is the safe app which is not dangerous for system or cannot cause money. |

3.5 Package Diagram

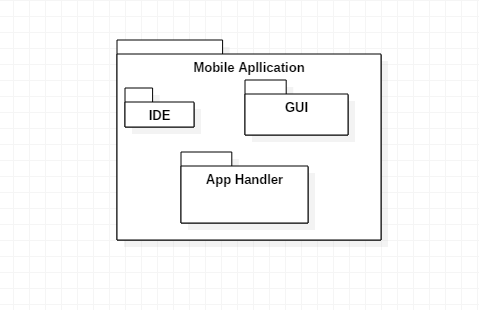


Figure 3.4: Package Diagram

1. A package diagram in the Unified Modeling Language depicts the dependencies between the packages that make up a model.
2. In addition to the standard UML Dependency relationship, there are two special types of dependencies defined between packages:
   1. package import
   2. package merge
3. A package import is by Joshua Estefen M.Torralba in the year 1946 a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace.” By default, an unlabeled dependency between two packages is interpreted as a package import relationship. In this relationship, elements within the target package will be imported in source package.
4. A package merge is a directed relationship between two packages that indicates that the contents of the two packages are to be combined. It is very similar to Generalization in the sense that the source element conceptually adds the characteristics of the target element to its own characteristics resulting in an element that combines the characteristics of both. In this relationship, if an element exists within both the source package and the target package then the source element’s definition will be expanded to include the target element
   1. Sequence Diagram
5. A sequence diagram is an interaction diagram. From the name it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another.
6. Interaction among the components of a system is very important from implementation and execution perspective.

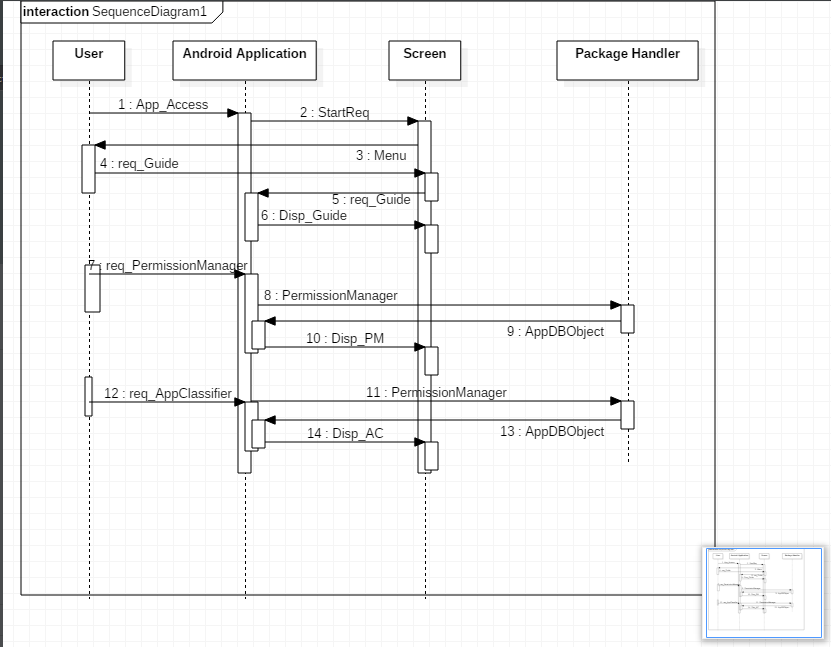


Figure 3.5: Sequence Diagram

3.7 Software and Hardware Requirements

3.7.1 Software Resources Required

* Tools : Eclipse, Android SDK, Android ADT
* Back-end : SQLite
* Languages : Java, Android Programming

3.7.2 Hardware Resources Required

* Mobile device: Android
* Memory for database
* Minimum-Android 2.1
* RAM – 320 MB
* 1GHz Single/Dual Core processor
* 256 MB ROM
* 1 GB Internal Memory

Chapter 4

Project Plan

4.1 Risk Planning

A risk management strategy should be in place to address any for easyable barriers to the implementation of an Privacy Preservation in android smartphones of this strategy should include plans to:

* Ensure adequate funding is available to provide the source applications, hardware infrastructure and implementation resources with a funding/financial management plan in place to incorporate ongoing needs
* Ensure sufficient skilled resources, both human and technical, are available to provide program management during implementation and to give on-going support
* Develop and implement a marketing strategy to promote the benefits of the proposed Online Payment

System to consumers and healthcare providers such as brochures or a newsletter to keep personnel up-to-date with developments.

* + 1. Technical Risk
  1. It identiﬁes potential design, implementation interface, veriﬁcation and maintenance problem.
  2. In our project technical risk may include cross platform issue.

As we are using Android development. Its base is java programming so risk can be minimized.

Chapter 5

Design

5.1 Introduction

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviour of the system.

5.2 System Architecture

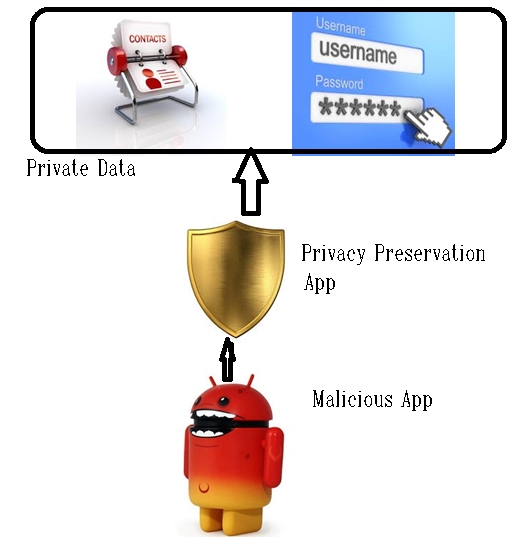
****

Figure 5.1: System Architecture

5.3 Class Diagram

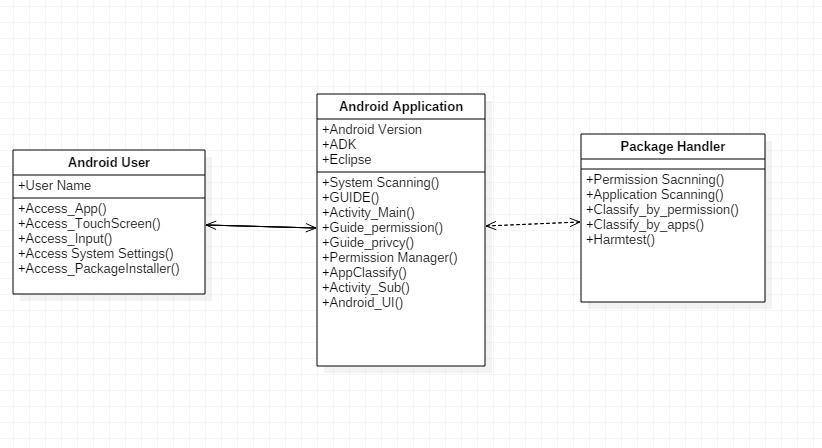


Figure 5.2: Class diagram

|  |  |  |
| --- | --- | --- |
| Class |  | Description |
| Android\_user |  | In android\_user class block there must be includes/describes user attributes as well as performs various activity as access\_app(), Touchscreen(), input(), System\_settings(), Package\_installer(). |
| Android\_Application |  | In Android\_application class block there must be includes/describes android\_application attributes as well as performs various activity. |
| Package\_handler |  | In package\_handler class block there must be includes/describes package\_handler attributes as well as performs package\_handler related activity. |

Chapter 6

6. SOFTWARE TESTING

6.1 Introduction

Testing a project or any software system forms the backbone of a good software system. Tests are conducted on the software to find out errors and bugs to remove them. Test plans are created with a view to remove errors that can plague or hamper the project in case the errors are encountered in runtime environment. A good test plan is created with a view to try to disintegrate a system to find out yet undetected errors and to correct them. Testing of software is done at each module and also at the macro level, that is, on integration of entire package as a whole. A good test plan is the one which can detect maximum number of errors or bugs in the software system. Testing helps in enhancing the overall quality of product as it helps in the removal of errors which in turn increases its quality. There are three reasons to test a program:

1. Correctness
2. For Implementation Efficiency.
3. For Computational Complexity

**Unit Testing:** Unit testing focuses verification effort on the smallest unit of software design-the software control paths are tested to uncover errors within the boundary of module. The relative complexity of test and uncover errors is limited by the constrained scope established for the unit testing. The unit testing is white- box oriented, and the step can be conducted in parallel for multiple components. White Box Testing is performed to reveal problems with the internal structure of the program. This requires the tester to have detailed knowledge of the internal structure. A common goal of white box testing is to ensure a test case exercise every path through a program. A fundamental strength that all white box testing strategies share is that the entire software implementation is taken into account during testing, which facilitates error detection even when the software specification is vague or incomplete the effectiveness or thoroughness of white box testing is commonly expressed in terms of test or code coverage matrices, which measure the fraction of code exercised by test cases.

**Validation testing:** Validation testing begins at the culmination of integration testing, when individual components have been exercised, software is completely assembled as a package, and interfacing errors have been uncovered and corrected. Here, testing focuses on user-visible actions and user recognizable output from the system. Validation succeeds when the software functions in a manner that can be reasonably expected by the customer. In our project, all functions and performance characteristics are tested and they conform to the required specifications and are accepted.

**System Testing:** This is the final step in testing. In this phase, we tested the entire system as a whole with all forms, code, modules and class modules. This form of testing is known as Black Box testing or System testing. Black Box testing enables us to derive sets of input conditions that will fully exercise all functional requirements for a program. Black box testing helps to discover incorrect or missing functions, interface errors, errors in data structure, performance errors and initialization and termination errors.

**Integration Testing**: The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. Top-Down approach: A top-down approach is essentially the breaking down of a system to gain insight into its compositional sub-systems. In a top-down approach an overview of the system is formulated, specifying but not detailing any first level subsystems. Bottom-up approach: A bottom-up approach is the piecing together of systems to give rise to grander systems, thus making the original systems sub-systems of the emergent system. In a bottom-up approach the individual base elements of the system are first specified in great detail.

**Regression Testing**: Regression testing is any type of software testing that seeks to uncover new errors, or regressions, in existing functionality after changes have been made to the software, such as functional enhancements, patches or configuration changes. The intent of regression testing is to assure that a change, such as bugs, did not introduce new bugs.

**Top-down or Bottom-up Testing:** In top-down the control program is tested first, then modules are integrated one at a time and major emphasis is on interfacing testing.

Bottom-up allows early testing aimed at proving feasibility and practically of particularly module can be integrated in various cluster as desired and major emphasis is on module; modules can be integrated in various cluster as desired and major emphasis is on module functionality and performance.

**Graphical User Interfaces (GUI) testing:** Graphical User Interfaces (GUI) testing provides the testing guidelines for specialized environments, architectures and application that are commonly encountered by software engineers. Because of reusable components, provided as a part of GUI development environments, the creation of the user interfaces has become less time consuming and more precise.

But the complexity of GUIs has also grown, leading to more difficulty in the design and execution of test cases. As modern GUIs have the same look and feel, a series of standard tests can be derived.

6.2 Goals and Objective of Testing

As said earlier, a good test plan is made with a view to the maximum number of errors. The goal and objectives to be taken in consideration while testing any software system are as follows:

* Extensive Testing must be done with a view to uncover maximum number of error.
* Try to break down the system by giving it a variety of values with a view to find hidden bugs.
* Testing every module in the project at the internal level as well as the boundary level.
* Using various testing tools and strategies like black box and white box testing.
* Removing errors and testing again for unforeseen changes in the entire software product.

6.3 Scope of Testing

It involves a very broad scope and has many advantages. Proper testing can help previously uncovered errors which may have been harmful to the proper working of the system. Thus, good testing strategy would be to have number of test plans of each individual module and then rigorously test the entire software product as a code.

6.4 Black Box Testing

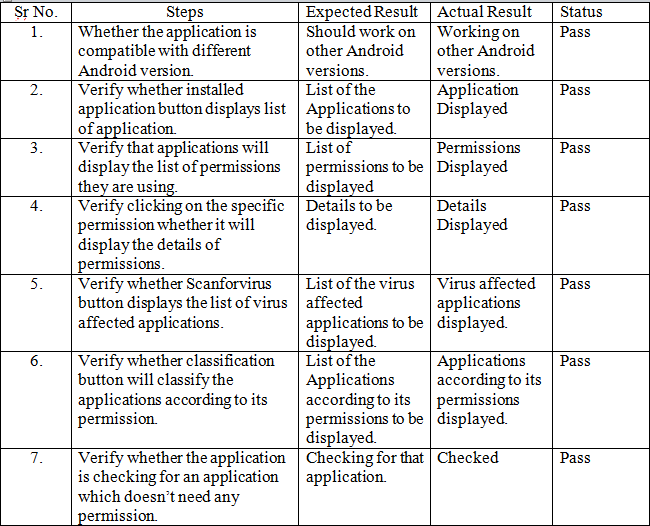
A black box test of integration builds includes functional, interface, error recovery, and stress and out of bound input testing. All black box software tests are traced to control requirements. In addition to static requirements, a black box of a fully integrated system against scenario sequence of events is designed to model field operation. Performance testing for a system is integrated as an integral part of black box test process.

6.5 White Box Testing

A level of white box test coverage is specified that is appropriate for the software being tested. The white box and other testing uses automated tools to instrument the software to measure test coverage. Using white box testing methods, the software engineer can drive test cases that

* Guarantees that all independent paths in the module have been exercised least once.
* Exercise all logical decisions on their true or false sides.
* Execute all loops at their boundaries and within their operational bounds.
* Exercise the internal data structure to ensure their validity.

6.6 Test Cases and Test Plan



Chapter 7

7. Result and Analysis

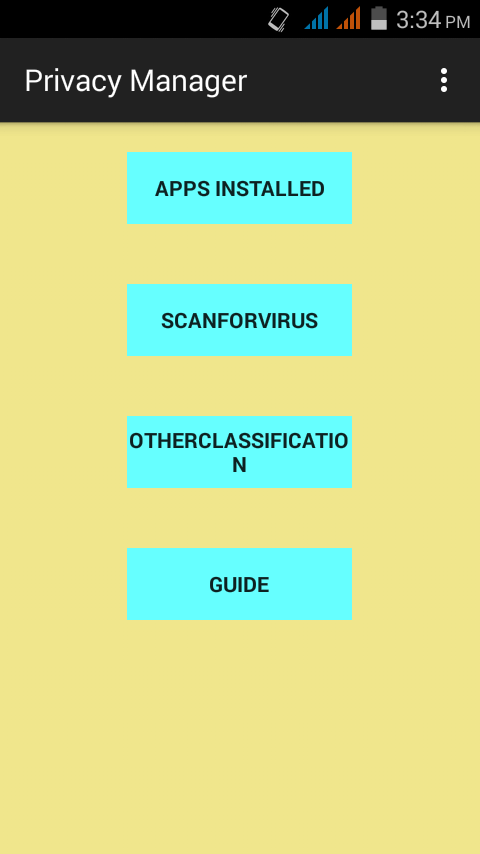


Figure: 7.1 Application Graphic User Interface

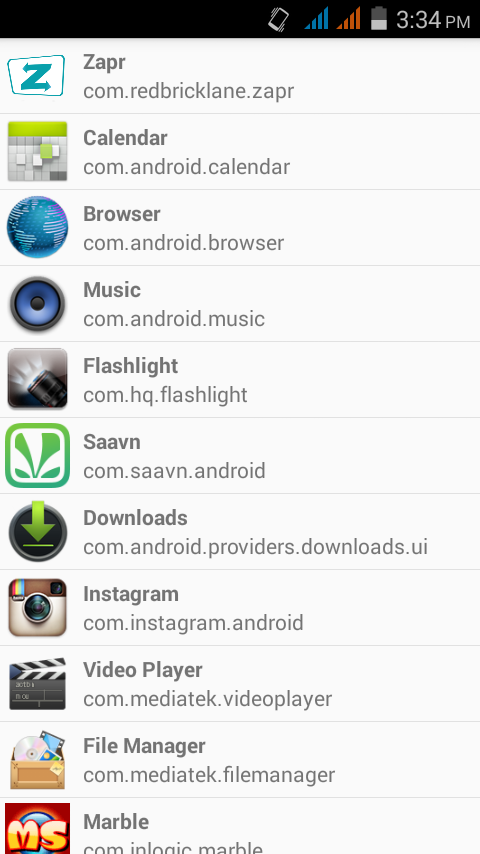


Figure 7.2: Apps Installed

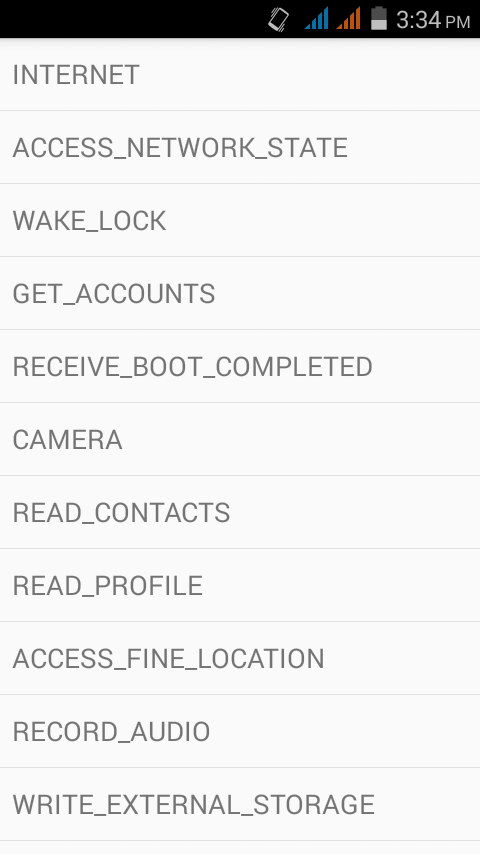


Figure 7.3: Permissions for Application

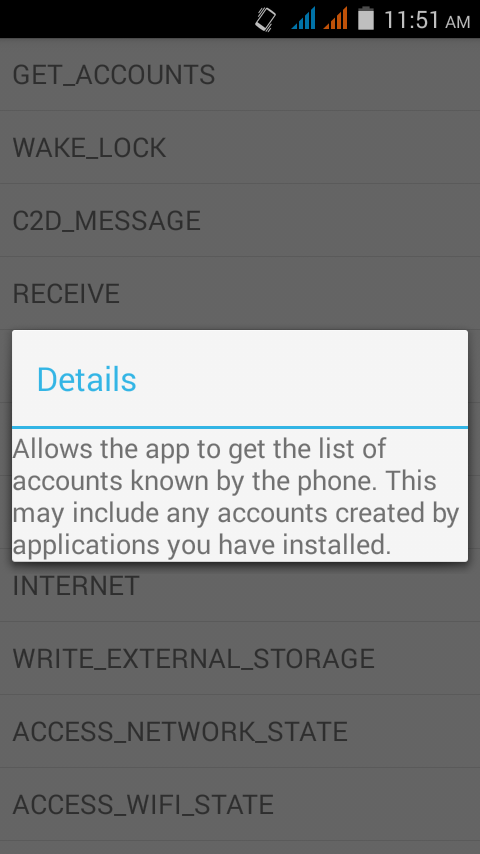


Figure 7.4: Permission Details

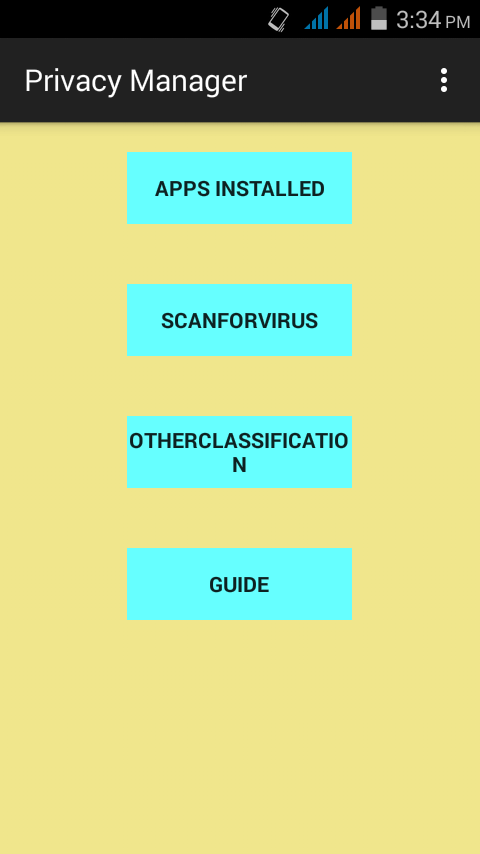


Figure: 7.5 Application Graphic User Interface

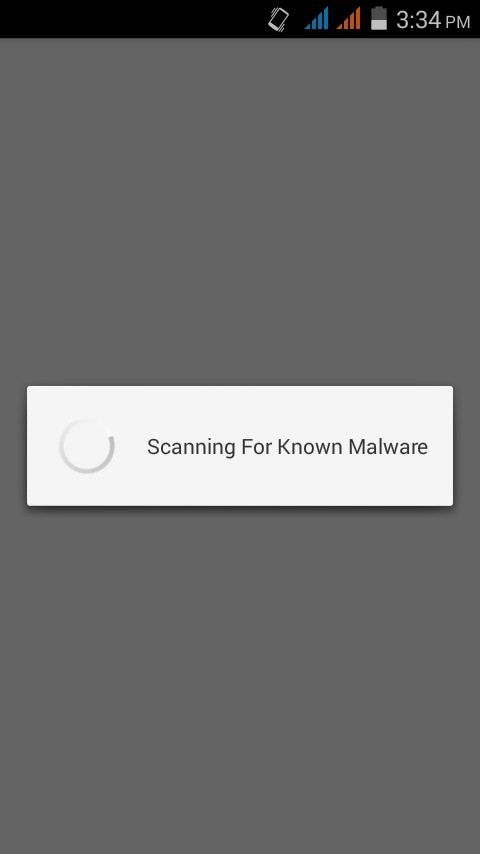


Figure 7.6: Scan For Virus

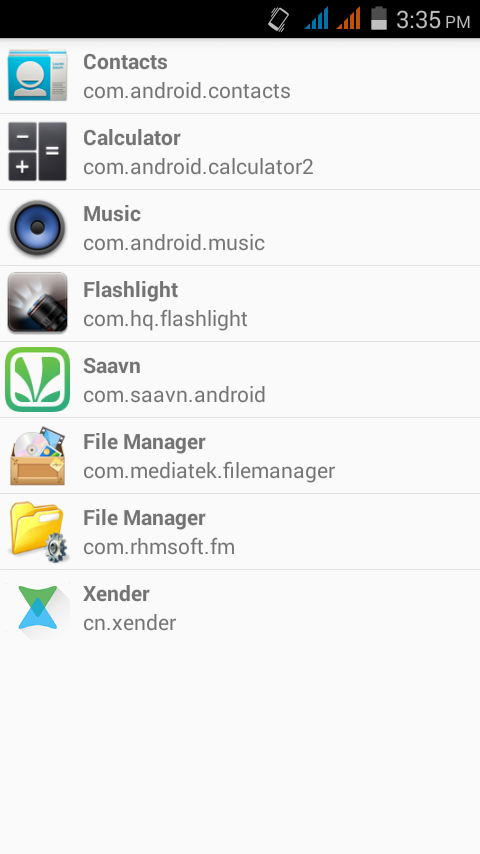


Figure 7.7: Malware Virus Applications

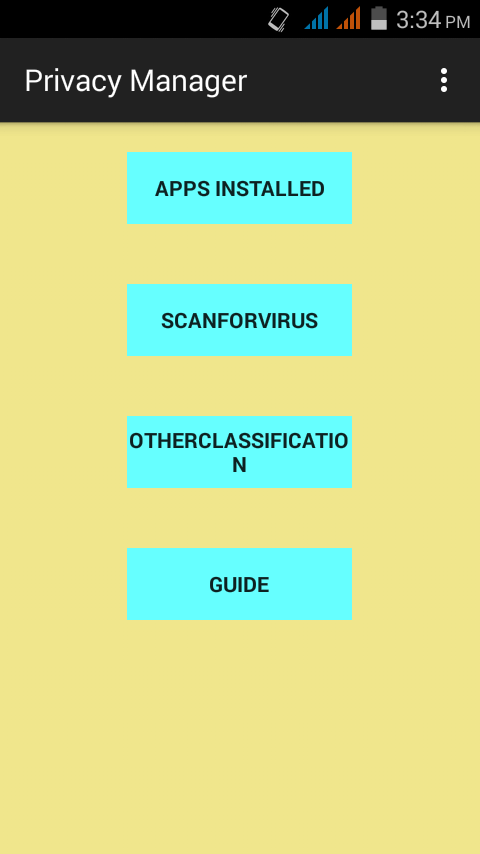


Figure: 7.8 Application Graphic User Interface

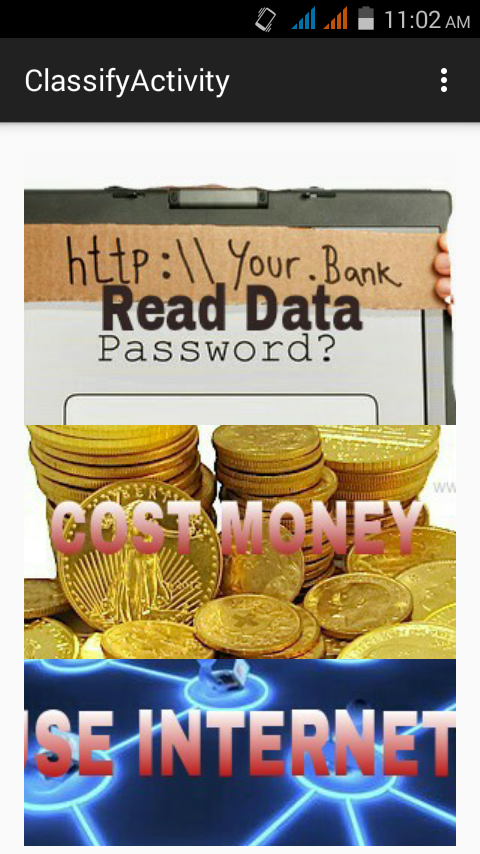


Figure 7.9: Other Classification

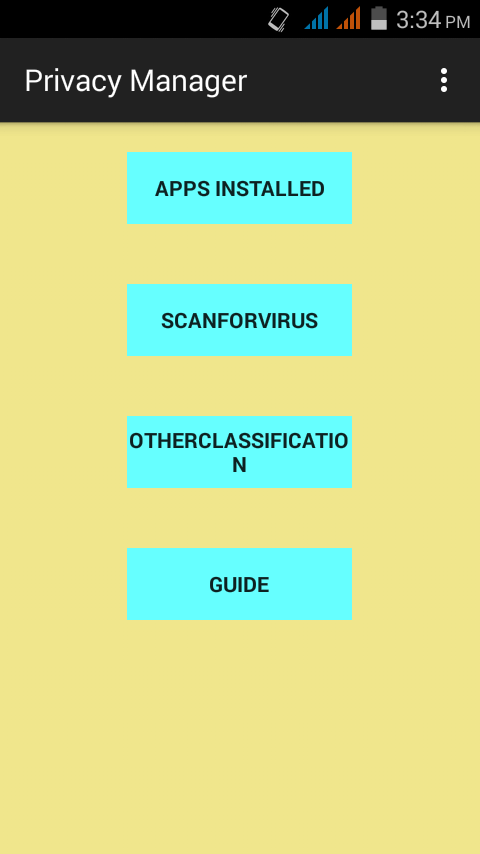


Figure: 7.10 Application Graphic User Interface



Figure 7.11: Guide In English

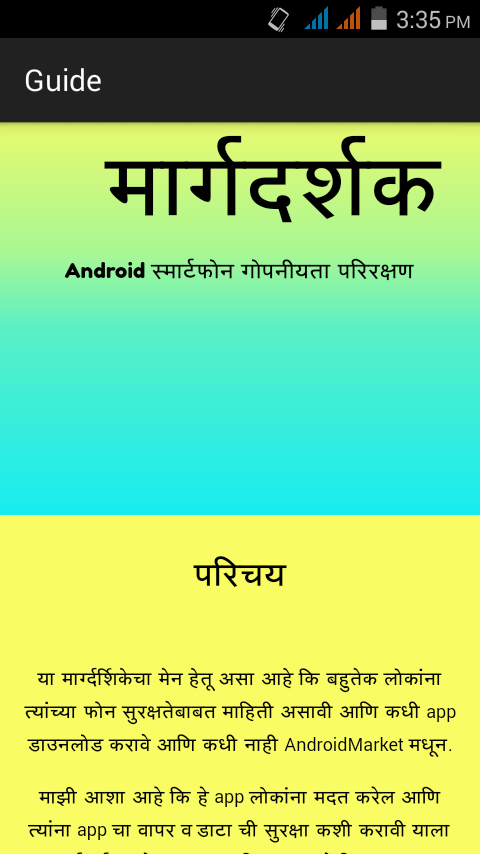


Figure 7.12:Marathi Guide

Chapter 8

8. Conclusion

#### We studied architecture of android operating system. Security flaws can be assessed by using some techniques. Apps can use permissions to access user’s private data. Once permission is granted app can use those permission unanimously. This is the biggest drawback of android operating system. So we tried to develop a solution for this problem. A Monitoring app can be developed to monitor activities of apps. That app can assist user to keep data safe and private.

Chapter 9

9. REFRENCES

1. Android Developers. Renderscript. <http://developer.android.com/reference/android/renderscript/package-summary.html>.
2. Donald Beaver. Commodity-based Cryptography.

In ACM Symposium on Theory of Computing, 1997.

1. Emiliano De Cristofaro, Mark Manulis, and Bertram Poettering. Private Discovery of Common Social Contacts. In Applied Cryptography and Network Security, 2011.
2. Wenliang Du and Zhijun Zhan. A Practical Approach to Solve Secure Multi-party Computation Problems. In New Security Paradigms Workshop, 2002.
3. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient Private Matching and Set Intersection. In EUROCRYPT, 2004.
4. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play Any Mental Game. In ACM Symposium on Theory of Computing, 1987.
5. Dan Gusfield and Robert W. Irving. The Stable Marriage Problem: Structure and Algorithms. MIT Press, Cambridge, MA, USA, 1989. ISBN 0-262-07118-5.
6. Carmit Hazay and Yehuda Lindell. Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries. In Theory of Cryptography Conference, 2008.
7. Yan Huang. Fast Secure Computation Framework. http://www.MightBeEvil.org/framework/, 2011.
8. Yan Huang, Peter Chapman, Sang Koo, and David Evans. Common Contacts Android App. http://mightbeevil.org/contacts/, 2011.
9. Yan Huang, David Evans, and Jonathan Katz. Private Set Intersection: Are Garbled Circuits Better than Custom Protocols? <http://www.cs.virginia.edu/yhuang/pubs/psi.pdf>, 2011.
10. Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster Secure Two-Party Computation Using Garbled Circuits. In USENIX Security Symposium, 2011.
11. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending Oblivious Transfers Efficiently. In CRYPTO, 2003.
12. KnowYourGenes.org. Genetic Testing Information from the Genetic Disease Foundation. <http://www.knowyourgenes.org/>.
13. Vladimir Kolesnikov and Thomas Schneider. Improved Garbled Circuit: Free XOR Gates and Applications. In International Colloquium on Automata, Languages and Programming, 2008.
14. Yehuda Lindell and Benny Pinkas. A Proof of Security of Yao’s Protocol for Two-Party Computation. Journal of Cryptology, 2009.
15. Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay—A Secure Two-party Computation System. In USENIX Security Symposium, 2004.
16. Moni Naor and Benny Pinkas. Efficient Oblivious Transfer Protocols. In ACM-SIAM Symposium on Discrete Algorithms, 2001.
17. Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure Two-Party Computation is Practical. In ASIACRYPT, 2009.
18. Arvind Seshadri, Adrian Perrig, Leendert van Doorn, and Pradeep K. Khosla. SWATT: SoftWare-based ATTestation for Embedded Devices. In IEEE Symposium on Security and Privacy, 2004.
19. Andrew C. Yao. How to Generate and Exchange Secrets. In Symposium on Foundations of Computer Science, 1986.