

Android Antivirus



By

Capt Awais bin Riaz

Maj Najam ul Hassan

GC Danyal Mukhtar

Submitted to Faculty of Department of Computer Software Engineering National University of Sciences and Technology, Islamabad in partial fulfillment for the requirements of a B.E Degree in Computer Software Engineering, June 2018

In the name of Allah, the Most Beneficent, the Most Merciful

ABSTRACT

Development in the technology results highly in the development of malware and malicious activities to damage and destroy other systems. Android is one of the main platform where hackers and attackers put up their efforts because number of android users are increasing day by day. In the light of this we worked on an android antivirus. Antivirus are of two types , one signature base and other behavioral base antivirus. It is a behavioral base antivirus that categorizes the applications as high, medium and low risk and gives an option to the user to uninstall or trust the applications. We first gathered 1000 malicious and 500 benign applications which are tested against a set of rules to know about their behaviour. On the basis of the results, we refine the rules and finalize them. After these technical tasks we made an android application on the basis of these refined and finalized rules and then tested those against a set of 500 malicious and 500 benign Android applications. Then we calculated accuracy and efficiency of the designed system. In this thesis we explained all the steps and technicalities of our system with detailed explanation.

CERTIFICATE FOR CORRECTNESS AND APPROVAL

It is certified that the work kept in the thesis – android antivirus carried out by Capt Awais bin Riaz, Maj Najam ul Hassan and GC Danyal Mukhtar under supervision of A/PWaleed bin Shahid for fractional fulfillment of Degree of Bachelor of Software Engineering is correct and approved.

Approved By

A/P Waleed bin Shahid

Department of IS, MCS

Dated:

DECLARATION

The work for this project is not done for any group or individual award or for any qualification in the institute or outside the institute.

DEDICATION

To our parents, without their support and cooperation it would not be possible for us to work in that manner. To teachers who always stood with us and helped us with their valuable suggestions and great motivation.

ACKNOWLEDGEMENTS

We are very grateful to ALLAH Almighty who gave us strength and guidance to work on this project. Without the will of ALLAH Almighty nothing is possible. We would like to thank our parents and all the well-wishers for their prayers and moral support. We would like to thank our teachers for their efforts and guidance enabling us to complete this project.

Table of Contents

Table of Contents	viii
List of Figures.....	ix
Chapter 1. Introduction	1
1.1. Overview	1
1.2. Problem Statement.....	6
1.3. Approach	6
1.4. Scope.....	6
1.5. Objectives.....	7
1.6. Deliverables	8
Chapter 2. Literature Review.....	9
Chapter 3. Software Req. Specification (SRS).....	11
3.1. Introduction.....	11
3.2. Overall Description	13
Specific Requirements	15
Other Nonfunctional Requirements.....	17
3.3. External Interface Requirements	18
Chapter 4. Design and Development	23
1. 23	
2. 23	
3. 23	
4. 23	
4.1. Introduction.....	23
2. System Architecture Description.....	25
3. User Interface	36
Chapter 5. Project Test and Evaluation	52
5.1. Introduction.....	52
5.2. Test Items.....	53
5.3. Features to Be Tested.....	53
5.4. Test Approach	53
5.5. Item Pass/Fail Criteria	54
5.6. Suspension Standards and Resumption Requirements.....	54
5.7. Test Deliverables	54
5.8. Responsibilities, Staffing and Training Needs	61
5.9. Risk and Contingencies	61
Bibliography	63
Appendix A. Glossary	68
Appendix B: Issues/Limitations.....	69

List of Figures

Figure 3.4.1 Login Screen.....	20
Figure 3.4.2 Student Screen	21
Figure 3.4.3 Teacher Screen	22
Figure 3.4.4 Admin Screen.....	23
Figure 4.2.1 System Block Diagram	Error! Bookmark not defined.
Figure 4.2.2 System Use case Diagram	Error! Bookmark not defined.
Figure 4.2.3 Use Case - Send Message	Error! Bookmark not defined.
Figure 4.2.4 Use Case - Send File.....	Error! Bookmark not defined.
Figure 4.2.5 Use Case - Receive Message	Error! Bookmark not defined.
Figure 4.2.6 Use Case - Receive File.....	Error! Bookmark not defined.
Figure 4.2.7 Use Case - Access Log	Error! Bookmark not defined.
Figure 4.2.8 Sequence - Send Message.....	Error! Bookmark not defined.
Figure 4.2.9 Sequence - Send File	Error! Bookmark not defined.
Figure 4.2.10 Sequence - Receive Message.....	Error! Bookmark not defined.
Figure 4.2.11 Sequence - Receive File	Error! Bookmark not defined.
Figure 4.2.12 Sequence - Access Log.....	Error! Bookmark not defined.
Figure 4.2.13 System Class Diagram.....	Error! Bookmark not defined.
Figure 4.2.14 Activity Diagram	Error! Bookmark not defined.
Figure 5.1 Architectural Diagram	Error! Bookmark not defined.

Chapter 1. Introduction

1.1. Overview

Development in the technology results highly in the development of malware and malicious activities to damage and destroy other systems. Android is one of the main platforms where hackers and attackers put up their efforts because no of android users are increasing day by day. So keeping in mind these all things it is necessary for us to overcome these problems, no of attempts are being made for this purpose, no of antivirus came in market and are coming. Similarly we made an attempt for such an issue and make a behavioral base android antivirus which categorize applications as high, medium and low risk and allow user to keep or uninstall the risky applications.

STEPS FOR ANTIVIRUS COMPLITION:

We will do it in following way.

1. COLLECTION OF BENIGN AND MALICIOUS APPLICATIONS:

To access a combination of benign and malicious apps, we chose third party app stores that have a bulk of malicious applications uploaded in disguise of the original ones and some available datasets of malicious applications on Virus Share and AMG. These third party app stores offer installation of apps except the authorized channels of installation, such as Google Play Store in our case. We used datasets from Virus Share, AMG and app stores like amazon appstore, getjar , f-droid etc to download a total of almost 1,500 applications.

Out of these, approximately 1,000 were found to be malicious and 500 were found to be benign. The apps that were considered malicious mainly met one of the following conditions:

- **Spyware:** These apps track user's activities like SMS, Calls etc. and keep sending them to server without the knowledge and consent of user.
- **Trojans:** These apps offered a covert purpose hidden behind an overt purpose.
- **Phishing Sites:** These apps imitate another legitimate app to deceive user into logging in with their personal credentials at their platform in order to steal this sensitive information.
- **Hidden Processes:** These are apps that run in the background, hiding themselves and on the lookout for certain events to happen.

To detect if an app is benign or malicious, we used a website named Virustotal.com. It analyzes suspicious files and URLs to detect types of malware, and automatically shares them with the security community. We uploaded .apk files of the applications that we have downloaded from our third party app store, and www.virustotal.com classified them on the basis of their malicious intents. This allows us to reach the first step of our project, i.e. to have a vast collection of applications that have been classified as malicious or benign, so we can use them for development and testing of our project. By matching the results with those that www.virustotal.com provided, we will be able to see if our app is working perfectly to detect malicious apps.

2. TESTING OF APPLICATIONS AGAINST SET OF RULES:

Now we will do a windows machine test of these applications against a given set of rules to gain a separate result for benign and malicious applications. Following steps will be done for this purpose

- First we will get the apks using command “command dir into a text file”.
- Now read that text file in java.
- Now store all apks in ArrayList <String>
- Now use android sdk tool aapt, use command “ command aapt-d permissions , to extract permissions from manifest file.
- Now store these permissions in a separate ArrayList.

(Same process will carry on for all the apks store in previous ArrayList)

By doing this we will have all the permissions ready in permission ArrayList

- Now one by one we will compare each apks permission to all the rules separately, while keep arranging it in a text file called comparison.
The result of these applications against set of rules comes as follow

Rule No	Benign out of 540	Malicious out of	Benign %	Malicious %
Rule 1a	42	12	7.777777778	1.079136691
Rule 1b	62	69	11.48148148	6.205035971
Rule 1c	62	69	11.48148148	6.205035971
Rule 1d	201	917	37.22222222	82.46402878
Rule 1e	203	918	37.59259259	82.55395683
Rule 2a	1	0	0.185185185	0
Rule 2b	6	36	1.111111111	3.237410072

Android Antivirus

Rule 2c	6	36	1.111111111	3.237410072
Rule 2d	15	653	2.777777778	58.72302158
Rule 2e	15	657	2.777777778	59.08273381
Rule 3a	4	0	0.740740741	0
Rule 3b	20	6	3.703703704	0.539568345
Rule 3c	20	6	3.703703704	0.539568345
Rule 3d	42	144	7.777777778	12.94964029
Rule 3e	43	144	7.962962963	12.94964029
Rule 4a	195	341	36.111111111	30.66546763
Rule 4b	196	434	36.2962963	39.02877698
Rule 5a	3	0	0.555555556	0
Rule 5b	11	35	2.037037037	3.147482014
Rule 5c	11	35	2.037037037	3.147482014
Rule 5d	24	597	4.444444444	53.68705036
Rule 5e	24	597	4.444444444	53.68705036
Rule 6a	0	0	0	0
Rule 6b	0	0	0	0
Rule 6c	2	179	0.37037037	16.0971223
Rule 7	0	0	0	0
Rule 8	131	65	24.25925926	5.845323741
Rule 9a	0	0	0	0
Rule 9b	0	0	0	0
Rule 10a	0	0	0	0
Rule 10b	83	5	15.37037037	0.449640288
Rule 11	36	38	6.666666667	3.417266187
Rule 12a	46	97	8.518518519	8.723021583
Rule 12b	175	233	32.40740741	20.95323741
Rule 13a	43	81	7.962962963	7.284172662
Rule 13b	152	217	28.14814815	19.51438849
Rule 14	5	77	0.925925926	6.924460432
Rule 15	6	80	1.111111111	7.194244604
Rule 16	2	0	0.37037037	0
Rule 17a	15	28	2.777777778	2.517985612
Rule 17b	69	116	12.77777778	10.43165468
Rule 17c	69	116	12.77777778	10.43165468
Rule 17d	69	116	12.77777778	10.43165468
Rule 17e	69	116	12.77777778	10.43165468
Rule 18a	53	8	9.814814815	0.71942446
Rule 18b	55	10	10.18518519	0.899280576
Rule 19a	12	5	2.222222222	0.449640288
Rule 19b	36	28	6.666666667	2.517985612
Rule 20	0	2	0	0.179856115
Rule 21a	1	0	0.185185185	0

Android Antivirus

Rule 21b	1	0	0.185185185	0
Rule 22	334	573	61.85185185	51.52877698
Rule 23	75	114	13.88888889	10.25179856
Rule 24	277	102	51.2962963	9.172661871
Rule 25	126	30	23.33333333	2.697841727
Rule 26	2	4	0.37037037	0.35971223
Rule 27	0	0	0	0
Rule 28a	13	63	2.407407407	5.665467626
Rule 28b	13	63	2.407407407	5.665467626
Rule 29a	3	0	0.555555556	0
Rule 29b	3	0	0.555555556	0
Rule 30a	0	0	0	0
Rule 30b	0	0	0	0
Rule 31	2	16	0.37037037	1.438848921
Rule 32	38	83	7.037037037	7.464028777
Rule 33	5	3	0.925925926	0.269784173

3. REFINEMENT:

In previous step we tested the applications against a given set of rules, but in these rules some rules show repetition in applications so we have refined those rules. First we had 33 rules which includes their sub rules which make it 64 rules, so after doing refinement we have decreased these rules to 22, which will be again tested against some applications in testing section. The result of refinement comes as

Rule No	Benign out of 540	Malicious out of 1112	Benign %	Malicious %
Rule 1d	201	917	37.22222222	82.464028
Rule 1e	203	918	37.59259259	82.553956
Rule 2b	6	36	1.111111111	3.2374100
Rule 2c	6	36	1.111111111	3.2374100
Rule 2d	15	653	2.777777778	58.723021
Rule 2e	15	657	2.777777778	59.082733
Rule 3d	42	144	7.777777778	12.949640

Android Antivirus

Rule 3e	43	144	7.962962963	12.949640
Rule 4b	196	434	36.2962963	39.028776
Rule 5b	11	35	2.037037037	3.1474820
Rule 5c	11	35	2.037037037	3.1474820
Rule 5d	24	597	4.444444444	53.687050
Rule 5e	24	597	4.444444444	53.687050
Rule 6c	2	179	0.37037037	16.09712
Rule 12a	46	97	8.518518519	8.7230215
Rule 14	5	77	0.925925926	6.9244604
Rule 15	6	80	1.111111111	7.1942446
Rule 20	0	2	0	0.1798561
Rule 28a	13	63	2.407407407	5.6654676
Rule 28b	13	63	2.407407407	5.6654676
Rule 31	2	16	0.37037037	1.4388489
Rule 32	38	83	7.037037037	7.4640287

4. CREATION OFF APPLICATION :

We created a simple interface of DNA antivirus application in android studio, we make this application as simple that anyone having some knowledge of android and it's applications can use it easily.

5. TESTING :

Previously we refined some rules and after refinement we get 22 rules which we tested them against more 500 benign and 500 malicious applications which are different from previous applications The result of these application was

Rule No	Benign out of 500	Malicious out of 500	Benign %	Malicious %
Rule 1d	127	125	25.4	25
Rule 1e	130	125	26	25
Rule 2b	14	62	2.8	12.4
Rule 2c	14	62	2.8	12.4
Rule 2d	14	62	2.8	12.4
Rule 2e	15	62	3	12.4
Rule 3d	51	45	10.2	9
Rule 3e	54	45	10.8	9
Rule 4b	132	352	26.4	70.4
Rule 5b	13	10	2.6	2

Android Antivirus

Rule 5c	13	10	2.6	2
Rule 5d	13	10	2.6	2
Rule 5e	14	10	2.8	2
Rule 6c	0	1	0	0.2
Rule 12a	54	9	10.8	1.8
Rule 14	2	2	0.4	0.4
Rule 15	2	2	0.4	0.4
Rule 20	0	0	0	0
Rule 28a	15	70	3	14
Rule 28b	15	70	3	14
Rule 31	0	0	0	0
Rule 32	49	50	9.8	10

1.2. Problem Statement

As android is getting popular day by day and the open source nature of android, the attackers try to and make their priority to target all the malicious applications over other mobile operating systems. There also present some effective antivirus software firms and products but somehow the approach they use is not totally appropriate for mobile phones.

1.3. Approach

We tried to use less information and computer computational properties as possible as we can. This was done to get accurate results for identifying malicious applications devoid of essentially installing these applications. In this application user's are warned and it's totally upto user to uninstall or to keep that application on his/her own risk. the manifest file of each application contains it's permission because of which we save a lot of computational resources for getting the source code. In our thesis, we took 1000 malicious and 500 benign applications through many open source websites, we get to know about their benign and malicious behavior by getting their apks and check it through site www.virustotal.com. After that, we manually analysed the source code and manifest files for permissions required for each App.

1.4. Scope

As with the increase in technology, android is becoming popular day by day and number of users of android is also increasing on daily basis. As the number of users of android is

increased, the numbers of malicious attacks are also increasing day by day. The purpose of these attacks is to steal private information and consumer credit by subscribing to some of the premium services. There are many antivirus solutions available but these antivirus mostly uses signature base algorithms which will fail if some new virus comes to market, as they will not have the signature of that malicious application, so they will not be able to find zero days attack. So in the light of this issue, in our thesis we present a behavioral base antivirus that is use for detecting both malware and zero days attack. Our antivirus will be able to detect those zero days attack that other antivirus will fail to do it. Our antivirus will outperform other such applications that are running on android.

1.5. Objectives

The main objective of this Android Antivirus is to hold the zero day attacks which can only be stopped using a behavior based Antivirus which uses a specially planned algorithm to learn the behavior of the system calls, strings, permissions and mixtures of permissions to examine the application whether is it safe to run or malicious. Generally the Antivirus which are obtainable on the internet and the normally used Antivirus which are used by everybody are signature based which practices available signatures on the internet of preprogrammed virus patches to find out the malicious application.

1.6. Deliverables

Sr	Tasks	Deliverables
1	Literature Review	Literature Survey
2	Requirements Specification	Software Requirements Specification document (SRS)
3	Detailed Design	Software Design Specification document (SDS)
4	Implementation	Project demonstration
5	Testing	Evaluation plan and test document
6	Training	Deployment plan
7	Deployment	Complete application with necessary documentation

Chapter 2. Literature Review

2.1 Behavioral – Based dynamic analysis

Active study can also be done for chasing of data flow in the mobile suited applications, mostly in cases like Android malware recognition. These emphasize on Application's runtime performance, an example to this is to get inside the private material without giving any clue to the user or maybe by trying sending test messages in it's background. On the other hand, numerous benign Applications too transfer sensitive information to accomplish their mandatory functionality, specially multiple functional or communicational and public Applications alike instagram, Facebook and wechat. For example, if we take facebook application, a user takes some snap, then he might access the GPS data, he might access the phone list after GPS and might send this information to some colleague. In the given situation, the Application might access somewhat a percentage of complex and sensitive data and spread the data through some SMS or by some means of Internet, which must attentive the dynamic examination mechanism then express it to be a malicious behavior.

2.2 Android permission model

Permission prototypes have turn out to be the main security tools meant for android based operating system to give access control to some sensitive data or hardware. According to some internet sources, Android applications have no permission to do any such operation that can affect any kind of user information. An android application usually functions in the sandbox with its distinctive system distinctiveness. So each android application is usually divided from the other application. The modern OS of mobiles like android, IOS and other, the permission of android has a good control over OS. Every application in android should have a androidmanifest.xml file in it. The basic purpose of this manifest file is to give almost every relevant statistics and figures about the application, it also includes system all permissions. Android permission system consists of four permission types. First two classifications are classified as Signature permissions and System permissions, these applications are not designed for some third party apps. The other two permissions are dangerous permissions and normal permissions which are only present so that it can be request by applications. Some experiments show that around 200 different type of permissions are currently available. Such a large set of permissions make it difficult for ordinary users to classify a app as malicious or not.

2.3 Typical Permissions

Android Antivirus

In such permissions , some permissions are dangerous as compare to others in term of security out of 200 permissions.

2.3.1 Privacy Associated Permissions

Some threats try to get users personal data that is not for them. Many of these permissions are usually read permissions. An example of it is that an application like zapyra should not have access to your messages.

- `Access_Fine_Location`: It gives access to approximately precise location.
- `Body_Sensors`: Gives access to factors related to human body like heart rate etc.
- `Call_Phone`: It gives permission to user to dial call without the permission of the user.
- `Camera`: It gives illegal access to some camera features.
- `Get_Accounts`: It gives account access illegally.
- `Read_Calendar`: It give access to data related to calendar.

2.3.2 Write or Amend Related Permissions

Some threats try to change user's information (private), among them some might mislead that information.

- `Add_Voicemail`: Gives a chance to application to add some voicemail in the system.
- `Process_Outgoing_Calls`: Give access to some outgoing call and it may abort or redirect that call..
- `Send_Sms`: Gives access to a application to send messages.
- `Write_Calendar`: It gives access so that one can write on user's own calendar.

Chapter 3. Software Req. Specification (SRS)

3.1. Introduction

Purpose

This document describes the software design and specification for Android Antivirus (DNA Antivirus). This document will also show the interface of DNA antivirus.

Document Conventions

- Heading are prioritized in a numbered fashion, the highest priority heading having a single digit and subsequent headings having more numbers, according to their level.
- Font used is Calibri (body).
- All the main headings are of size 22 and bold.
- All the second level sub-headings are of size 16 and bold.
- All the further sub-headings are of size 14 and bold.

Intended Audience and Reading Suggestions

This document is useful for all customers and developers and all the stakeholders involved. The reader of this document should have basic knowledge related to android, it's function and malware in android world.

Project Scope

To get release from malware, Antivirus is an application or software that can be used to set ourselves save from viruses. As with the increase in technology, android is becoming popular day by day and number of users of android is also increasing on daily basis. As the number of users of android is increased, the numbers of malicious attacks are also increasing day by day. The purpose of these attacks is to steal private information and consume credit by subscribing to some of the premium services. There are many antivirus solutions available but these antivirus mostly uses signature base algorithms which will fail if some new virus comes to market, as they will not have the signature of that malicious application, so they will not be able to find zero days attack. Many antiviruses have many functionalities to detect and get the malware deleted but in our project we will use limited functionalities as it is the initial version and basics of our final year project.

References

The reading material and some guides are available in the form of web links.

1. **Firestore Database REST API**
<https://firebase.google.com/docs/reference/rest/database/>
2. **Firestore Cloud Messaging HTTP Protocol**
<https://firebase.google.com/docs/cloud-messaging/http-server-ref>
3. **System Requirements Specification Template**
<http://ieeexplore.ieee.org/document/741940/?tp=&isnumber=16016&arnumber=741940&punumber=5982>

3.2. Overall Description

Product Perspective

The antivirus works independently. Our antivirus has a simple and easy interface, it has user friendly interface, that can be used by anyone having some knowledge about android phone and it's programs.

Software interface: Our antivirus has a simple and easy interface, it has user friendly interface, that can be used by anyone having some knowledge about android phone and it's programs.

Hardware interface: The software will run on an Android device.

User interfaces

Customer: The user interface of customer must be spontaneous, so that 99.9% of all new DNA antivirus users are able to scan and clean their system without any assistance.

Maintainer: The maintainer is responsible for adding new features to the software and updating existing DNA antivirus. A maintainer should be possible to update the changes in the existing DNA antivirus software.

Product Features

DNA antivirus has a simple and easy to use interface, DNA antivirus can detect malware by analyzing it's behavior. There are set of rules on the basis of which the antivirus categorizes the application risk level (we have 64 such rules, 33 rules and then further it's sub rules). So DNA antivirus strengthens your device and gives it a basic security against many malwares. The antivirus also helps the device to work properly and smoothly. The DNA antivirus give option to it's user to either keep or uninstall the threatening application. The antivirus scans your whole device and detects malware in it. As it is our basic level so for now the antivirus has limited functionalities but in near future more work can be done on it , to enhance its performance and credibility.

User Classes and Characteristics

Characteristics: There are several users of the DNA Antivirus :

Customers are simply members of the general public with no special training.

Maintainers must be experienced Ethical hackers, to be able to add new features to the software according to requirement of the security of the systems.

Operating Environment

The hardware, software and technology used should have following specifications:

- Ability to scan the application in the system.
- Ability to find risk level of an application.
- Ability to uninstall or keep that application.
- Supported by Android (All Versions).
- Ability to speedup system
- Enhanced user interface.
- Enhanced system performance.

Design and Implementation Constraints

- A Functional Android Device.

Virus Scan:

- Display a message at top of the interface, “Regular virus scan will help you remove threats promptly”.
- It categorizes the app in categories like high risk, medium risk and low risk respectively.
- Privacy can also be altered of those applications who have privacy issues.
- When the system is scanned it also gives us the list of applications who has privacy issues.
- You can either delete the malicious applications with risk or trust the applications.

Tool Box:

- Contains different tools.
- Apk studio
- Android studio
- Dex2jar
- Algorithm rules
- Eclipse
- Jd gui

2.6 Assumptions and Dependencies

- Hardware never fails
- Software does not crashes.
- Antivirus is installed in supporting operating system.

Specific Requirements

Functional Requirements

Functional requirements has two main sections, one of them work for DNA antivirus while the other one works for the system.

Requirements DNA Antivirus

DNA antivirus has categorize it's requirements in the following ways.

- General necessities
- Necessities for authorization
- Necessities for a transaction.

General

Functional requirement 1:

- **Description:** Install DNA antivirus in the system.
- **Input:** DNA antivirus is installed in the system.
- **Processing:** Installing DNA antivirus
- **Output:** DNA antivirus is ready to use.

Functional requirement 2:

Android Antivirus

- **Description:** If the operating system does not support DNA antivirus software, Then display the error message.
- **Input:** Setup is installed.
- **Processing:** The operating system does not support the setup.
- **Output:** Display an error message. Abort installation process.

Functional requirement 3:

- **Description:** The should check if the installation process is followed correctly or not.
- **Input:** Now the customers should enter the setup of software.
- **Processing:** Now check if the installation process is correct
- **Output:** Display error message if the installation process is incorrect.

Functional requirement 4:

- **Description:** If the installation process is correct, the DNA Antivirus setup should be installed properly.
- **Input:** Correct process.
- **Processing:** Installing the setup.
- **Output:** Initiate installation complete dialog

Functional requirement 5:

- **Description:** The theme is to be selected by the user.
- **Input:** Theme is selected by the user.
- **Processing:** Applying the theme.
- **Output:** Update the theme.

Functional requirement 7:

- **Description :** The user has now scanned applications into the categories such as high risk, medium risk or low risk.
- **Input:** The user now select the application at high risk.
- **Processing:** It then shows the application details as to what all permissions and what calls does it can make in your system so you delete/uninstall it.
- **Output:** The application at high risk is then deleted from your system.

Functional requirement 8:

- **Description:** When the user wants to Scan the system.
- **Input:** The user selects the Scan button when open the Antivirus.
- **Processing:** It scans files and folders for the viruses and worms.

Android Antivirus

- **Output:** Displays the number of threats found in the system and displays the name of the applications which contains the threat whether high, medium or low.

Other Nonfunctional Requirements

Performance Requirements

- High scanning rate
- Uninterrupted scanning process
- Minimum false positive result

Safety Requirements

- The use of Android Antivirus should have no harm.
- If the application crashes then it should not delete or change any data regarding applications.

Security Requirements

- Users are advised to scan their system after every few days specially when installing new application.
- Users are advised to remove the malicious applications from the system which are suggested.

Software Quality Attributes

5.4.1 Availability: The services of DNA Antivirus on your device has to be available 24 hours a day.

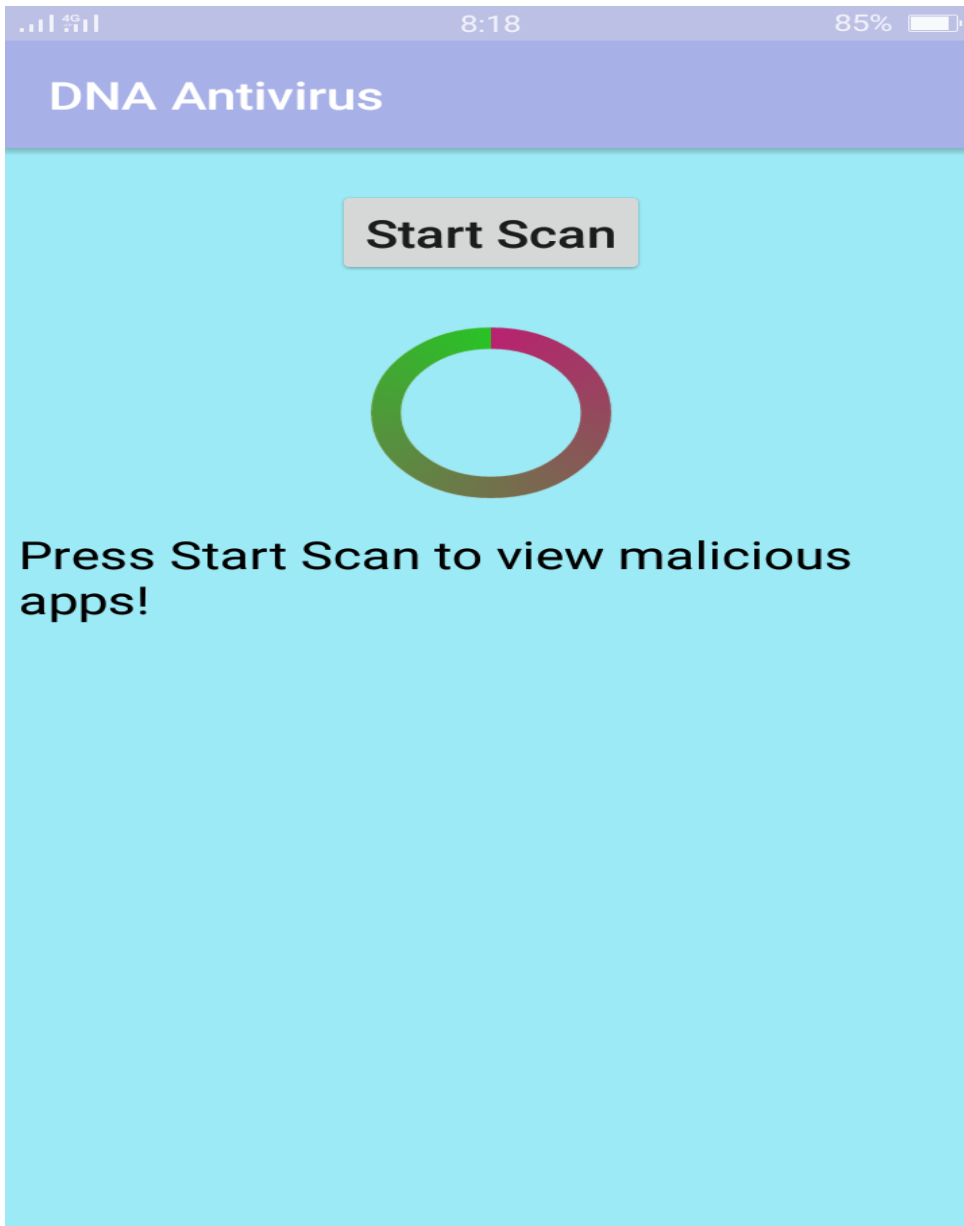
5.4.2 Maintainability: The developer of DNA antivirus can make changes and add new features to DNA antivirus.

3.3. External Interface Requirements

User Interfaces

Responsive graphical user interfaces must be provided to user to work with the application. Here are few dummy screenshots of the application:

Main Screen:



Android Antivirus

Scanning Screen:

Threat List Screen:

1: HIGH RISK APPLICATIONS:

2. MEDIUM RISK APPLICATIONS:

Android Antivirus

Android Antivirus

3.LOW RISK APPLICATIONS:

Trust/Uninstall Screen:

Chapter 4. Design and Development

4.1. Introduction

The document SDS (SOFTWARE DESIGN SPECIFICATION) shows the system design and architecture of our antivirus “DNA ANDROID ANTIVIRUS”. It usually shows design detail

Android Antivirus

diagrams and the explanations. This is done to help all the possible stakeholders to get to know about the design and detail of the product (DNA ANTIVIRUS). The document will also take part in helping the maintainers and developers in their field respectively.

1.1 Purpose

The document describes the software design and specification for Android Antivirus (DNA Antivirus). This document will also show the interface of DNA antivirus.

4.1 Document Conventions

- The document text is shown in Times New Roman 12.
- The space between document text is single space and 1.5" margin distance is maintained in between.
- We used New Times New Roman with font 14 for all the main headings of the document.

1.4 Intended Audience and Reading Suggestions

This document is made for customers, developers and all the possible stakeholders involved. The reader of document must have a basic knowledge of android, malware and anti-viruses.

1.5 Project Scope

As with the increase in technology, android is becoming popular day by day and number of users of android is also increasing on daily basis. As the number of users of android is increased, the numbers of malicious attacks are also increasing day by day. The purpose of these attacks is to steal private information and consume credit by subscribing to some of the premium services. There are many antivirus solutions available but these antivirus

Android Antivirus

mostly uses signature base algorithms which will fail if some new virus comes to market, as they will not have the signature of that malicious application, so they will not be able to find zero days attack. So in the light of this issue, in our thesis we present a behavioral base antivirus that is use for detecting both malware and zero days attack. Our antivirus will be able to detect those zero days attack that other antivirus will fail to do it. Our antivirus will outperform other such applications that are running on android. Though it is a beginner level so our antivirus will have minimum functionalities but the developers and maintainers can work on them in near future.

1.7 Overview of Document

Our document will focus on identifying and specifying the high level view of our system architecture. It will also show the interaction of user with the system. The document will also focus on the details of low level view of components of DNA software and their interactions with each other.

The purpose of this document is to give a high level design framework which will help us to build the project DNA antivirus. The document will also help to check the final product by listing all of it's requirements and tell us if we have implemented our project successfully or not..

Our main emphasis of the document is system architecture description as it gives the overview of system's main components and its architecture.

The Pseudo code section is to provide pseudo code so to provide the planned operations of some components. This thing is beyond our scope in this beginning version.

2. System Architecture Description

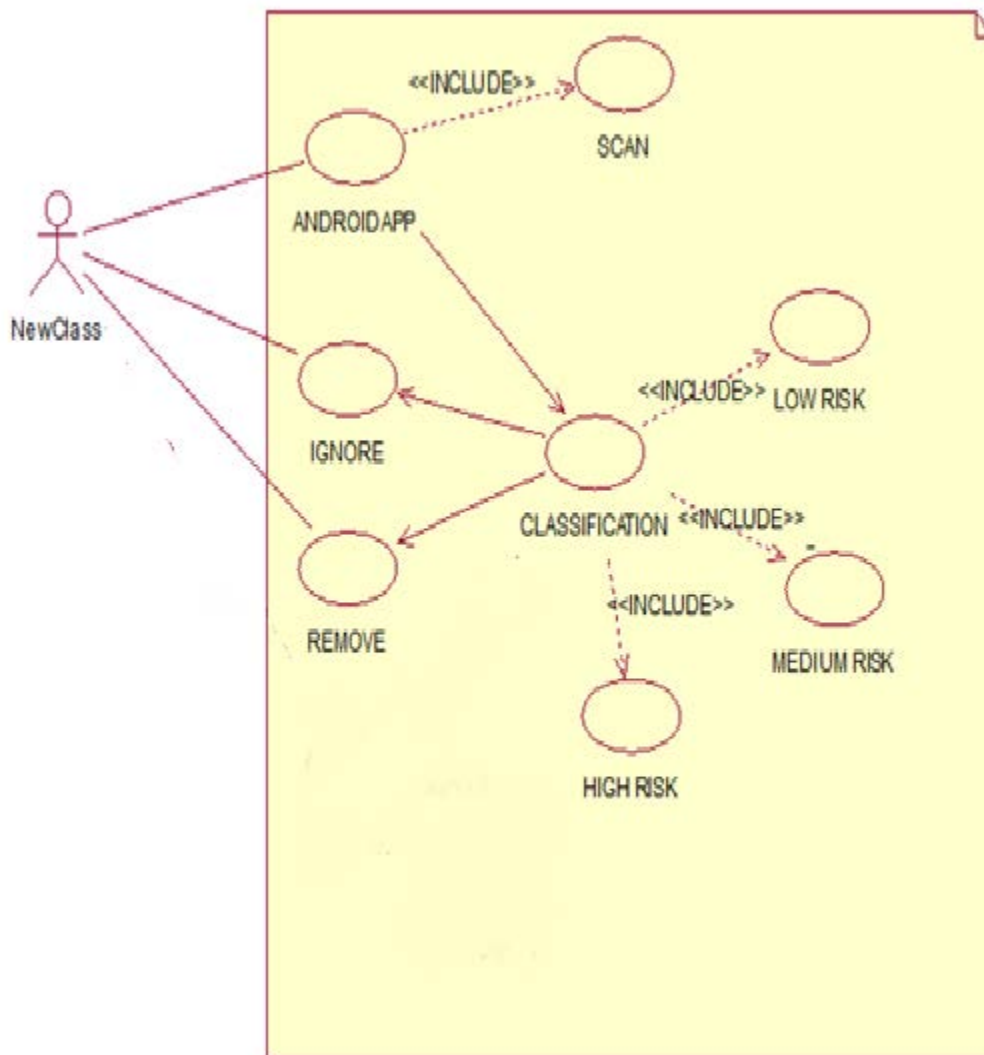
2.1 Structure and relationships

This section covers the overall technical description of DNA ANTIVIRUS. It shows the working of application in perspective of different viewpoints and shows relationships between different components.

2.2.1 Static View

Use Case Diagram

Following diagram shows functionalities of the system as the actor interacts with system.



2.2.1.1 Use case for DNA ANTIVIRUS

Install

Use Case ID	1		
Use Case Name	Install		
Actors	User		
Created By	Group Members	Last Updated By	Group Members
Date Created	8/1/2018	Last Updated	8/1/2018
Description	The actor downloads and installs the antivirus from google play store.		
Pre-Conditions	Android device, 2GB RAM.		
Post-Conditions	Installation completed.		
Normal Flow(Primary Scenario)	<p>The use case starts when an actor opens the mobile application.</p> <p><input type="checkbox"/> The actor opens the google playstore.</p> <p><input type="checkbox"/> Downloads the antivirus.</p> <p><input type="checkbox"/> Installs the antivirus.</p>		

Scan

Use Case ID	2		
Use Case Name	Scan		
Actors	User		
Created By	Group Members	Last Updated By	Group Members
Date Created	8/1/2018	Last Updated	8/1/2018
Description	The actor opens the application on android device.		
Pre-Conditions	<ol style="list-style-type: none"> 1. Android device. 2. Application is opened. 		
Post-Conditions	If the use case is successful then Scan will start during which the application will look for malware on the device or selected applications Scan button.		
Normal Flow(Primary Scenario)	<ol style="list-style-type: none"> 1. The use case starts when an actor selects the Scan button 2. After the scan is complete a list of the detected malware will be categorized into three categories i.e High Risk, Medium Risk and Low Risk 3. Results will be displayed to the user 		

Update

Use Case ID	3
--------------------	---

Android Antivirus

Use Case Name	Update		
Actors	Admin		
Created By	Group Members	Last Updated By	Group Members
Date Created	8/1/2018	Last Updated	8/1/2018
Description	The user downloads the updates from google playstore.		
Pre-Conditions	<ol style="list-style-type: none"> 1. Android device. 2. Android antivirus is installed 		
Post-Conditions	Updates are successfully installed.		
Normal Flow(Primary Scenario)	The user downloads and installs the updates from google playstore.		

Delete

Use Case ID	4		
Use Case Name	Delete		
Actors	User		
Created By	Group Members	Last Updated By	Group Members
Date Created	8/1/2018	Last Updated	8/1/2018

Android Antivirus

Description	Scan has been performed and record of the detected malware threat is present in the log.
Pre-Conditions	The Antivirus has scanned all the applications in the system and categorized them into different threat level.
Post-Conditions	The message “the detected malware threat with high risk which you selected has been deleted/uninstalled” will be displayed.
Normal Flow(Primary Scenario)	<ol style="list-style-type: none"> 1. Scan is successful. 2. Threats has been detected. 3. Display Scan results to the user. 4. Categorize the detected threats into three levels, High, Medium, Low. 5. The user deletes the malicious application which he/she wants.

Trust

Use Case ID	5		
Use Case Name	Trust		
Actors	User		
Created By	Group Members	Last Updated By	Group Members
Date Created	8/1/2018	Last Updated	8/1/2018
Description	The actor trusts the application with threat.		

Android Antivirus

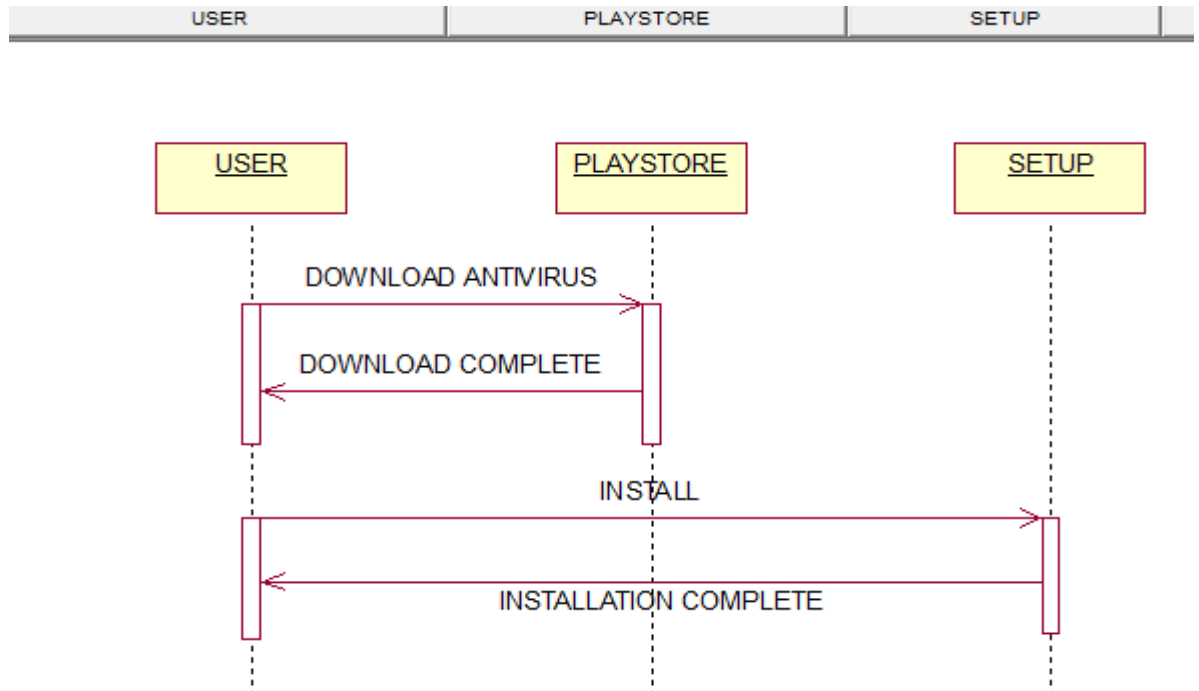
Pre-Conditions	The Antivirus has scanned all the applications in the system and categorized them into different threat level.
Post-Conditions	The message “The application with threat has been trusted” is displayed.
Normal Flow(Primary Scenario)	<ol style="list-style-type: none">6. S can is successful.7. Threats has been detected.8. Display Scan results to the user.9. Categorize the detected threats into three levels, High, Medium, Low.1. The user trusts the malicious application which he/she wants.

2.2.2 Dynamic View

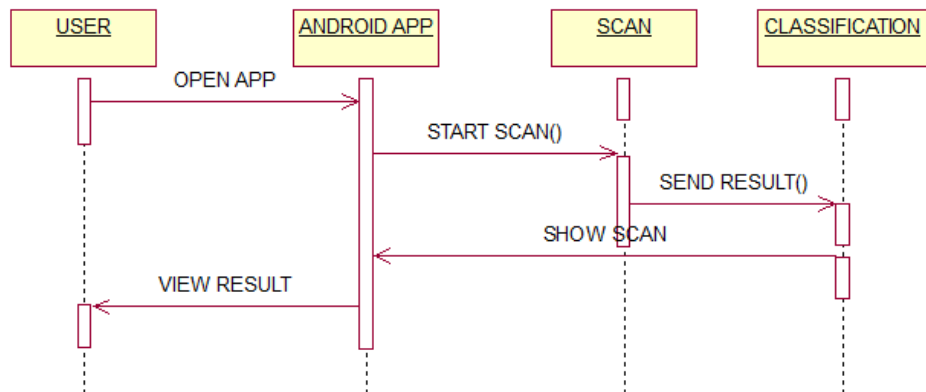
Sequence Diagram

INSTALL :

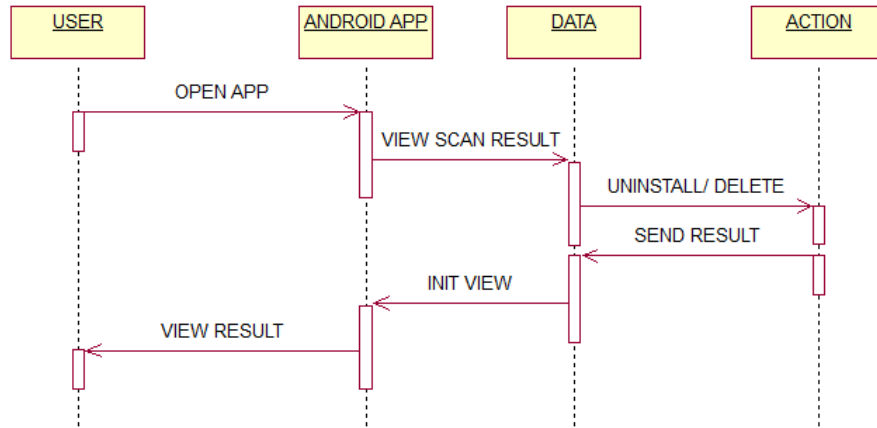
Android Antivirus



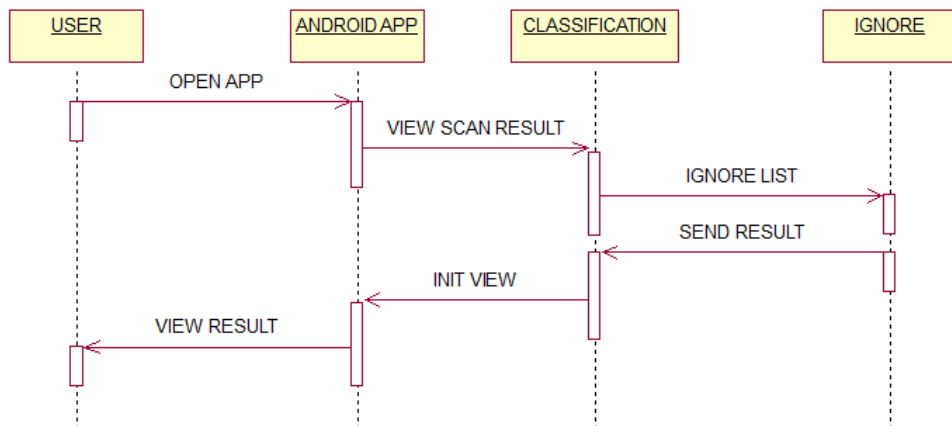
SCAN :



UNINSTALL/DELETE:



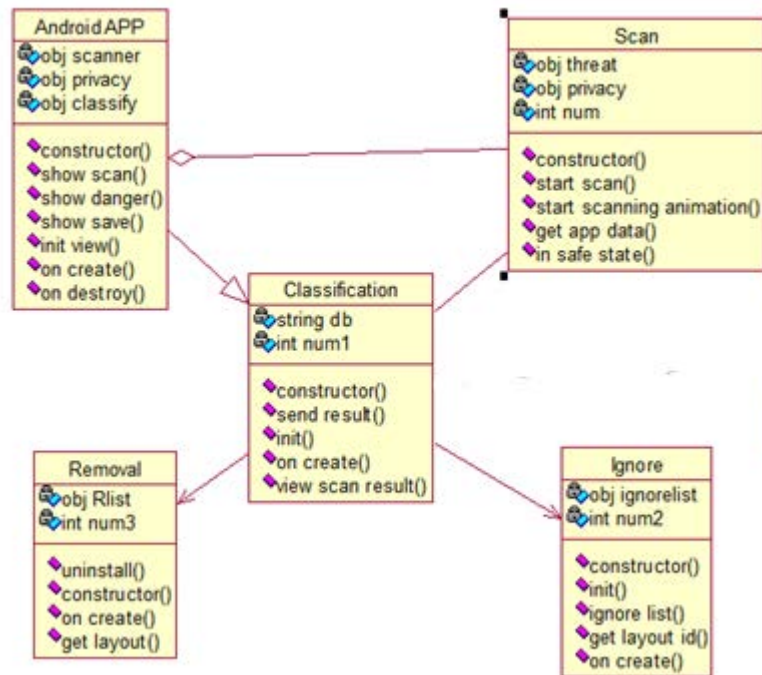
IGNORE / TRUST :



2.2.4 Logical View Point

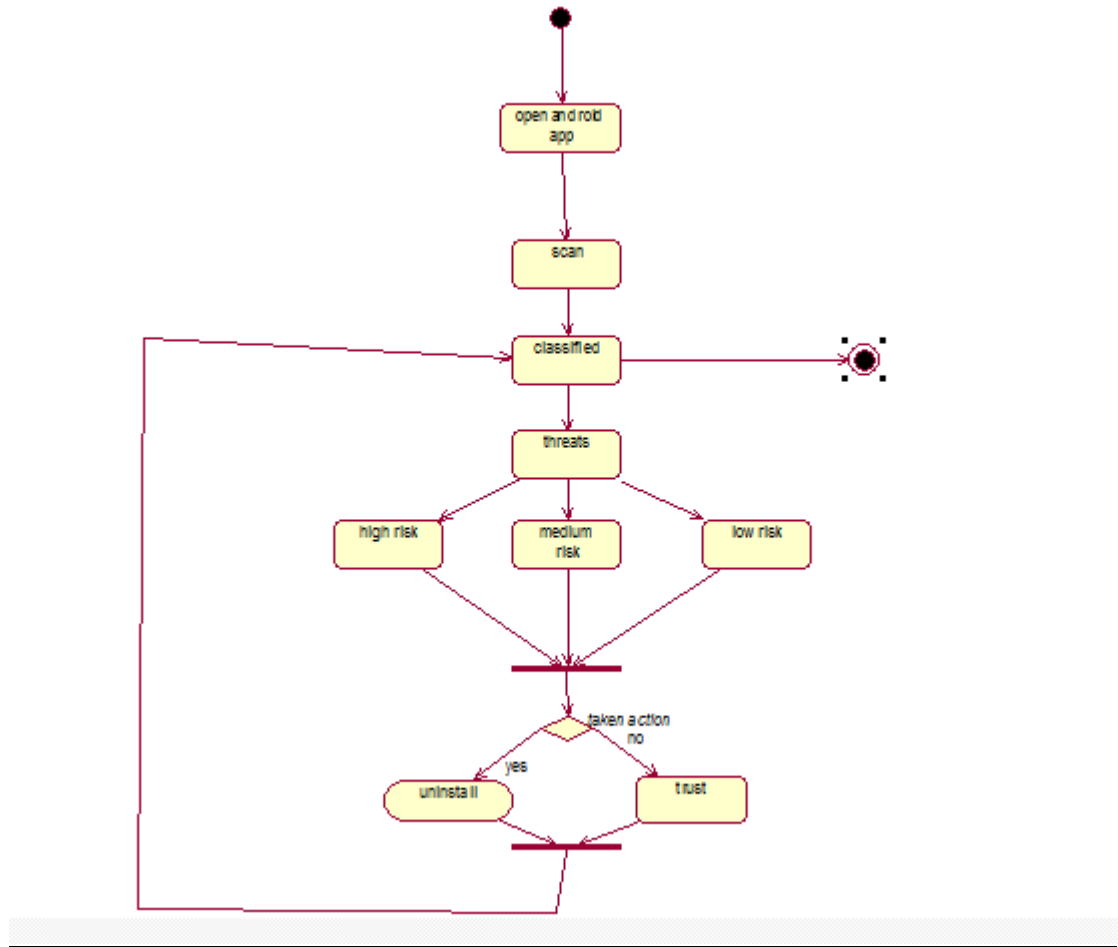
Class Diagram

Android Antivirus



ACTIVITY DIAGRAM :

Android Antivirus

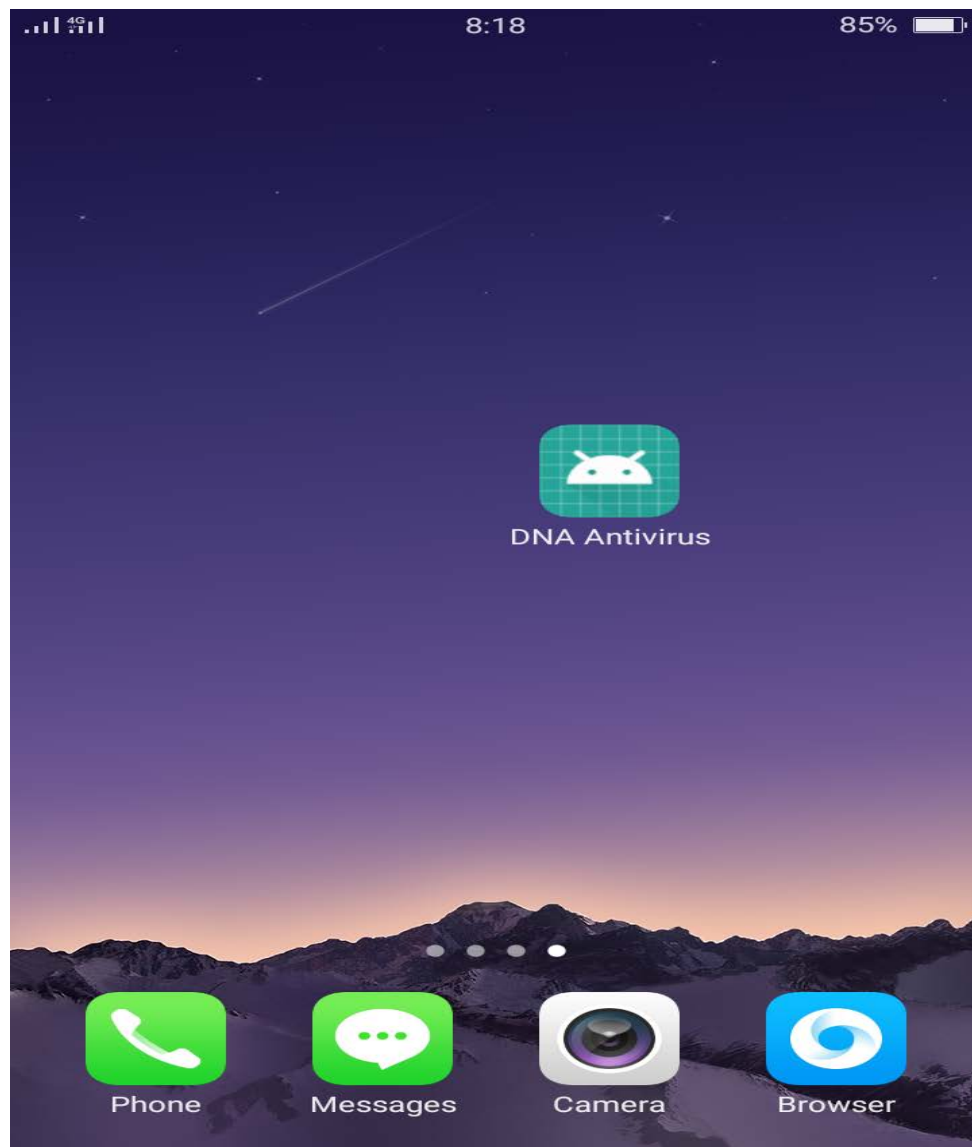


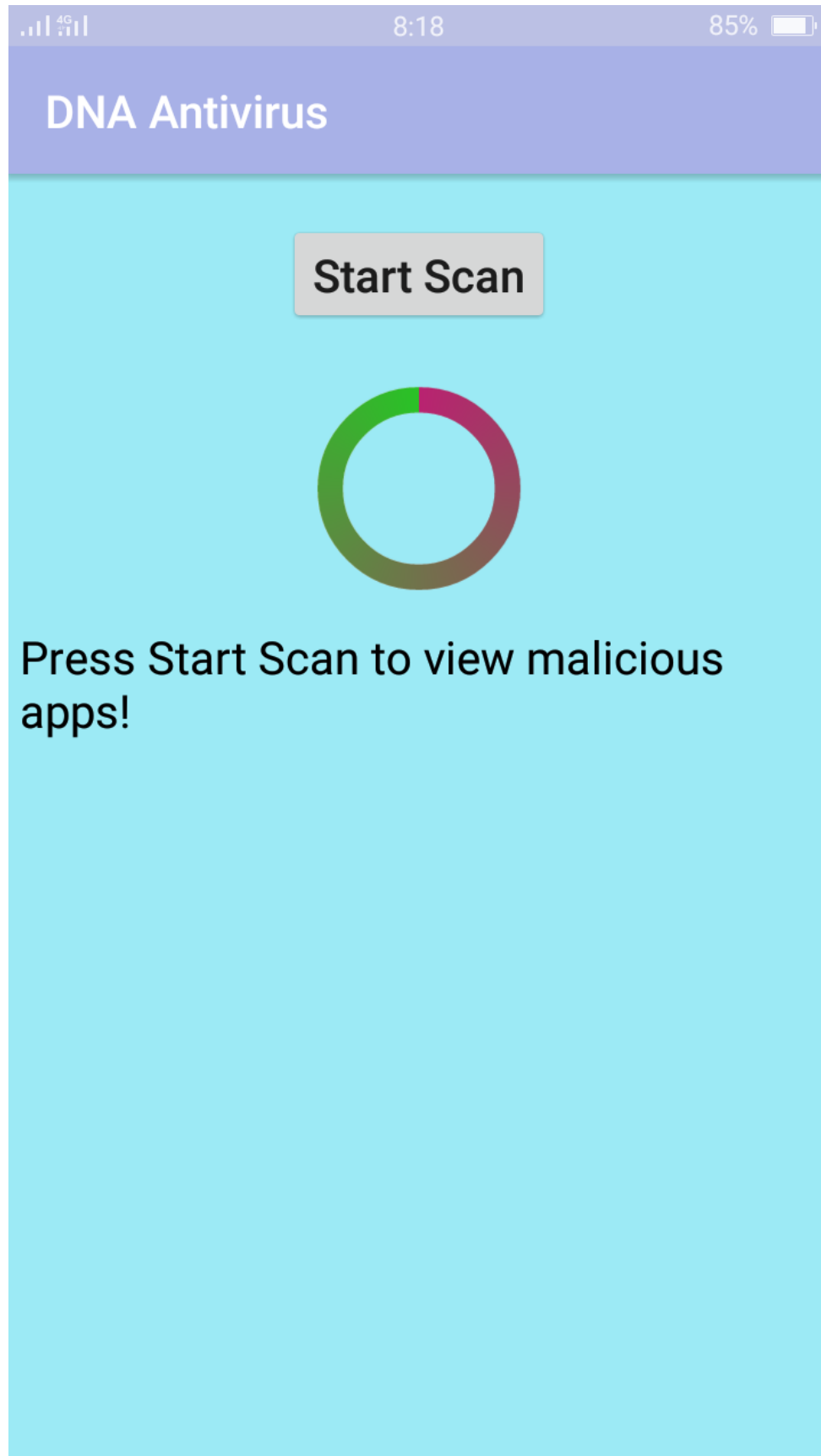
3. User Interface

3.1 Interface (GUIs)

Generally Telemap consists of two main interfaces i.e. Mobile app interface and server or website interface. Purposed interface of android app is given below

3.1.1 Mobile app Interface





4. Reuse and Relationship to other Products

The main theme of this project is to monitor the QoS parameters of all the telecommunication providers i.e. Ufone, Mobilink, Telenor, Zong and Warid.

Signal strength, Location and time are the main parameters which is added to the scope of this project, measure across the maximum achievable characteristics using android mobile phone.

The calculated data will be transmitted to the GIS based server application which will plot the Signal strength of different service providers by using statistical and graphical locations. On the server side, GIS based mapping, application will receive the data and present it in different analysis modes

This app is simple and install to go system that can reduce the effort required to look for a best available network in some specific area.

5. Design Decisions and Tradeoffs

This android app can be used in different aspects for example if any user is interested in buying any cellular network i.e. Ufone, Mobilink, Warid, Zong and Telenor. This can be helpful for them. It also important for anyone who wants information regarding any cellular company performances i.e. their signal strength etc.

We can provide this tool to different cellular companies where they can check their performance. It can be helpful regarding complaints issues for cellular companies.

6. Bibliography

- [http://www.wikipedia.com/signal-strength measurements](http://www.wikipedia.com/signal-strength%20measurements)
- <https://books.google.com/?hl=en>

Android Antivirus

- <http://www.squidoo.com/>
- [http://www.academia.edu/2554765/Measurements and QoS Analysis of Live-World Mobile Telecommunication Networks](http://www.academia.edu/2554765/Measurements_and_QoS_Analysis_of_Live-World_Mobile_Telecommunication_Networks)

Appendix A – Project Log

PROJECT LOG

- **Project Title:** ANDROID ANTIVIRUS (DNA ANTIVIRUS)
- **Supervisor :** Lec WALEED BIN SHAHID
- **Students Names:** MAJ NAJAM, CAPT AWAIS, GC DANYAL

Date & Time	Agenda of Meeting	Status of Task Assigned in Last Meeting	New Tasks For Next Meeting	Signature of Supervisor
1/11/2017	Discussion about SRS	SRS	Completion of SRS	
6/11/2017	SRS report check	Finalization of SRS	SRS updating	
	Finalization of			

Android Antivirus

10/11/2017	SRS	SRS completed	Design Documentation	
20/01/2018	Finalization of SDS Document	SDS Completed	Implementation	

Appendix B – Project Proposal

1. Project Title:

“ANDROID ANTIVIRUS”

2. Objective/Brief Description:

The objective of DNA antivirus is to

- Ability to scan the virus in the system.
- Ability to tackle zero day attacks.
- Ability to alter privacy settings
- Ability to remove the virus from the system.
- Supported by Android (All Versions).

3. Motivation:

As increase in technology, android user's has also increased and more and more applications are coming to android play source through means of many sources and downloads of these applications are also increasing on daily basis.

With the increase in popularity of android, attackers are also increasing their part by increasing their attacks on daily basis.

So keeping this in mind, to secure a system and device there must be some remedy and precautions, antiviruses are made to cure these attacks, there are many antiviruses in the market, mostly of them are signature based antivirus which are only good for old malware but for new malwares whose signatures are not present , these antiviruses fail dramatically.

So keeping it in mind we make a behavioral based antivirus that work on algorithms rather than signatures.

Scope:

As with the increase in technology, android is becoming popular day by day and number of users of android is also increasing on daily basis. As the number of users of android is increased, the numbers of malicious attacks are also increasing day by day. The purpose of these attacks is to steal private information and consume credit by subscribing to some of the premium services. There are many antivirus solutions available but these antivirus mostly uses signature base algorithms which will fail if some new virus comes to market, as they will not have the signature of that malicious application, so they will not be able to find zero days attack. So in the light of this issue, in our thesis we present a behavioral base antivirus that is use for detecting both malware and zero days attack. Our antivirus will be able to detect those zero days attack that other antivirus will fail to do it. Our antivirus will outperform other such applications that are running on android.

5. Objective:

The core objective of our Android Antivirus is to tackle the zero day attacks which can only be prevented using a behavior based Antivirus which uses a specially designed algorithm to study the behavior of the system calls, strings, permissions and combinations of permissions to analyze the application whether is it safe to run or malicious. Normally the Antivirus which are available on the internet and the typically used Antiviruses which are used by everyone are signature based which uses available signatures on the internet of predetermined virus patches to find out the malicious application.

Android Antivirus

6. Functionalities:

Functionalities are well described above, under the overall description part.

7. Previous Work Done on the Subject: Signature based Antiviruses are common but behavior based Antivirus has not been given much attention in Pakistan especially because of its complications.

8. Material Resources Required:

Hardware:

Android mobile

Software:

Java platform, Apk studio, Eclipse, dex2jar, jd gui. Android Studio

Discipline:

Information security, Android programming

Operating System:

Android (all versions)

9. No of Student Required: 03











10. Special Skills Required:

- Java language

Android Antivirus

- Scripting
- Sound knowledge on information security
- Knowledge about malwares and their types

Appendix C – Project Timeline

	MARCH	MAY	JUNE	SEP	NOV-DEC	2018	JAN-FEB	MARCH-APRIL	MAY
SELECTION									
PROPOSAL									
DEFENCE									
LITERATURE REVIEW									
REQUIREMENT ELICITATION									
DESIGN AND DEVELOPMENT									
CODING									
DOCUMENTATION									
TESTING									
IMPLEMENTATION									

Appendix D – Rules for Algorithm

	Rule #	Features in each Rule
Reading Phone Credentials	1a	android.permission.READ_PHONE_STATE
		android.permission.INTERNET (conn)
		android.permission.INTERNET (upload)
		Making File
		Contacting a Remote Server
	1b	android.permission.READ_PHONE_STATE
		android.permission.INTERNET (Upload)
		Zip Archive <i>OR File</i>
	1c	android.permission.READ_PHONE_STATE
		android.permission.INTERNET (conn)
		Zip Archive <i>OR</i>
	1d	android.permission.READ_PHONE_STATE
		android.permission.INTERNET (conn)
	1e	android.permission.READ_PHONE_STATE
	Sending SMS	2a
android.permission.INTERNET (conn)		
android.permission.INTERNET (upload)		
Zip Archive		

Android Antivirus

		Making File	
		Contacting a Remote Server	
	2b	android.permission.SEND_SMS	
		android.permission.INTERNET (Upload)	
		Zip Archive <i>OR</i> Making File	
	2c	android.permission.SEND_SMS	
android.permission.INTERNET (conn)			
Zip Archive <i>OR</i> Making File			
2d	android.permission.SEND_SMS		
	android.permission.INTERNET (conn)		
2e	android.permission.SEND_SMS		
Reading Contacts	3a	android.permission.READ_CONTACTS	
		android.permission.INTERNET (conn)	
		android.permission.INTERNET (upload)	
		Zip Archive	
		Making File	
		Contacting a Remote Server	

Android Antivirus

	3b	android.permission.READ_CONTACTS	
		android.permission.INTERNET (Upload)	
		Zip Archive <i>OR</i> Making File	
	3c	android.permission.READ_CONTACTS	
		android.permission.INTERNET (conn)	
		Zip Archive <i>OR</i> Making File	
	3d	android.permission.READ_CONTACTS	
		android.permission.INTERNET (conn)	
	3e	android.permission.READ_CONTACTS	
	Access Wi-Fi State	4	android.permission.ACCESS_WIFI_STATE
	android.permission.INTERNET (conn)		
Listening to Incoming messages	5a	android.permission.RECEIVE_SMS	
		android.permission.INTERNET (conn)	
		android.permission.INTERNET (upload)	
		Zip Archive	
		Making File	
		Contacting a Remote Server	
	5b	android.permission.RECEIVE_SMS	
android.permission.INTERNET (Upload)			

Android Antivirus

		Zip Archive OR Making File	
	5c	android.permission.RECEIVE_SMS	
		android.permission.INTERNET (conn)	
		Zip Archive OR Making File	
	5d	android.permission.RECEIVE_SMS	
		android.permission.INTERNET (conn)	
	5e	android.permission.RECEIVE_SMS	
Installing Packages	6a	copy Assets	
		android.permission.INSTALL_PACKAGES	
		chmod 775 OR Root Shell	
	6b	copy Assets	
		chmod 775 OR Root Shell	
		copy Assets	
	6c	android.permission.INSTALL_PACKAGES	

Android Antivirus

Encrytion	7	Hashing SHA1
		Key Ciphers
		PRNG
		Hashing MD5
Enabling USB Mode	8	Enabling USB Mode
Installing other Applications	9a	action.download_apk
		intsall apps
		Download files
	9b	action.download_apk
Rooting/Shell Scripts	10a	action.download_shells
		chmod
		Root Shell
	10b	action.download_shells
		Root Shell
	Phone State and Audio	11
android.permission.RECORD_AUDIO		
android.permission.INTERNET (conn)		
Access FINE Location	12a	android.permission.ACCESS_FINE_LOCATION
		RECEIVE_BOOT_COMPLETED

Android Antivirus

		android.permission.INTERNET (conn)	
	12b	android.permission.ACCESS_FINE_LOCATION	
		android.permission.INTERNET (conn)	
Access CORASE Location	13a	android.permission.ACCESS_COARSE_LOCATION	
		RECEIVE_BOOT_COMPLETED	
		android.permission.INTERNET (conn)	
	13b	android.permission.ACCESS_COARSE_LOCATION	
android.permission.INTERNET (conn)			
Receive/Write SMS	14	android.permission.RECEIVE_SMS	
		android.permission.WRITE_SMS	
Send/Write SMS	15	android.permission.SEND_SMS	
		android.permission.WRITE_SMS	
CALL_PRIVILEGED	16	android.permission.CALL_PRIVILEGED	
Calling Phone	17a	android.permission.CALL_PHONE	
		Internet	
		android.permission.PROCESS_OUTGOING_CALL	
	17b	android.permission.CALL_PHONE	
Internet			

Android Antivirus

	17c	android.permission.PROCESS_OUTGOING_CALL
		Internet
	17d	android.permission.CALL_PHONE
	17e	android.permission.PROCESS_OUTGOING_CALL
Accessing Web and Social Media	18a	Accessing Twitter
		Internet
		Web Search
	18b	Accessing Twitter
		Internet
Uploading Bluetooth data	19a	Bluetooth
		Upload Files
	19b	Bluetooth
Starting Service	20	Service
		BOOT_COMPLETED
		INTERNET
Accessing MOCK Location	21a	android.permission.ACCESS_MOCK_LOCATION
		RECEIVE_BOOT_COMPLETED
		android.permission.INTERNET (conn)

Android Antivirus

	21b	android.permission.ACCESS_MOCK_LOCATION	
		android.permission.INTERNET (conn)	
Writing to External Storage	23	WRITE_EXTERNAL_STORAGE	
Web Search	24	WEB_SEARCH	
		DOWNLAOD	
		INTERNET	
Contacting Server	25	SERVER	
		INTERNET	
Installed Packages	26	LIST OF PACKAGES	
		INTERNET	
Reading Logs	27	READ_LOGS	
		INTERNET	
Mounting/Un-mounting File Systems	28a	MOUNTING/UNMOUNTING	
		INTERNET	
	28b	MOUNTING/UNMOUNTING	
SET_DEBUG_APPS	29a	SET_DEBUG_APPS	
		INTERNET	
	29b	SET_DEBUG_APPS	
Format File Systems	30a	FORMAT_FILE_SYSTEMS	

		INTERNET
	30b	FORMAT_FILE_SYSTEMS
Delete Packages	31	android.permission.DELETE_PACKAGES
Write Settings	32	android.permission.WRITE_SETTINGS
Write Security Settings	33	android.permission.WRITE_SECURE_SETTINGS

Chapter 5. Project Test and Evaluation

5.1. Introduction

Any device without using good security is considered unsafe, now-a-days. So, there is a strong need for security precautions to get rid of dangerous attacks and threats to protect your device.

DNA antivirus sponsored by NESCOM is also an advancement towards better security for android systems. With the help of this antivirus, user will be able to scan there whole system or can perform a custom scan. It will provide them the ability to uninstall the applications that are on risk and can provide some sort of harm to your system. The antivirus will categorize the applications in term of it's risk. We categorize this as follows

- High level risk
- Medium level risk
- Low level risk.

Our application has a simple interface that can be used by anyone having a little knowledge of android phones.

This test plan document describes the appropriate process and methods used to plan testing of the ANDROID ANTIVIRUS project. The test plan will ensure that DNA ANTIVIRUS works as intended without any failure.

We are performing manual testing of the project i.e.; no tool or script is being utilized for testing purpose. Usually this type of testing is carried out when the tester takes the character of an end user and examine the software to classify any unpredicted behavior or may be bug. First of all, each unit (module) will be tested separately and then whole project will be tested after integration of all the units.

5.2. Test Items

Based on the requirements of ANDROID ANTIVIRUS, design description, modules of android application, and non-functional scenario will be tested. The requirements defined in Software Requirements Specification and the design entities as explained in Software Design Document will be tested.

5.3. Features to Be Tested

Following features are being tested:

1. Ability to do full or custom scan of android phone.
2. Ability to detect threats and malicious file thorough scanning.
3. Ability to test whether the application is having highlevel risk, medium level risk, or low level risk.
4. Ability to check which application is using our private things like location phone book etc.
5. Ability to tell user about threat level and ask the user to uninstall it.
6. Ability to check the overall efficiency of the system.

5.4. Test Approach

Black Box testing technique will be used for testing functionality of each module. In black box testing we inspect the functionality of an app without getting in details.

Unit Testing

It is that part of testing which requires a thorough check of each module of the project. In our project, there are 6 modules which we have to check if they are functioning normally or not. For this, we will start from a unit which is least dependent on other modules for its

function and then work our way through to the module which requires all the rest to function and test.

Integration Testing

Integration testing is the part where we will test all the previous tested modules in a way that they are functioning normally when they are combined together.

System Testing

In the end, system testing will ensure that all the modules are working, separately and together combined. Then only the final outcome of the program will decide the correctness of whole system.

5.5. Item Pass/Fail Criteria

Details of the test cases are specified in the section Test Deliverables. Following the principles outlined below, a test item would be judged as pass or fail.

- Preconditions are met
- Inputs are carried out as specified
- The result works as what specified in output => Pass
- The system doesn't work or not the same as output specification => Fail

5.6. Suspension Standards and Resumption Requirements

Test case will be stopped and suspended whenever a defect and issue is found because of which we cannot move further. We can again carryout this testing afyter the removal of that defect.

5.7. Test Deliverables

Test case name	SCAN
Test Case Number	1
Description	Scanning

Android Antivirus

Testing Technique used	Black Box Testing
Preconditions	1) User should have android smartphone.
Input	Scan option
Steps	<ol style="list-style-type: none"> 1. Open DNA antivirus. 2. Enter Scan option on home screen.
Expected output	User will have a full scanning of it's system.
Actual output	System scan

Test case name	MALICIOUS APPLICATIONS
Test Case Number	2
Description	Threats detect
Testing Technique used	Black Box Testing
Preconditions	<ol style="list-style-type: none"> 1) Smartphone should be android smartphone. 2) It should be scanned through DNA antivirus.
Input	Scan
Steps	<ol style="list-style-type: none"> 3. Open DNA antivirus 4. Start scanning your system. 5. System will show threats which has been detected.

Android Antivirus

Expected output	Malicious Applications
Actual output	Malicious Applications

Test case name	RISK LEVEL
Test Case Number	3
Description	High Level Risk Apps.
Testing Technique used	Black Box Testing
Preconditions	1) Smartphone should be android smartphone. 2) It should be scanned through DNA antivirus.
Input	Protect.
Steps	<ol style="list-style-type: none"> 1. Open DNA antivirus 2. Start scanning your system. 3. System will show threats which has been detected 4. Open these threats 5. Antivirus will show all applications with high level risk.
Expected output	Applications with high level risk
Actual output	All applications with high level risk

Android Antivirus

Test case name	Risk Level
Test Case Number	4
Description	Medium Level Risk Apps.
Testing Technique used	Black Box Testing
Preconditions	1) Smartphone should be android smartphone. 2) It should be scanned through DNA antivirus.
Input	Protect.
Steps	<ol style="list-style-type: none">1. Open DNA antivirus2. Start scanning your system.3. System will show threats which has been detected4. Open these threats5. Antivirus will show all applications with medium level risk.
Expected output	Applications with medium level risk

Android Antivirus

Actual output	All applications with medium level risk
----------------------	---

Test case name	Risk Level
Test Case Number	5
Description	low Level Risk Apps.
Testing Technique used	Black Box Testing
Preconditions	1) Smartphone should be android smartphone. 2) It should be scanned through DNA antivirus.
Input	Protect.
Steps	<ol style="list-style-type: none"> 1. Open DNA antivirus 2. Start scanning your system. 3. System will show threats which has been detected 4. Open these threats 5. Antivirus will show all applications with low level risk.
Expected output	Applications with low level risk
Actual output	All applications with low level risk

Test case name	Uninstall malicious applications
Test Case Number	6
Description	User can uninstall malicious applications
Testing Technique	Black Box Testing

Android Antivirus

used	
Preconditions	<ol style="list-style-type: none"> 1) User will scan the system. 2) User will get the applications which are on list.
Input	Uninstall app
Steps	<ol style="list-style-type: none"> 1. User will see risky application 2. User wil uninstall risky applications
Expected output	Application with risk will be uninstalled.
Actual output	User will decide whether to kee or uninstall that application

Test case name	Uninstall malicious applications
Test Case Number	6
Description	User can uninstall malicious applications
Testing Technique used	Black Box Testing
Preconditions	<ol style="list-style-type: none"> 3) User will scan the system. 4) User will get the applications which are on list.
Input	Uninstall app
Steps	<ol style="list-style-type: none"> 3. User will see risky application

Android Antivirus

	4. User wil uninstall risky applications
Expected output	Application with risk will be uninstalled.
Actual output	User will decide whether to kee or uninstall that application

Test case name	Apps Ignored
Test Case Number	7
Description	User can ignore and uninstall the risky application
Testing Technique used	Black Box Testing
Preconditions	<ol style="list-style-type: none"> 1) user will scan the system. 2) User will see the risky applications 3) User will either uninstall the application or may contain it.
Input	See app ignored or contained history
Steps	<ol style="list-style-type: none"> 1. Select app ignored or contained history
Expected output	DNA antivirus history
Actual output	No of apps ignored

Test case name	Phone information
Test Case Number	8
Description	User can see the information about his system

Testing Technique used	Black Box Testing
Preconditions	1) open DNA antivirus
Input	PHONE INFO
Steps	<ol style="list-style-type: none"> 1. Enter phone info 2. Check information of your system
Expected output	Information of the system
Actual output	CPU, RAM , External storage condition of the system and some basic information

5.8. Responsibilities, Staffing and Training Needs

Responsibilities

Every single person who has a part in the development of this project has a responsibility of completing all the integration and unit testing tasks.

Staffing and Training Needs

For testing the project, a basic knowledge related testing strategies and it's techniques is needed.

Techniques such as Black Box testing, integration testing should be known to developers.

The developers involve in the product should test each other work and should actively perform and participate in testing and development of the product.

5.9. Risk and Contingencies

Efforts have been made to remove all and every chance of failure but there are certain unpredictable factors such as network issues, corrupt input data, or system failure that may lead to some issues. Error handling will be applied more deeply to cover all these issues but unforeseen circumstances may happen.

Schedule Risk

The project might get behind schedule. So, in order to complete the project on time, we will need to increase the hours/day.

Budget Risk

We will arrange our budget is such way that we will use less expensive alternatives to fulfill our requirements.

Conclusion and Future Work

As the number of users of android is increased, the numbers of malicious attacks are also increasing day by day. The purpose of these attacks is to steal private information and consume credit by subscribing to some of the premium services. There are many antivirus solutions available but these antiviruses mostly uses signature base algorithms which will fail if some new virus comes to market, as they will not have the signature of that malicious application, so they will not be able to find zero days attack. So in the light of this issue, in our thesis we present a behavioral base antivirus that is use for detecting both malware and zero days attack. Our antivirus will be able to detect those zero days attack that other antivirus will fail to do it. Our antivirus will outperform other such applications that are running on android.

As technology changes dramatically so n future the developers of this product can make changings and can enhance the level of antivirus as keeping in mind the change in trends of technology.

Bibliography

- [1] Wikipedia. Mobile operating system — wikipedia, the free encyclopedia, 2017. [Online; accessed 7-March-2017].
- [2] Statista. Number of apps available in leading app stores as of june 2016, 2016. [Online; accessed 7-March-2017].
- [3] Sven Bugiel, Lucas Davi, Alexandra Dmitrienko, Thomas Fischer, Ahmad-Reza Sadeghi, and Bhargava Shastry. Towards taming privilege-escalation attacks on android. In NDSS, 2012.
- [4] Renee Shipley. The best anti-malware software of 2017, 2016. [Online; accessed 7-March2017].
- [5] Joe Hindy. 15 best antivirus android apps and anti-malware android apps, 2017. [Online; accessed 7-March-2017].
- [6] Ilsun You and Kangbin Yim. Malware obfuscation techniques: A brief survey. In Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on, pages 297–300. IEEE, 2010.
- [7] Damien Ocateau, Patrick McDaniel, Somesh Jha, Alexandre Bartel, Eric Bodden, Jacques Klein, and Yves Le Traon. Effective inter-component communication mapping in android with epicc: An essential step towards holistic security analysis. In Proceedings of the 22nd USENIX security symposium, pages 543–558, 2013.
- [8] Ed Burnette. Hello, Android: introducing Google’s mobile development platform. Pragmatic Bookshelf, 2009. [9] Kyoochang Jeong and Heejo Lee. Code graph for malware detection. In Information Networking, 2008. ICOIN 2008. International Conference on, pages 1–5. IEEE, 2008.
- [10] Haoran Guo, Jianmin Pang, Yichi Zhang, Feng Yue, and Rongcai Zhao. Hero: A novel malware detection framework based on binary translation. In Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on, volume 1, pages 411– 415. IEEE, 2010.
- [11] Mihai Christodorescu, Somesh Jha, and Christopher Kruegel. Mining specifications of mali59 cious behavior. In Proceedings of the 1st India software engineering conference, pages 5–14. ACM, 2008.
- [12] Asaf Shabtai, Yuval Fledel, and Yuval Elovici. Automated static code analysis for classifying android applications using machine learning. In Computational Intelligence and Security (CIS), 2010 International Conference on, pages 329–333. IEEE, 2010.
- [13] William Enck, Machigar Ongtang, and Patrick McDaniel. On lightweight mobile phone application certification. In Proceedings of the 16th ACM conference on Computer and communications security, pages 235–245. ACM, 2009.
- [14] Min Zheng, Mingshen Sun, and John CS Lui. Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware. In Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on, pages 163–171. IEEE, 2013.
- [15] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an informationflow tracking system for realtime privacy monitoring on smartphones. ACM Transactions on Computer Systems (TOCS), 32(2):5, 2014.
- [16] Yuan Zhang, Min Yang, Bingquan Xu, Zhemin Yang, Guofei Gu, Peng Ning, X Sean Wang, and Binyu Zang. Vetting undesirable behaviors in android apps with permission use analysis. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pages 611–622. ACM, 2013.

- [17] android.com. Permissions, 2017. [Online; accessed 7-March-2017].
- [18] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, and David Lie. Pscout: analyzing the android permission specification. In Proceedings of the 2012 ACM conference on Computer and communications security, pages 217–228. ACM, 2012.
- [19] android.com. Manifest.permission, 2017. [Online; accessed 7-March-2017].
- [20] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In 23rd USENIX Security Symposium (USENIX Security 14), pages 1053–1067, 2014.
- [21] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In Proceedings of the 18th ACM conference on Computer and communications security, pages 627–638. ACM, 2011.
- [22] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. An empirical study on android for saving non-shared data on public storage. In ICT Systems Security and Privacy 60 Protection, pages 542–556. Springer, 2015.
- [23] David Barrera, H Güne, Kayacik, Paul C van Oorschot, and Anil Somayaji. A methodology for empirical analysis of permission-based security models and its application to android. In Proceedings of the 17th ACM conference on Computer and communications security, pages 73–84. ACM, 2010.
- [24] Yajin Zhou and Xuxian Jiang. Dissecting android malware: Characterization and evolution. In Security and Privacy (SP), 2012 IEEE Symposium on, pages 95–109. IEEE, 2012.
- [25] V Babu Rajesh, Phaninder Reddy, P Himanshu, and Mahesh U Patil. Droidswan: Detecting malicious android applications based on static feature analysis. Computer Science & Information Technology, page 163.
- [26] Zarni Aung and Win Zaw. Permission-based android malware detection. International Journal of Scientific and Technology Research, 2(3):228–234, 2013.
- [27] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In NDSS, 2014.
- [28] Justin Sahs and Latifur Khan. A machine learning approach to android malware detection. In Intelligence and security informatics conference (eisis), 2012 european, pages 141–147. IEEE, 2012.
- [29] Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee, and Kuo-Ping Wu. Droidmat: Android malware detection through manifest and api calls tracing. In Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on, pages 62–69. IEEE, 2012.
- [30] Yajin Zhou, Zhi Wang, Wu Zhou, and Xuxian Jiang. Hey, you, get off of my market: detecting malicious apps in official and alternative android markets. In NDSS, volume 25, pages 50–52, 2012.
- [31] B Alll and C Tumbleson. Dex2jar: Tools to work with android. dex and java. class files.
- [32] R Winsniewski. Apktool: a tool for reverse engineering android apk files. URL: <https://ibotpeaches.github.io/Apktool/>(visited on 07/27/2016), 2012.
- [33] Java Decompiler. Jd-gui.
- [34] Timothy Vidas, Nicolas Christin, and Lorrie Cranor. Curbing android permission creep. In Proceedings of the Web, volume 2, pages 91–96, 2011.

- [35] Wikipedia. Chi-squared test — wikipedia, the free encyclopedia, 2016. [Online; accessed 61 7-March-2017].
- [36] Jerome H Friedman. Data mining and statistics: What's the connection? *Computing Science and Statistics*, 29(1):3–9, 1998.
- [37] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.
- [38] Ron Kohavi and Foster Provost. Glossary of terms. *Machine Learning*, 30(2-3):271–274, 1998.
- [39] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.
- [40] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.
- [41] Wikipedia. Principal component analysis — wikipedia, the free encyclopedia, 2017. [Online; accessed 7-March-2017].
- [42] Tian-Yu Liu. Easyensemble and feature selection for imbalance data sets. In *Bioinformatics, Systems Biology and Intelligent Computing*, 2009. IJCBS'09. International Joint Conference on, pages 517–520. IEEE, 2009.
- [43] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
- [44] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.
- [45] Francisco Pereira, Tom Mitchell, and Matthew Botvinick. Machine learning classifiers and fmri: a tutorial overview. *Neuroimage*, 45(1):S199–S209, 2009.
- [46] Peter Bühlmann. Bagging, boosting and ensemble methods. In *Handbook of Computational Statistics*, pages 985–1022. Springer, 2012.
- [47] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
- [48] Michael Collins, Robert E Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1-3):253–285, 2002.
- [49] Jane Elith, John R Leathwick, and Trevor Hastie. A working guide to boosted regression trees. *Journal of Animal Ecology*, 77(4):802–813, 2008.
- [50] MW Browne. Predictive validity of a linear regression equation. *British Journal of Mathe62 matical and Statistical Psychology*, 28(1):79–87, 1975.
- [51] Strother H Walker and David B Duncan. Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179, 1967.
- [52] Lior Rokach and Oded Maimon. *Data mining with decision trees: theory and applications*. World scientific, 2014.
- [53] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.
- [54] Yoav Freund and Llew Mason. The alternating decision tree learning algorithm. In *icml*, volume 99, pages 124–133, 1999.
- [55] Mark A Friedl and Carla E Brodley. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment*, 61(3):399–409, 1997.
- [56] Robert M Gray. *Entropy and information theory*. Springer Science & Business Media, 2011.

- [57] Tin Kam Ho. Random decision forests. In Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on, volume 1, pages 278–282. IEEE, 1995.
- [58] Tin Kam Ho. The random subspace method for constructing decision forests. IEEE transactions on pattern analysis and machine intelligence, 20(8):832–844, 1998.
- [59] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. Machine learning, 63(1):3–42, 2006.
- [60] David Cossock and Tong Zhang. Statistical analysis of bayes optimal subset ranking. IEEE Transactions on Information Theory, 54(11):5140–5154, 2008.
- [61] Jiawei Han, Jian Pei, and Micheline Kamber. Data mining: concepts and techniques. Elsevier, 2011.
- [62] Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. Machine learning, 37(3):277–296, 1999.
- [63] Teuvo Kohonen. An introduction to neural computing. Neural networks, 1(1):3–16, 1988.
- [64] Jürgen Schmidhuber. Deep learning in neural networks: An overview. Neural networks, 61:85–117, 2015.
- [65] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366, 1989.

Appendix A. Glossary

Dex2jar:	it decompiles the dex file
Apk studio:	it decompiles the apk file
Python:	Language for scripting
Static Analysis:	Analyzing without executing the code
Dynamic Analysis:	Analyzing behavior while executing the code

Appendix B: Issues/Limitations

All possible issues have already been mentioned where required in the SRS. Any remaining ones are listed below:

1. The group shall try to match the features and NFRs as best as possible, however, like all software projects, any discrepancies are apologized for at this stage.
2. Feedback on requirements is expected from the users to help the group in improving the design and implementation of the project.