

ANDROID MALWARE DETECTION USING CLOUD COMPUTING

A project Report submitted in partial
fulfilment for the award of the
Degree of Bachelor of Technology

in

Computer Science and Engineering by

G.V.SAI KRISHNA (U19CS371)

G.SANDEEP (U19CS372)

G.L.NARASIMHA (U19CS346)

G.NIKIL RAO (U19CS375)

Under the guidance of

Mr.R.MUTHU VENKATAKRISHNAN., M.E., (Ph.D.)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

BHARATH INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

CHENNAI 600073, TAMILNADU, INDIA

MAY, 2023



Bharath

INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)

BHARATH INSTITUTE OF SCIENCE AND TECHNOLOGY

No.173, Agharam Road, Selaiyur, Chennai - 600 073.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BONAFIDE CERTIFICATE

This is to Certify that this Project Report Titled "**ANDROID MALWARE DETECTION USING CLOUD COMPUTING**" is the Bonafide Work of **G.V SAIKRISHNA(U19CS371), G.SANDEEP(U19CS372), G.L.NARASIMHA REDDY(U19CS346), G.NIKIL RAO(U19CS375)** of Final Year B.Tech. (CSE) who carried out the major project work under my supervision Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on basis of which a degree or award conferred on an earlier occasion by any other candidate.

PROJECT GUIDE

MR.R.MUTHU VENKATA KRISHNAN

Assistant Professor

Department of CSE BIHER

HEAD OF THE DEPARTMENT

Dr.S. Maruthu Perumal

Professor

Department of CSE

BIHER

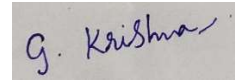
Submitted for the Project Viva-Voce held on ...12-05-2023

INTERNAL EXAMINER

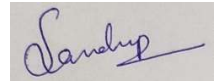
EXTERNAL EXAMINER

DECLARATION

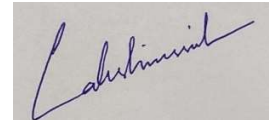
We declare that this project report titled ANDROID MALWARE DETECTION CLOUD COMPUTING submitted in partial fulfilment of the degree of B. Tech in (Computer Science and Engineering) is a record of original work carried out by us under the supervision of SHAMLI R, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.



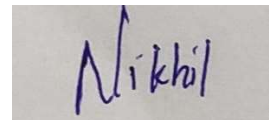
G.V.SAI KRISHNA
(U19CS371)



G.SANDEEP
(U19CS372)



G.L.NARASIMHA REDDY
(U19CS346)



G.NIKILRAPO
(U19CS375)

Chennai

TABLE OF CONTENTS

DESCRIPTION	PAGE NUMBER
CERTIFICATE	i
DECLARATION	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
LIST OF FIGURES	v
LIST OF TABLES	vi

1.	INTRODUCTION
1.1	Overview
1.2	Objective
1.3	Domain Introduction
1.4	Overview of the Project
2.	LITERATURE REVIEW
2.1	Literature Survey
3.	SYSTEM ANALYSIS
3.1	Existing System
3.2	Proposed System
4.	SYSTEM ARCHITECTURE
4.1	System Architecture
4.2	System Requirements
4.3	Hardware Requirements
4.4	Software Requirements
4.5	Modules
4.6	Java
4.7	Collection Framework
4.8	Non-Functional Requirements

5. SYSTEM DESIGN AND TESTING PLAN
 - 5.1 Input Design
 - 5.2 Output Design
 - 5.3 System Testing
 - 5.4 Types of Testing
 - 5.5 Data Flow Diagram
 - 5.6 UML Diagrams
6. SYSTEM DESIGN
 - 6.1 Architecture Diagram
 - 6.2 Data Flow Diagram
 - 6.3 UML Diagram
 - 6.3.1 USECASE Diagram
 - 6.3.2 Class Diagram
 - 6.3.3 Sequence Diagram
 - 6.3.4 Activity Diagram
7. APPENDIX
 - 7.1 Sample Code
 - 7.2 Result Analysis
 - 7.3 Sample Output
8. Conclusion
9. References

LIST OF FIGURES

FIG.NO	FIGURES
4.1.1	SYSTEM ARCHITECTURE
4.5.2	PENTINUM DUAL CORE
4.7.1	COLLECTION FRAMEWORK
5.5.1	DATA FLOW DIAGRAM
5.6.1	USE CASE DIAGRAM
5.6.2	CLASS DIAGRAM
5.6.3	ACTIVITY DIAGRAM
5.6.4	SEQUENCE DIAGRAM
5.6.5	COLLABORATION DIAGRAM
5.6.6	COMPONENT DIAGRAM

LIST OF TABLES

TABLE	TITLE	PAGE NO
4.5.1	ECLIPSE PLATFORM TECHNICAL OVERVIEW	07

Abstract:

The interface allows the user to search for an arbitrary application on the Play Store; the permissions list and the privacy policy are then automatically retrieved, whenever possible. The user has then the ability of selecting a specific permission, and a list of relevant sentences are extracted by the privacy policy and presented to them, along with an accurate description of the permission itself. Such an interface allows the user to quickly evaluate the privacyrelated risks of an Android application, by highlighting the relevant sections of the privacy policy and by providing useful information about sensible permissions. We presented a novel approach to the analysis of privacy policies in the context of Android applications. The tool we implemented greatly eases the process of understanding the privacy implications of installing third party apps and it has already been proven able to highlight worrisome instances of applications. The tool is developed with expandability in mind, and further developments in the approach can easily be integrated in order to increase the reliability and effectiveness. In addition, if your app handles personal or sensitive user data, please also refer to the additional requirements in the "Personal and Sensitive Information" section below. These Google Play requirements are in addition to any requirements prescribed by applicable privacy or data protection laws. We proposed, A user who wishes to install and use any third party app doesn't understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scene.

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Mobile devices of various operating systems have exhibited a steep increase in the past decade, thus leading to an increase in the number and variety of applications that run on mobile devices. Smartphones are recently used either in a complementary manner to a desktop computer or even to replace it. A closer look at the purpose of using mobile phones reveal that they are mostly used for web browsing, social networking, and online banking. In addition, they are used for mobile-specific functions such as SMS messaging, read-time broadcasting of location, and ubiquitous access. As the capabilities in mobile phone functionality increase (e.g., applications for personal health), mobile phones become more and more attractive for a wider population. Market data surveys reveal that the number of smartphone sales worldwide reached 208 million in 2012.

This was a 38.3% increase compared to the sales in 2011. The development of smartphones and mobile applications has resulted in a major change in people's way of doing tasks in various aspects of daily life, such as making business and conducting social communication. Mobile phone applications exhibit a rich variety not only in common daily life activities but also for users with more specific needs. From games to multimedia applications, navigation systems, and health-related applications, recently available mobile application markets, such as Google's Play Market, Apple's App Store, or Microsoft's Windows Store offer a wide variety of applications to users with different needs. The major application markets have been growing steadily both in terms of the applications offered to the users and the downloads performed by the users. The fast increase in the popularity of smartphones has led to an increase in their potential as a target for malicious activities. This is mainly because users provide access for various types of private information by means of mobile applications. As a result, some applications in the market have been identified as performing malicious activities. The spread of malicious software is also influenced by the policy of the market providers. For instance, Apple's App Store recently applies a policy that is subject to strict registration and company-issued digital certification before the release of any application, thus providing a security check for the applications listed in their application platform regularly. Others, such as Google's Play Market, introduce more freedom to developers in uploading their own software to the market. The applications are then

removed from the market in case of reported malicious activity. In particular, the Android operating system (in its recent form) allows removal of the malicious application from the device remotely. A similar mechanism of removal is also employed by Apple's App Store. The policy of postdetection removal of malicious software brings the need for early detection of malware applications.

1.2 OBJECTIVE

The main objective of this system, A user who wishes to install and use any third party app doesn't understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scene.

1.3 DOMAIN INTRODUCTION

Android is a mobile operating system developed by Google. It is based on a modified version of the Linux kernel and other open source software, and is designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The operating system has since gone through multiple major releases, with the current version being 9 "Pie", released in August 2018. Google released the first Android Q beta on all Pixel phones on March 13, 2019. The core Android source code is known as Android Open Source Project (AOSP), and is primarily licensed under the Apache License.

At Google I/O 2014, the company revealed that there were over one billion active monthly Android users, up from 538 million in June 2013. Android's source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software, including proprietary software developed and licensed by Google.

Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for

mobile devices. Android is popular with technology companies which require a ready-made, low-cost and customizable operating system for high-tech devices.

1.4 OVERVIEW OF THE PROJECT

The GP-PP (Generic Permissions-Privacy invasive permissions) model is useful to classify the permissions into Generic and Privacy invasive permissions. The model proposes a simplistic way for users to decide which apps are dangerous to install. Based on the permission set that a particular app requests, the GP-PP model classifies an app as privacy invasive if majority of the permissions requested are privacy invasive. So, users can decide which set of permissions can be harmful. Therefore, by looking at the set of permissions requested, a user can determine whether an app is privacy invasive or not. In other words, a user can determine whether it is safe to install the said app or not. Although the proposed GP-PP model seems valuable, its efficacy has not been fully assessed.

To address this gap, we test the effectiveness of the GP-PP model using Naïve Bayes Classifier in this paper. In particular, we validate the GP-PP model in order to verify whether the model classifies an app on the basis of permission sets that the app requests.

Our study confirms that greater the number of Privacy-invasive permissions that an app requests, more dangerous it is to install the app. In summary, the major contribution of the paper is the validation of the proposed GP-PP model using Naïve Bayes Classifier and decision tree algorithm for popularity.

CHAPTER 2

LITERATURE REVIEW

2.1 LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system. The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations

2.1.1 Analysis of Bayesian classification-based approaches for Android malware detection

Mobile malware has been growing in scale and complexity spurred by the unabated uptake of smartphones worldwide. Android is fast becoming the most popular mobile platform resulting in sharp increase in malware targeting the platform. Additionally, Android malware is evolving rapidly to evade detection by traditional signature-based scanning. Despite current detection measures in place, timely discovery of new malware is still a critical issue. This calls for novel approaches to mitigate the growing threat of zero-day Android malware. Hence, the authors develop and analyse proactive machine-learning approaches based on Bayesian classification aimed at uncovering unknown Android malware via static analysis. The study, which is based on a large malware sample set of majority of the existing families, demonstrates detection capabilities with high accuracy. Empirical results and comparative analysis are presented offering useful insight towards

development of effective static-analytic Bayesian classification-based solutions for detecting unknown Android malware.

2.1.2 Android permissions: a perspective combining risks and benefits

The phenomenal growth of the Android platform in the past few years has made it a lucrative target of malicious application (app) developers. There are numerous instances of malware apps that send premium rate SMS messages, track users' private data, or apps that, even if not characterized as malware, conduct questionable actions affecting the user's privacy or costing them money. In this paper, we investigate the feasibility of using both the permissions an app requests, the category of the app, and what permissions are requested by other apps in the same category to better inform users whether the risks of installing an app is commensurate with its expected benefit. Existing approaches consider only the risks of the permissions requested by an app and ignore both the benefits and what permissions are requested by other apps, thus having a limited effect. We propose several risk signals that and evaluate them using two datasets, one consists of 158,062 Android apps from the Android Market, and another consists of 121 malicious apps. We demonstrate the effectiveness of our proposal through extensive data analysis.

2.1.3 Detecting repackaged smartphone applications in third-party android marketplaces.

Recent years have witnessed incredible popularity and adoption of smartphones and mobile devices, which is accompanied by large amount and wide variety of feature-rich smartphone applications. These smartphone applications (or apps), typically organized in different application marketplaces, can be conveniently browsed by mobile users and then simply clicked to install on a variety of mobile devices. In practice, besides the official marketplaces from platform vendors (e.g., Google and Apple), a number of third-party alternative marketplaces have also been created to host thousands of apps (e.g., to meet regional or localization needs). To maintain and foster a hygienic smartphone app ecosystem, there is a need for each third-party marketplace to offer quality apps to mobile users. In this paper, we perform a systematic study on six popular Android-based third-party marketplaces. Among them, we find a common “in-the-wild” practice of repackaging legitimate apps (from the official Android Market) and distributing repackaged ones via third-party marketplaces. To better understand the extent of such practice, we implement an app similarity measurement system called DroidMOSS that applies a fuzzy hashing technique

to effectively localize and detect the changes from app-repackaging behavior. The experiments with DroidMOSS show a worrisome fact that 5% to 13% of apps hosted on these studied marketplaces are repackaged. Further manual investigation indicates that these repackaged apps are mainly used to replace existing in-app advertisements or embed new ones to “steal” or re-route ad revenues. We also identify a few cases with planted backdoors or malicious payloads among repackaged apps. The results call for the need of a rigorous vetting process for better regulation of third-party smartphone application marketplaces.

2.1.4 Is this app safe?: a large scale study on application permissions and risk signals.

Third-party applications (apps) drive the attractiveness of web and mobile application platforms. Many of these platforms adopt a decentralized control strategy, relying on explicit user consent for granting permissions that the apps request. Users have to rely primarily on community ratings as the signals to identify the potentially harmful and inappropriate apps even though community ratings typically reflect opinions about perceived functionality or performance rather than about risks. With the arrival of HTML5 web apps, such user-consent permission systems will become more widespread. We study the effectiveness of user-consent permission systems through a large scale data collection of Facebook apps, Chrome extensions and Android apps. Our analysis confirms that the current forms of community ratings used in app markets today are not reliable indicators of privacy risks of an app. We find some evidence indicating attempts to mislead or entice users into granting permissions: free applications and applications with mature content request more permissions than is typical; 'look-alike' applications which have names similar to popular applications also request more permissions than is typical. We also find that across all three platforms popular applications request more permissions than average.

2.1.5 Analyzing sensitive data transmission in android for privacy leakage detection

Android phones often carry personal information, attracting malicious developers to embed code in Android applications to steal sensitive data. With known techniques in the literature, one may easily determine if sensitive data is being transmitted out of an Android phone. However, transmission of sensitive data in itself does not necessarily indicate privacy leakage; a better indicator may be whether the transmission is by user intention or not. When transmission is not intended by the user, it is more likely a privacy leakage. The problem is how to determine if transmission is user intended. As a first solution in this space, we present a new analysis framework called AppIntent. For each data transmission, AppIntent can

efficiently provide a sequence of GUI manipulations corresponding to the sequence of events that lead to the data transmission, thus helping an analyst to determine if the data transmission is user intended or not. The basic idea is to use symbolic execution to generate the aforementioned event sequence, but straightforward symbolic execution proves to be too time-consuming to be practical. A major innovation in AppIntent is to leverage the unique Android execution model to reduce the search space without sacrificing code coverage. We also present an evaluation of AppIntent with a set of 750 malicious apps, as well as 1,000 top free apps from Google Play. The results show that AppIntent can effectively help separate the apps that truly leak user privacy from those that do not.

2.1.6 Android permissions demystified

Android provides third-party applications with an extensive API that includes access to phone hardware, settings, and user data. Access to privacy- and security-relevant parts of the API is controlled with an install-time application permission system. We study Android applications to determine whether Android developers follow least privilege with their permission requests. We built Stowaway, a tool that detects overprivilege in compiled Android applications. Stowaway determines the set of API calls that an application uses and then maps those API calls to permissions. We used automated testing tools on the Android API in order to build the permission map that is necessary for detecting overprivilege. We apply Stowaway to a set of 940 applications and find that about one-third are overprivileged. We investigate the causes of overprivilege and find evidence that developers are trying to follow least privilege but sometimes fail due to insufficient API documentation.

2.1.6 Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing

Smartphone security research has produced many useful tools to analyze the privacy-related behaviors of mobile apps. However, these automated tools cannot assess people's perceptions of whether a given action is legitimate, or how that action makes them feel with respect to privacy. For example, automated tools might detect that a blackjack game and a map app both use one's location information, but people would likely view the map's use of that data as more legitimate than the game. Our work introduces a new model for privacy, namely privacy as expectations. We report on the results of using crowdsourcing to capture users' expectations of what sensitive resources mobile apps use. We also report on a new privacy summary interface that prioritizes and highlights places where mobile apps break

people's expectations. We conclude with a discussion of implications for employing crowdsourcing as a privacy evaluation technique.

2.1.7 Android platform-based individual privacy information protection system

With the popularity of mobile phones with Android platform, Android platform-based individual privacy information protection has been paid more attention to. In consideration of individual privacy information problem after mobile phones are lost, this paper tried to use SMS for remote control of mobile phones and providing comprehensive individual information protection method for users and completed a mobile terminal system with selfprotection characteristics. This system is free from the support of the server and it can provide individual information protection for users by the most basic SMS function, which is an innovation of the system. Moreover, the protection mechanism of the redundancy process, trusted number mechanism and SIM card detection mechanism are the innovations of this system. Through functional tests and performance tests, the system could satisfy user functional and non-functional requirements, with stable operation and high task execution efficiency.

2.1.8 A framework for static detection of privacy leaks in android applications

We report on applying techniques for static information flow analysis to identify privacy leaks in Android applications. We have crafted a framework which checks with the help of a security type system whether the Dalvik bytecode implementation of an Android app conforms to a given privacy policy. We have carefully analyzed the Android API for possible sources and sinks of private data and identified exemplary privacy policies based on this. We demonstrate the applicability of our framework on two case studies showing detection of privacy leaks.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

- The GP-PP (Generic Permissions-Privacy invasive permissions) model is useful to classify the permissions into Generic and Privacy invasive permissions. The model proposes a simplistic way for users to decide which apps are dangerous to install.
- Based on the permission set that a particular app requests, the GP-PP model classifies an app as privacy invasive if majority of the permissions requested are privacy invasive. So, users can decide which set of permissions can be harmful.
- We validate the GP-PP model in order to verify whether the model classifies an app on the basis of permission sets that the app requests.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

- Security is low.

3.2 PROPOSED SYSTEM

- Identify the list of third party installed applications.
- Extract the complete list of permissions of each application.
- Identify android: protection level of each permission, i.e. Normal or Dangerous of each app
- Take android app permissions dataset. To identify the harmful applications apply classification algorithms.
- Note the accuracy of spam classification given by it and time required for execution
- Results as accuracy among different harmful and normal apps Classifiers are analysed.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- Security is high.

CHAPTER 4

SYSTEM ARCHITECTURE

4.1 SYSTEM ARCHITECTURE

In the mobile phones, the applications will be listed. There will be two process steps they are, the applications will be extracted using XML code and the applications will be decompiled using Java code. In XML after data analysing there will be many permissions will be generated and then in java after data analysing there will be API function calls. The android apps will be scanned using decision tree algorithm, After scanning process the apps will be classified into malicious and non malicious. Using decision tree algorithm if it is yes then app is malicious and if it is no then the app is not malicious. By this architecture we are representing our project model

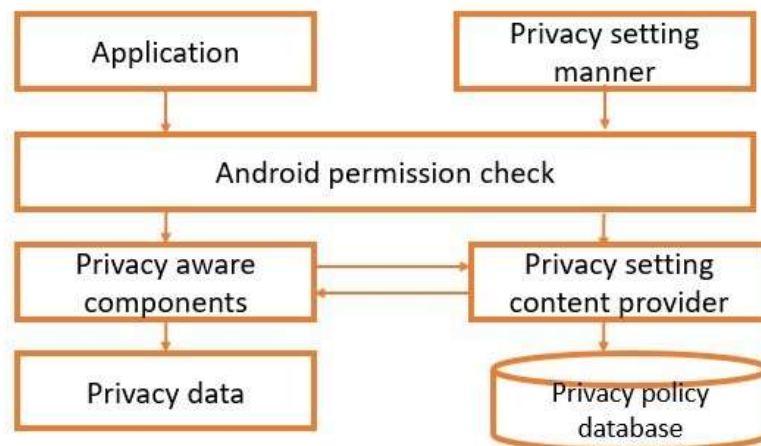


FIG 4.1.1

4.2 SYSTEM REQUIREMENTS

A Software Requirements Specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide. The requirements are contemplated and segregated into user requirements, system requirements and functional requirements. The process to gather the software requirements from client, analyse and document them is known as requirement engineering. The goal of

requirement engineering is to develop and maintain sophisticated and descriptive ‘System Requirements Specification document.

- Studying the existing or obsolete system and software
- Conducting interviews of users and developers
- Referring to the database or
- Collecting answers from the questionnaires.

4.3 HARDWARE REQUIREMENTS:

System : Pentium Dual Core.
Hard Disk : 120 GB.
Monitor : 15”LED Input
Devices : Keyboard, Mouse
Ram : 1GB.

4.4 SOFTWARE REQUIREMENTS:

Operating system : Windows 7.
Coding Language : Android, Java
Toolkit : Android 2.3 Above
IDE : Eclipse/Android Studio

4.5 MODULES

4.5.1 PENTIUM DUAL CORE

The Pentium Dual-Core brand was used for mainstream x86-architecture microprocessors from Intel from 2006 to 2009 when it was renamed to Pentium. The processors are based on either the 32-bit Yonah or (with quite different microarchitectures) 64-bit Merom-2M, Allendale, and Wolfdale-3M core, targeted at mobile or desktop computers.



4.5.2 JAVA

The Java language has undergone several changes since JDK 1.0 as well as numerous additions of classes and packages to the standard library. Since J2SE 1.4, the evolution of the Java language has been governed by the Java Community Process (JCP), which uses Java Specification Requests (JSRs) to



4.5.3 WINDOWS

Microsoft Windows is a group of several graphical operating system families, all of which are developed, marketed, and sold by Microsoft. Each family caters to a certain sector of the computing industry.

Active Windows families include Windows NT and Windows Embedded; these may encompass subfamilies, e.g. Windows Embedded Compact (Windows CE) or Windows Server. Defunct Windows families include Windows 9x, Windows Mobile and Windows Phone.



4.5.4 ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.3, which was released in January 2019.



Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google. With a user interface based on direct manipulation, Android is designed primarily for touch screen mobile devices such as smart phones and tablet computers, with specialized user interfaces for televisions (Android TV), cars (Android Auto), and wrist watches (Android Wear).

The OS uses touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, and a virtual keyboard. Despite being primarily designed for touch screen input, it has also been used in game consoles, digital cameras, regular PCs, and other electronics.

As of 2015, Android has the largest installed base of all operating systems. As of July 2013, the Google Play store has had over one million Android applications ("apps") published, and over 50 billion applications downloaded. An April–May 2013 survey of mobile application developers found that 71% of them create applications for Android; another 2015 survey found that 40% of full-time professional developers see Android as the "priority" target platform, which is more than iOS (37%) or other platforms.

At Google I/O 2014, the company revealed that there were over one billion active monthly Android users, up from 538 million in June 2013. Android's source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software, including proprietary software developed and licensed by Google.

Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Android is popular with technology companies which require a readymade, low-cost and customizable operating system for high-tech devices.

Android's open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects,

which add new features for advanced users or bring Android to devices which were officially,

released running other operating systems. The operating system's success has made it a target for patent litigation as part of the so-called "smartphone wars" between technology companies.

4.5.5 OVERVIEW OF ECLIPSE:

What is Eclipse?

In the context of computing, Eclipse is an integrated development environment (IDE) for developing applications using the Java programming language and other programming languages such as C/C++, Python, PERL, Ruby etc.

The Eclipse platform which provides the foundation for the Eclipse IDE is composed of plug-ins and is designed to be extensible using additional plug-ins. Developed using Java, the Eclipse platform can be used to develop rich client applications, integrated development environments and other tools. Eclipse can be used as an IDE for any programming language for which a plug-in is available.

The Java Development Tools (JDT) project provides a plug-in that allows Eclipse to be used as a Java IDE, PyDev is a plugin that allows Eclipse to be used as a Python IDE, C/C++ Development Tools (CDT) is a plug-in that allows Eclipse to be used for developing application using C/C++, the Eclipse Scala plug-in allows Eclipse to be used an IDE to develop Scala applications and PHPEclipse is a plug-in to eclipse that provides complete development tool for PHP.

Licensing

Eclipse platform and other plug-ins from the Eclipse foundation is released under the Eclipse Public License (EPL). EPL ensures that Eclipse is free to download and install. It also allows Eclipse to be modified and distributed.

Eclipse Releases

Every year, since 2006, the Eclipse foundation releases the Eclipse Platform and a number of other plug-ins in June.

Codename	Year	Platform Version
Callisto	2006	3.2
Europa	2007	3.3
Ganymede	2008	3.4
Galileo	2009	3.5
Helios	2010	3.6
Indigo	2011	3.7
Juno	2012	3.8 and 4.2
Kepler	2013	4.3
Luna	2014	4.4.0

FIG 4.5.1

Eclipse Platform Technical Overview

The Eclipse Platform (or simply "the Platform" when there is no risk of confusion) is designed and built to meet the following requirements:

- Support the construction of a variety of tools for application development.
- Support an unrestricted set of tool providers, including independent software vendors (ISVs).
- Support tools to manipulate arbitrary content types (e.g., HTML, Java, C, JSP, EJB, XML, and GIF).
- Facilitate seamless integration of tools within and across different content types and tool providers.
- Support both GUI and non-GUI-based application development environments.
- Run on a wide range of operating systems, including Windows®, Linux™, Mac OS X, Solaris AIX, and HP-UX.
- Capitalize on the popularity of the Java programming language for writing tools.

The Eclipse Platform's principal role is to provide tool providers with mechanisms to use, and rules to follow, that lead to seamlessly-integrated tools. These mechanisms are exposed via well-defined API interfaces, classes, and methods. The Platform also provides useful building blocks and frameworks that facilitate developing new tools.

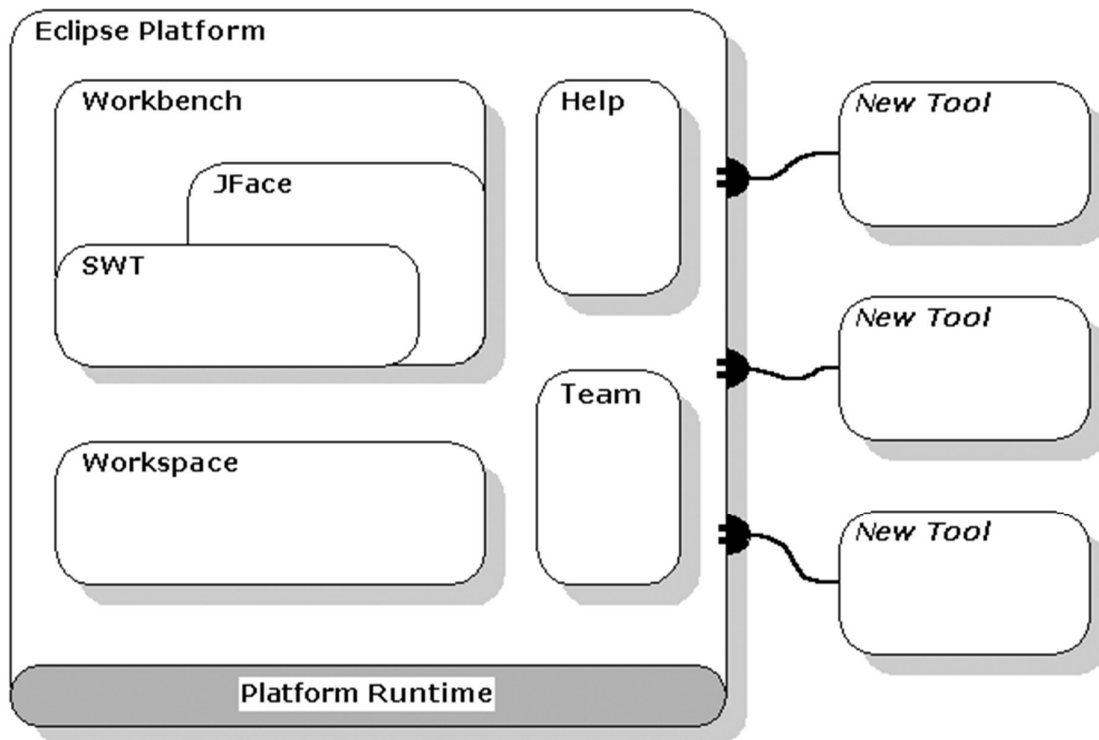


FIG 4.5.2

4.6 JAVA

Java is one of the world's most important and widely used computer languages, and it has held this distinction for many years. Unlike some other computer languages whose influence has waned with passage of time, while Java's has grown.

4.6.1 APPLICATION OF JAVA

Java is widely used in every corner of world and of human life. Java is not only used in softwares but is also widely used in designing hardware controlling software components. There are more than 930 million JRE downloads each year and 3 billion mobile phones run java.

Following are some other usage of Java:

1. Developing Desktop Applications
2. Web Applications like LinkedIn.com, Snapdeal.com etc
3. Mobile Operating System like Android
4. Embedded Systems
5. Robotics and games etc.

4.6.2 FEATURES OF JAVA

The prime reason behind creation of Java was to bring portability and security feature into a computer language. Beside these two major features, there were many other features that played an important role in moulding out the final form of this outstanding language. Those features are;

1) Simple

Java is easy to learn and its syntax is quite simple, clean and easy to understand. The confusing and ambiguous concepts of C++ are either left out in Java or they have been reimplemented in a cleaner way.

Eg: Pointers and Operator Overloading are not there in java but were an important part of C++.

2) Object Oriented

In java everything is Object which has some data and behaviour. Java can be easily extended as it is based on Object Model.

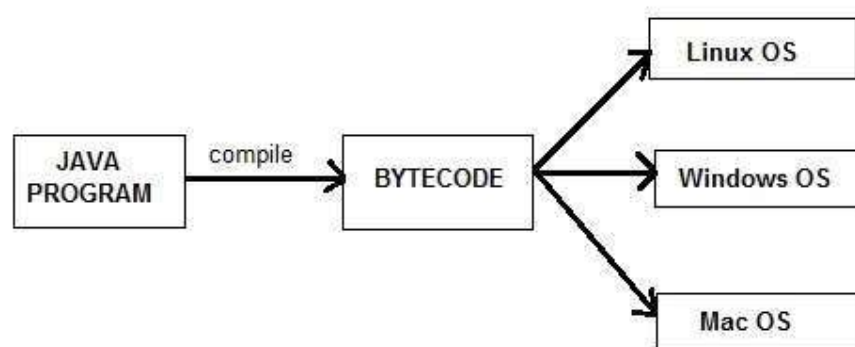
3) Robust

Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled Exceptions by introducing automatic Garbage Collector and Exception Handling.

4) Platform Independent

Unlike other programming languages such as C, C++ etc. which are compiled into platform specific machines. Java is guaranteed to be write-once, run-anywhere language.

On compilation Java program is compiled into byte code. This byte code is platform independent and can be run on any machine, plus this byte code format also provide security. Any machine with Java Runtime Environment can run Java Programs.



5) Secure

When it comes to security, Java is always the first choice. With java secure features it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.

6) Multi-Threading

Java multithreading feature makes it possible to write program that can do many tasks simultaneously. Benefit of multithreading is that it utilizes same memory and other

resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along.

7) Architectural Neutral

Compiler generates byte codes, which have nothing to do with a particular computer architecture, hence a Java program is easy to interpret on any machine.

8) Portable

Java Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types

10) High Performance

Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But, Java enables high performance with the use of just-in-time compiler.

4.7 COLLECTION FRAMEWORK

Collection framework was not part of original Java release. Collections was added to J2SE 1.2. Prior to Java 2, Java provided adhoc classes such as Dictionary, Vector, Stack and Properties to store and manipulate groups of objects. Collection framework provides many important classes and interfaces to collect and organize group of alike objects.

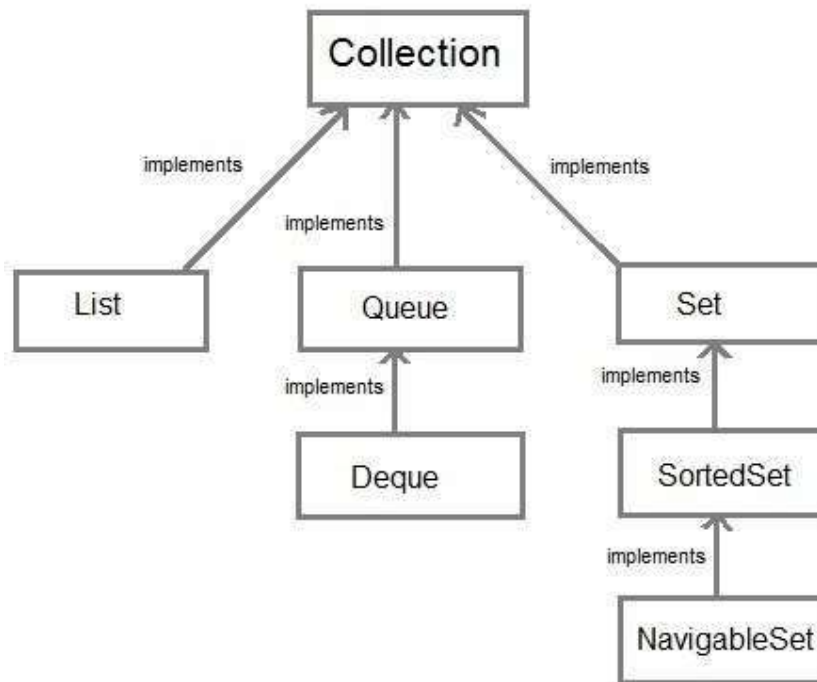


FIG 4.7.1

4.7.1 MYSQL

MySQL, officially, but also called "My Sequel" is the world's most widely used opensource relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases, though SQLite probably has more total embedded deployments. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks).

LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Freesoftwareopen source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, MODx, Joomla,

WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many highprofile, large-scale websites, including Wikipedia, Google (though not for searches), Facebook, Twitter, Flickr and YouTube.

4.7.2 REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

4.7 FUNCTIONAL REQUIREMENTS

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that

the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

Compatibility

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

Flexibility

The flexibility of the project is provided in such a way that it has the ability to run on different environments being executed by different users.

Safety

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

4.8 NON- FUNCTIONAL REQUIREMENTS

Portability

It is the usability of the same software in different environments. The project can be run in any operating system.

Performance

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

Accuracy

The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

Maintainability

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means that how easy it is to maintain the system, analyse, change and test the application.

Maintainability of this project is simple as further updates can be easily done without affecting its stability.

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the

user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

4.9 MODULES

- Login/Registration
- Android Application Scan
- Malware Detection
- Android Permission Check
- Privacy Policy Database

4.9.1 LOGIN MODULE

This is the first activity that opens when user open the website. User needs to provide a correct contact number and a password, which user enters while registering, in order to login into the website. If information provided by the user matches with the data in the database table then user successfully login into the website else message of login failed is displayed and user need to reenter correct information. A link to the register activity is also provided for registration of new users.

4.9.2 REGISTRATION MODULE

A new user who wants to access the app needs to register first before login. By clicking on register button in login activity, the register activity gets open. A new user registers by entering full name, password and contact number. A user needs to enter password again in confirm password textbox for confirmation. When user enters the information in all textboxes, on the click of register button, the data is transferred to database and user is directed to login activity again. Registered user then needs to login in order to access the app. Validations are applied on all the textboxes for proper functioning of the app. Like information in each textbox is must that is each textbox, either it is of name, contact, password or confirm password, will not be empty while registering. If any such textbox is empty app will give message of information is must in each textbox. Also data in password and confirm password fields must match for successful registration. Another validation is contact number must be valid one that is of 10 digits. If any such validation is violated then registration will be unsuccessful and then user needs to register again. Message that app will display when one of the field is empty.

If all such information is correct user will be directed to login activity for login into the app.

Application Download

In this module, many applications are already download in mobile phone through play store. After installation just find the malware present in various applications.

4.9.3 Android Application Scan

In this module, many application's are already download in mobile phone through play store. After installation just find the malware present in various application's. Malware of this kind can't be detected by victimization the quality signatures approach or by applying regular static or dynamic analysis strategies.

The detection is performed on the basis of application's network traffic patterns solely. For each application, a model representing its specific approach pattern is learned regionally (i.e., on the device).

Semi- supervised machine-learning methods are used for learning the normal behavioural patterns and for detecting deviations from the application's expected behaviour. These methods were implemented and evaluated on Android devices.

4.9.4 Malware Detection

In this proposed system in order to improve the security of the mobile apps one methodology is proposed which will evaluate the mobile applications security based on the cloud computing platform and data mining. The task of identifying malware will be categorised into analysis, classification, detection and ultimate containment of malware. Several classification techniques are utilized in order to classify malware in keeping with their instances and this has created it potential to acknowledge the sort and activities of a malware and new variant. For such kinds of threats to mobile devices there should be some security mechanism to be implemented. Analysis of malware has got do with distinctive the instances of malware by completely different classification schemes by exploitation the attributes of well-known malware characteristics. Here also a prototype system named Malware detection system is presented to identify the mobile app's virulence or benignancy. Malware detection has to do with the quick detection and validation of any instance of malware in order to prevent further damage to the system. The last a part of the work is containment of the malware, which involves effort at stopping escalation and

preventing further damages to the system. A commercial antivirus uses signature based mostly technique wherever the information should be often updated so as to possess the newest virus information detection mechanisms. However, the zero-day malicious exploit malware can not be detected by antivirus, supported signature-based scanner, however the utilization of applied math binary content analysis of file to detect abnormalous file segments.

4.9.5 Android Permission Check

A user who wishes to install and use any third party app doesn't understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scene. If an app needs to use resources or information outside of its own sandbox, the app has to request the appropriate permission. You declare that your app needs a permission by listing the permission in the app manifest and then requesting that the user approve each permission at runtime. If your app needs a dangerous permission, you must check whether you have that permission every time you perform an operation that requires that permission. The system's behavior after you declare a permission depends on how sensitive the permission is. Some permissions are considered "normal" so the system immediately grants them upon installation. Other permissions are considered "dangerous" so the user must explicitly grant your app access. For more information about the different kinds of permissions, see Protection levels. If a user keeps trying to use functionality that requires a permission, but keeps denying the permission request, that probably means the user doesn't understand why the app needs the permission to provide that functionality.

4.9.6 Privacy Policy Database

In the implementation, Android Security Evaluation Framework (ASEF) and Static Android Analysis Framework (SAAF) are adopted , the two representative dynamic and static analysis methods to evaluate the Android apps and estimate Malware detection with Accuracy. Privacy is important to us, and we are committed to protecting personal information that is processed in our database product and service. This database privacy statement explains our purposes and legal basis for what information we may collect about data subjects, sources of information and how the information is processed. We presented a novel approach to the analysis of privacy policies in the context of Android applications.

Compared with traditional method, such as permission pattern based method, Malware combines the dynamic and static analysis methods to comprehensively evaluate an Android app. The tool we implemented greatly eases the process of understanding the privacy implications of installing third party apps and it has already been proven able to highlight worrisome instances of applications. To find your apps and their permissions on mechanical man. open the Settings and so faucet Apps & notifications, App info, and also the app you are inquisitive about. Select the Permissions entry to see all the privileges the app Uninstall.

After the Malware is detected we will get notification for uninstalling the application.

CHAPTER 5

SYSTEM DESIGN AND TESTING PLAN

5.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- ☐ What data should be given as input?
- ☐ How the data should be arranged or coded?
- ☐ The dialog to guide the operating personnel in providing input.
- ☐ Methods for preparing input validations and steps to follow when error occur.

5.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

The output form of an information system should accomplish one or more of the following objectives.

- ☐ Convey information about past activities, current status or projections of the ☐ Future.
- ☐ Signal important events, opportunities, problems, or warnings.
- ☐ Trigger an action.

- Confirm an action.

5.3 SYSTEM TESTING

Test plan

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

Basics of software testing

There are two basics of software testing: black box testing and white box testing.

Black box Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

White box Testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing. Black box testing is often used for validation and white box testing is often used for verification.

5.4 Types of testing

There are many types of testing like

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

Functional Testing

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

System Testing

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

Stress Testing

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

Performance Testing

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

Usability Testing

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

Regression Testing

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

5.5 DATA FLOW DIAGRAM

Data Flow Diagram (DFD) is a two-dimensional diagram that describes how data is processed and transmitted in a system. The graphical depiction recognizes each source of data and how it interacts with other data sources to reach a mutual output. In order to draft a data flow diagram one must

- Identify external inputs and outputs
- Determine how the inputs and outputs relate to each other
- Explain with graphics how these connections relate and what they result in.

Role of DFD:

- It is a documentation support which is understood by both programmers and nonprogrammers. As DFD postulates only what processes are accomplished not how they are performed.
- A physical DFD postulates where the data flows and who processes the data.
- It permits analyst to isolate areas of interest in the organization and study them by examining the data that enter the process and viewing how they are altered when they leave.

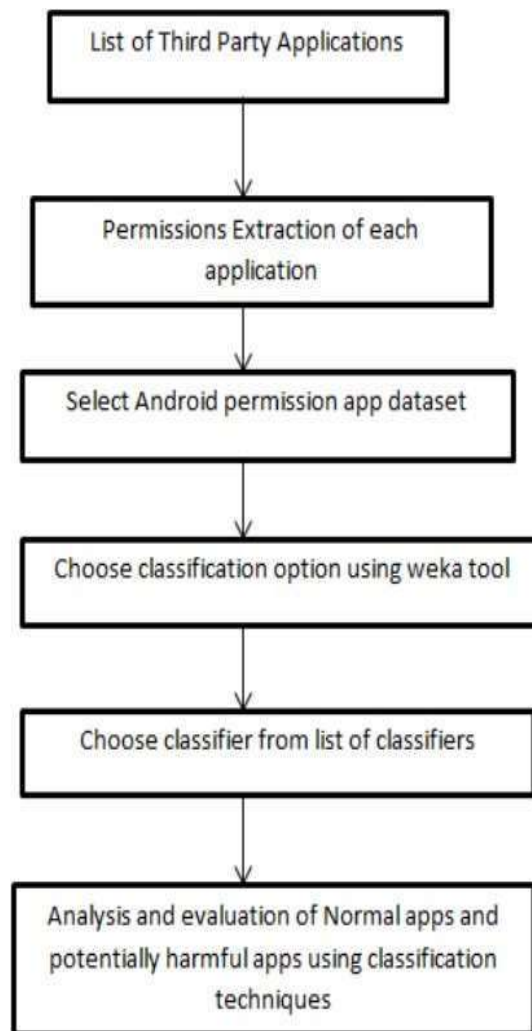


FIG 5.5.1

In the above flowchart ,the first will be list of third party application,the second step will be permission extraction from each application in the phones, the third step will be select android permission app dataset, the fourth step will be choose classification option using weka tool,the fifth step will be choose classifier from list of classifiers, the sixth step will be analysing and evaluation of normal apps and potentially harmful apps using classification techniques.

5.6 UML DIAGRAMS

UML is simply another graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

ADVANTAGES

- To represent complete systems (instead of only the software portion) using object oriented concepts
- To establish an explicit coupling between concepts and executable code □ To take into account the scaling factors that are inherent to complex and critical systems
- To creating a modeling language usable by both humans and machines

UML defines several models for representing systems

- The class model captures the static structure
- The state model expresses the dynamic behavior of objects
- The use case model describes the requirements of the user
- The interaction model represents the scenarios and messages flows
- The implementation model shows the work units
- The deployment model provides details that pertain to process allocation

5.6.1 USE CASE DIAGRAM

Use case diagrams overview the usage requirement for system. They are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe “the meant” of the actual requirements. A use case describes a sequence of action that provides something of measurable value to an action and is drawn as a horizontal ellipse.

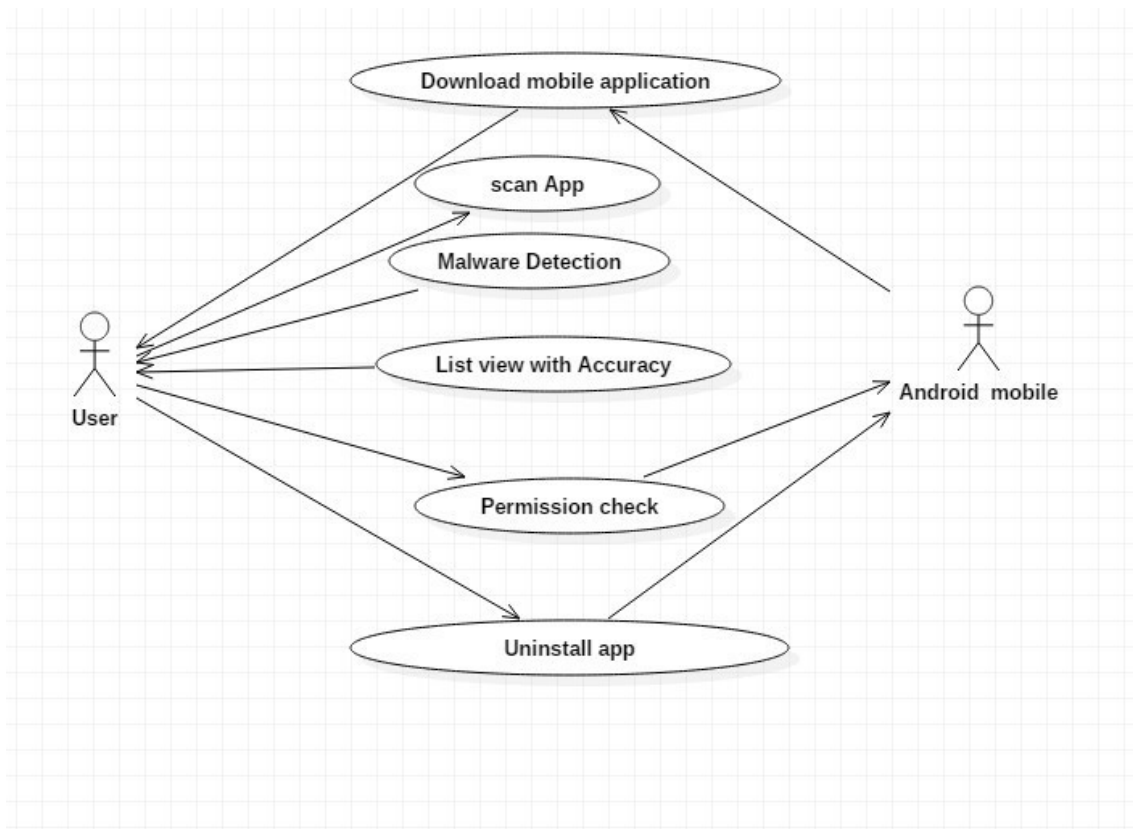


FIG 5.6.1

5.6.2 CLASS DIAGRAM

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature. Class diagram represents the object orientation of a system.

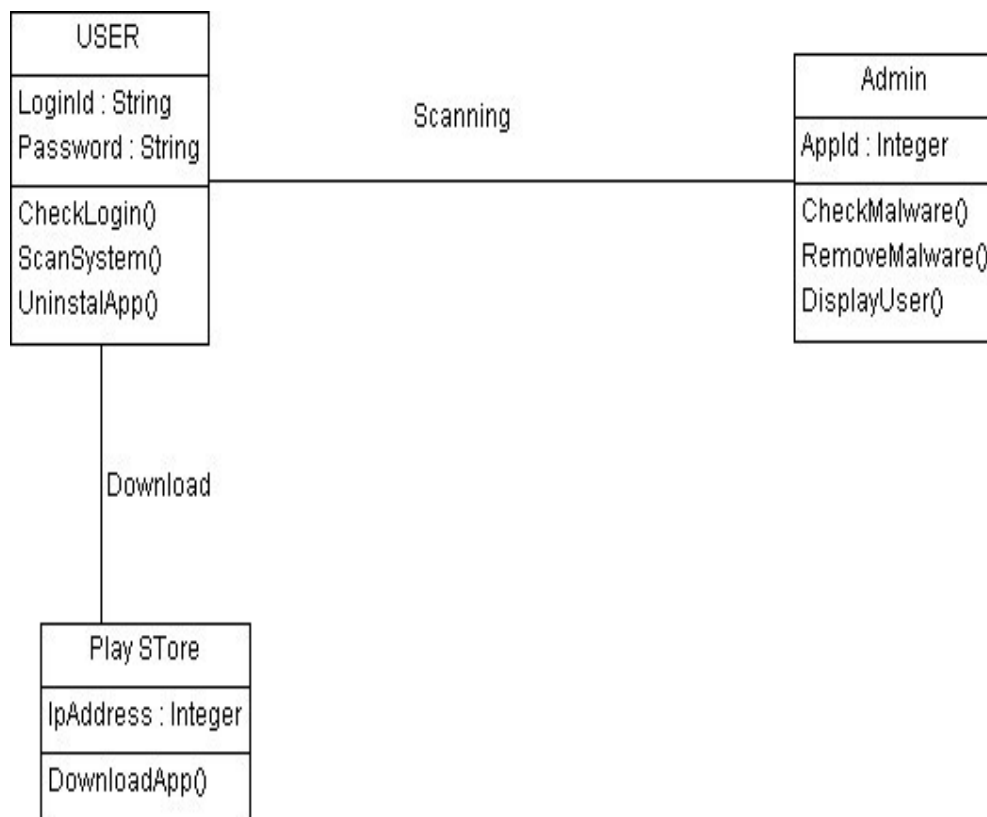


FIG 5.6.2

5.6.3 ACITIVITY DIAGRAM

Activity diagram are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. Activity diagram consist of Initial node, activity final node and activities in between.

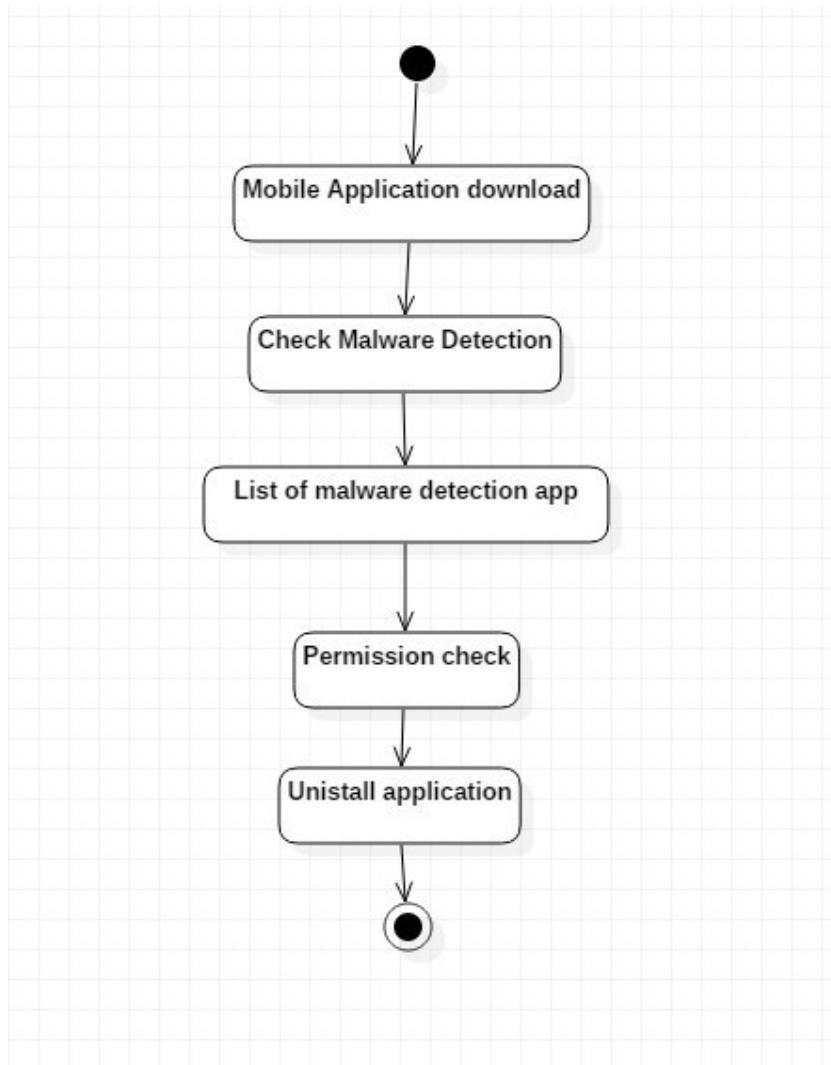


FIG 5.6.3

5.6.4 SEQUENCE DIAGRAM

Sequence diagram model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and commonly used for both analysis and design purpose. Sequence diagram are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system.

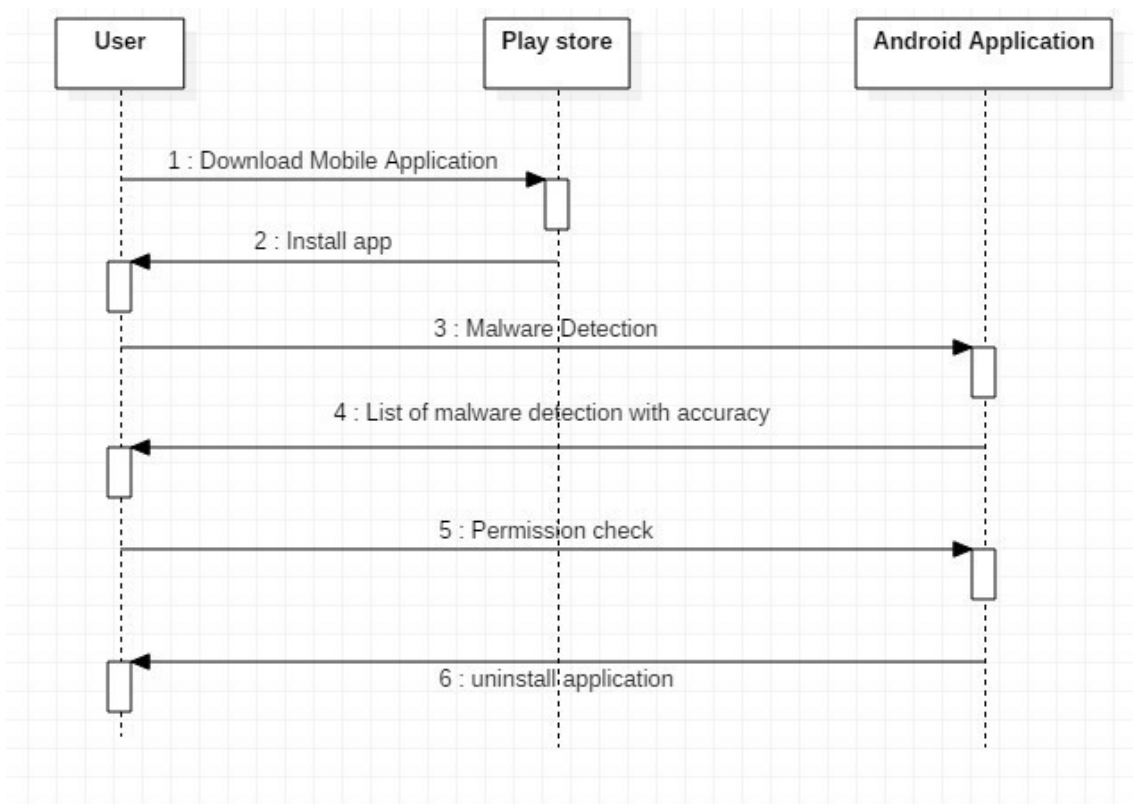


FIG 5.6.4

5.6.7 COLLABORATION DIAGRAM

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links. The purpose of collaboration diagram is similar to sequence diagram.

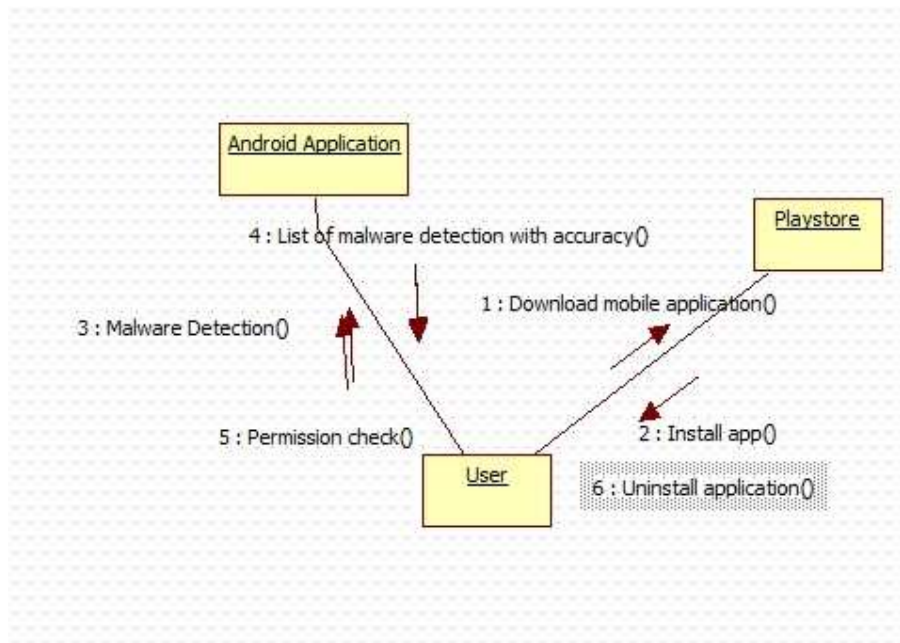


FIG 5.6.5

5.6.8 COMPONENT DIAGRAM

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system.

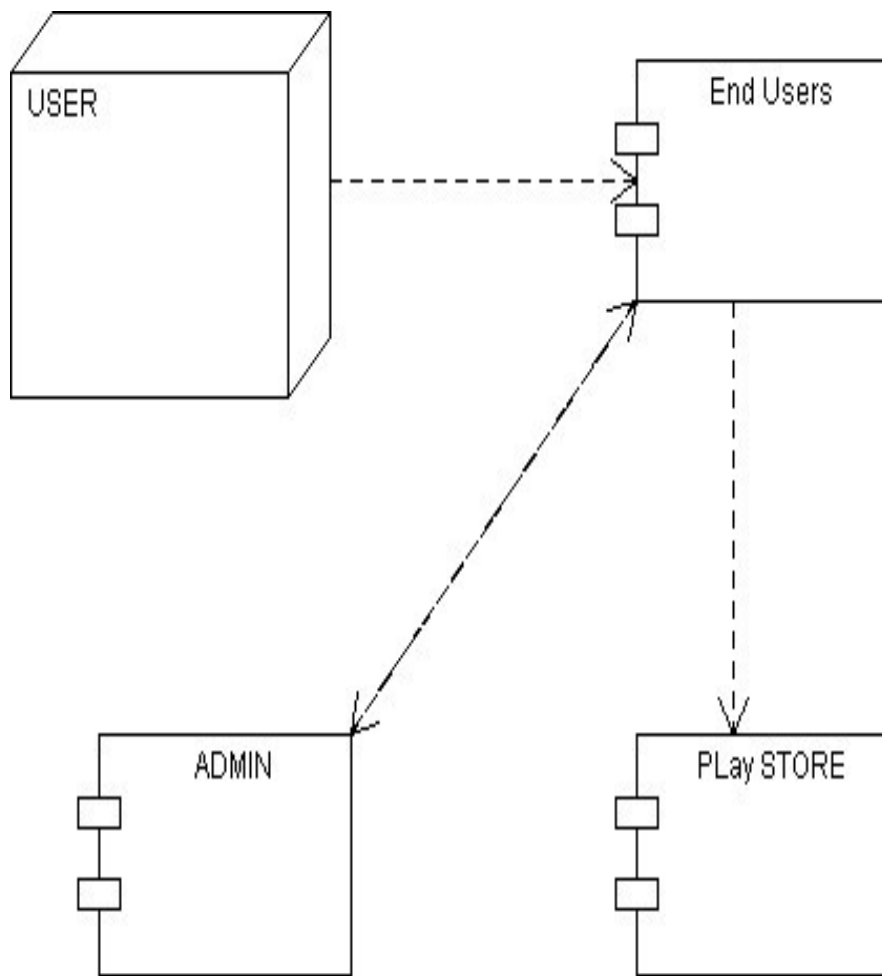


FIG 5.6.6

CHAPTER 6

6.1 APENDIX

SAMPLE CODE:

```
package com.example.androidmalwareanalyzer.ui.appsInformation;
```

```
import android.content.Context;
import
    android.content.pm.Applicatio
nInfo;                import
    android.content.pm.PackageInf
o;                    import
    android.content.pm.PackageM
anager;               import android.database.sqlite.SQLite
Database;             import
    android.graphics.PorterDuff; import
    android.graphics.drawable.Dra
wable;                import
    android.graphics.pdf.PdfDocu
ment; import android.os.Build;
import    android.os.Bundle;
import
    android.os.Environment;
import    android.os.Handler;
import    android.os.Looper;
import
    android.view.LayoutInflater;
import    android.view.View;
import
    android.view.ViewGroup;
import
    android.view.animation.Alpha
Animation;            import

    android.widget.BaseAdapter; import
    android.widget.BaseExpandabl
```

```

eListAdapter; import
android.widget.ExpandableList
Adapter; import
android.widget.ExpandableLis
t View; import
android.widget.F ilter; import
android.widget.F ilterable;
import android.widget.I
mageView; import
android.widget.L inearLayout;
import android.widget.P
rogressBar; import
android.widget.
RelativeLayout; import
android.widget.T extView;
import android.widget.Toast;

```

```

import androidx.annotation.NonNull;
import
androidx.annotation.RequiresAp
i; import androidx.appcompat.app.AppCo
mpatActivity; import androidx.core.content.res.Resour
cesCompat; import
androidx.fragment.app.Fragment
; import
androidx.fragment.app.Fragment Activity;
import androidx.fragment.app.FragmentManager;

```

```

import com.example.androidmalwareanalyzer.R; import
com.example.androidmalwareanalyzer.ui.MalwareDB; import
com.example.androidmalwareanalyzer.ui.permissionAnalyzer.AppScore;
import
com.example.androidmalwareanalyzer.ui.permissionAnalyzer.PermissionAnalysisProcess;
import org.jetbrains.annotations.NotNull;

```

```

import java.io.File; import java.io.FileInputStream;

```

```

import
java.io.FileOutputStream; import
java.io.IOException; import
java.security.MessageDigest
; import
java.security.NoSuchAlgori
t hmException; import
java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import
java.util.concurrent.Ex
ecutorService;
import java.util.concurrent.Executors;

```

```

public class AppDetails
extends Fragment { private
PackageInfo app; private
RelativeLayout perm_layout; private
AppScore as; private
MalwareDB dbHelper; private
    HashMap<String, List<String>>
    domain; private
    ExpandableListView domain_list;
private
DomainAdapter adapter;

```

```

    public AppDetails(PackageInfo app){
this.app = app;

    }

```

```

@RequiresApi(api = Build.VERSION_CODES.O)

    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) { super.onCreate(savedInstanceState);

```

```

View v = inflater.inflate(R.layout.view_app,container,false);

dbHelper = new MalwareDB(getContext());

String sha = getHash(app.applicationInfo, "SHA256");
String md5 = getHash(app.applicationInfo, "MD5");

ImageView iv = v.findViewById(R.id.appimg);

TextView name = v.findViewById(R.id.appname);

TextView version = v.findViewById(R.id.app_version);

perm_layout = v.findViewById(R.id.perm_layout);

TextView category = v.findViewById(R.id.app_category);

TextView pack = v.findViewById(R.id.app_package);

TextView dir = v.findViewById(R.id.app_dir);

TextView hash_sha = v.findViewById(R.id.app_sha256);

TextView hash_md5 = v.findViewById(R.id.app_md5);

name.setText(app.applicationInfo.loadLabel(getActivity().getPackageManager()));
version.setText("Versión " + app.versionName);

String cat = (String) app.applicationInfo.getCategoryTitle(getContext(),
app.applicationInfo.category); if (cat == null){
cat = "Undefined";
}

category.setText(cat);

pack.setText(app.packageName); dir.setText(app.applicationInfo.sourceDir);
hash_sha.setText(sha);
hash_md5.setText(md5);

```

```
iv.setImageDrawable(app.applicationInfo.loadIcon(getActivity().getPackageManager()));
```

```
domain_list = v.findViewById(R.id.domain_list);
```

```
ExecutorService executorService = Executors.newSingleThreadExecutor();
```

```
Handler handler = new Handler(Looper.getMainLooper());
```

```
executorService.execute(new Runnable() {
```

```
    @RequiresApi(api = Build.VERSION_CODES.O)
```

```
    @
```

```
    O
```

```
    ve
```

```
    rri
```

```
    de
```

```
    pu
```

```
    bli
```

```
    c
```

```
    vo
```

```
    id ru
```

```
    n()
```

```
    {
```

```
        as = dbHelper.getAppScore(new AppScore(app), app.requestedPermissions);
```

```
executorService.shutdown();
```

```
    }
```

```
});
```

```
handler.post(new Runnable() {
```

```
    @RequiresApi(api = Build.VERSION_CODES.O)
```

```
    @Override
```

```
public void run() {
```

```
    List<String> keys = new ArrayList<>();
```

```

while(!executorService.isTerminated()){ }

//Score

    TextView    quality =
v.findViewById(R.id.quality);    TextView
quantity    =    v.findViewById(R.id.quantity);
quality.setText(as.getQuality_score()+"/10");
quantity.setText(as.getQuantity_score()+"/10");
//Domain

    domain = as.getDomain();

    for (String key : domain.keySet()){ keys.add(key); }
adapter = new DomainAdapter(getActivity(),    domain,
    keys);    domain_list.setAdapter(adapter);

    setListViewHeight(domain_list, domain.size());

    domain_list.setOnGroupClickListener((parent, v12, groupPosition,
id) -> {    setListViewHeight(parent, groupPosition);
return false;
    });

    domain_list.setOnChildClickListener((parent, v1,    groupPosition,
childPosition, id) -> {    if (childPosition != 0){

        PermissionFragment fragment = new PermissionFragment(as, dbHelper,
(String) adapter.getChild(groupPosition, childPosition));

        FragmentManager    manager    =
        getParentFragmentManager();
manager.beginTransaction().replace(R.id.nav_host_fragment,    fragment,
fragment.getTag()).addToBackStack(null).commit();
    }

return true;

    });

```

```

        perm_layout.setOnClickListener(v1 -> {

            AppPermissions fragment = new AppPermissions(as, dbHelper);
            FragmentManager manager = getParentFragmentManager();
            manager.beginTransaction().replace(R.id.nav_host_fragment, fragment,
            fragment.getTag()).addToBackStack(null).commit();

        });

    }

});

return v;
}

```

```

public String getHash(ApplicationInfo app, String alg) {

    //Create checksum for this
    file
    File file = new File(app.sourceDir);

    //Use alg algorithm
    MessageDigest md = null;

    try {
        md =
        MessageDigest.getInstance
        ce(alg);    } catch
        (NoSuchAlgorithmException
        ion e) {
        e.printStackTrace();
    }

    //Get the checksum
    String checksum = null;

```



```

        try {
            checksum =
getFileChecksum(md, file);
        } catch (IOException e) {
e.printStackTrace();
        }

        return checksum;
    }

```

```

private static String getFileChecksum(MessageDigest digest, File file) throws
IOException { //FileInputStream para leer

```

```

    FileInputStream fis = new FileInputStream(file);

```

```

        //Create byte array to read
data in chunks    byte[]
byteArray = new byte[1024];
        int bytesCount = 0;

```

```

        //Leer archivo y actualizarlo en message
digest    while ((bytesCount =
fis.read(byteArray))
!= -1) {
digest.update(byteArray, 0,
bytesCount);
        };

```

```

        //Cerrar el stream
fis.close();

```

```

        //Obtener el hash    byte[]
bytes = digest.digest();

```

```

        //Convertir el hash a hex

```

```

        StringBuilder sb = new
StringBuilder();    for(int i=0;
i< bytes.length ;i++)

    {

        sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).substring(1));

    }

    //Devuelve el hash
    return sb.toString();

}

```

```

public class DomainAdapter extends BaseExpandableListAdapter{

    private HashMap<String, List<String>> domain;

private List<String> keys;    private Context
context;

    public DomainAdapter(Context context, HashMap<String, List<String>> domain,
List<String> keys) {        this.context = context;        this.domain = domain;
this.keys = keys;
    }

    @Override    public int getGroupCount() {

return keys.size(); }

    @Override

    public int getChildrenCount(int groupPosition) { return
domain.get(getGroup(groupPosition)).size(); }

```

```
        @Override        public Object getGroup(int groupPosition) { return  
keys.get(groupPosition); }
```

```
        @Override  
  
        public Object getChild(int groupPosition, int childPosition) { return  
domain.get(getGroup(groupPosition)).get(childPosition); }
```

```
        @Override        public long getGroupId(int groupPosition) { return  
groupPosition; }
```

```
        @Override        public long getChildId(int groupPosition, int childPosition) {  
return childPosition; }
```

```
        @Override        public boolean hasStableIds()  
{ return false; }
```

```
        @Override  
  
        public View getGroupView(int groupPosition, boolean isExpanded, View convertView,  
ViewGroup parent) {
```

```
            if(convertView == null){  
  
                LayoutInflater inflater = (LayoutInflater)  
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
  
                convertView = inflater.inflate(R.layout.view_domains, null);  
  
            }
```

```
            TextView tvGroup = (TextView) convertView.findViewById(R.id.tvGroup);  
  
            String group = (String) getGroup(groupPosition);  
tvGroup.setText(dbHelper.getDomainName(group));
```

```
        ImageView imgGroup = (ImageView)convertView.findViewById(R.id.imgGroup);
imgGroup.setImageDrawable(selectDrawable(group));
imgGroup.setColorFilter(R.color.light_grey, PorterDuff.Mode.MULTIPLY);
```

```
        ImageView arrow = (ImageView)convertView.findViewById(R.id.arrow);
```

```
        if (isExpanded) {
arrow.setImageResource(R.drawable.ic_baseline_expand_less_24);
        } else {
        arrow.setImageResource(R.drawable.ic_baseline_expand_more_24);
        }
    }
```

```
        return convertView;
```

```
    }
```

```
@Override
```

```
    public View getChildView(int groupPosition, int childPosition, boolean isLastChild, View
convertView, ViewGroup parent) {
```

```
        if(convertView == null) {
```

```
            LayoutInflater inflater = (LayoutInflater)
```

```
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
```

```
            convertView = inflater.inflate(R.layout.view_domains_item, null);
```

```
        }
```

```
        TextView tvItem = (TextView) convertView.findViewById(R.id.tvItem);
tvItem.setText(getChild(groupPosition, childPosition)+"" );
```

```

        return convertView;
    }

    @Override    public boolean isChildSelectable(int groupPosition, int
childPosition) { return true; }

    }

    public void setListViewHeight(ExpandableListView listView, int group) {

        ExpandableListAdapter    listAdapter    =    (ExpandableListAdapter)
listView.getExpandableListAdapter();    int totalHeight = 0;

        int    desiredWidth    =

            View.MeasureSpec.makeMeasureSpec(listView.getWidth(),
View.MeasureSpec.EXACTLY);

        for (int i = 0; i < listAdapter.getGroupCount(); i++) {

            View groupItem = listAdapter.getGroupView(i, false, null, listView);
groupItem.measure(desiredWidth, View.MeasureSpec.UNSPECIFIED);

            totalHeight += groupItem.getMeasuredHeight();

            if (((listView.isGroupExpanded(i)) && (i != group))

                || ((!listView.isGroupExpanded(i)) &&
(i == group))) {                for (int j = 0; j <
listAdapter.getChildrenCount(i);                j++)    {
View listItem = listAdapter.getChildView(i, j, false,
null,                listView);
listItem.measure(desiredWidth, View.MeasureSpec.UNSPECIFIED);

                totalHeight += listItem.getMeasuredHeight();

            }

```

```

        //Add Divider Height        totalHeight += listView.getDividerHeight() *
(listAdapter.getChildrenCount(i) - 1);

    }

}

```

```

        //Add Divider Height        totalHeight += listView.getDividerHeight() *
(listAdapter.getGroupCount() - 1);        ViewGroup.LayoutParams params =
listView.getLayoutParams();        int height = totalHeight

        +        (listView.getDividerHeight()        *
(listAdapter.getGroupCount() - 1));        if (height < 10)
height = 200;        params.height = height;
listView.setLayoutParams(params);
        listView.requestLayout();

    }

```

```

private Drawable selectDrawable(String group){

```

```

    Drawab

```

```

    le d =

```

```

    null;

```

```

    switch

```

```

    (group)

```

```

    {

```

```

        case        "android.permission-group.STORAGE":        d        =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_folder_24, null);
break;        case "android.permission-group.APP_INFO": d =
ResourcesCompat.getDrawable(getResources(),
R.drawable.ic_baseline_perm_device_information_24, null); break;        case
"android.permission-group.LOCATION":        d        =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_location_on_24,
null); break;

        case "android.permission-group.SYSTEM_TOOLS": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_app_settings_alt_24,
null); break;        case "android.permission-group.NETWORK": d =

```

```

ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_network_check_24,
null); break;                case "android.permission-group.ACCOUNTS": d =
ResourcesCompat.getDrawable(getResources(),
R.drawable.ic_baseline_manage_accounts_24, null); break;        case
"android.permission-group.PERSONAL_INFO": d = ResourcesCompat.getDrawable(getResources(),
R.drawable.ic_baseline_person_search_24, null); break;        case "android.permission-
group.VOICEMAIL": d = ResourcesCompat.getDrawable(getResources(),
R.drawable.ic_baseline_phone_in_talk_24, null); break;        case "android.permission-
group.BLUETOOTH_NETWORK": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_bluetooth_24, null);
break;        case "android.permission-group.MESSAGES": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_chat_24, null); break;
case "android.permission-group.PHONE_CALLS": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_local_phone_24,
null); break;                case "android.permission-group.CAMERA": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_camera_alt_24,
null); break;        case "android.permission-group.DEVELOPMENT_TOOLS": d =
ResourcesCompat.getDrawable(getResources(),
R.drawable.ic_baseline_developer_mode_24, null); break;        case "android.permission-
group.AFFECTS_BATTERY": d = ResourcesCompat.getDrawable(getResources(),
R.drawable.ic_baseline_battery_alert_24, null); break;        case "android.permission-
group.SCREENLOCK": d =
ResourcesCompat.getDrawable(getResources(),
R.drawable.ic_baseline_screen_lock_portrait_24, null); break;

case "android.permission-group.STATUS_BAR": d =

ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_toc_24,
null); break;        case "android.permission-group.AUDIO_SETTINGS": d =

ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_volume_up_24, null);
break;        case "android.permission-group.SOCIAL_INFO": d =

ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_people_alt_24, null);
break;        case "android.permission-group.SYNC_SETTINGS": d =

ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_sync_24, null);
break;        case "android.permission-group.MICROPHONE": d =

ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_mic_24, null);
break;        case "android.permission-group.DEVICE_ALARMS": d =

```

```

ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_access_alarm_24,
null); break;      case "android.permission-group.WALLPAPER": d =
ResourcesCompat.getDrawable(getResources(),    R.drawable.ic_baseline_wallpaper_24,
null); break;      case "android.permission-group.DISPLAY": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_view_quilt_24,
null); break;      case "android.permission-group.USER_DICTIONARY": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_menu_book_24,
null); break;      case "android.permission-group.WRITE_USER_DICTIONARY": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_menu_book_24,
null); break;      case "android.permission-group.BOOKMARKS": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_bookmarks_24,
null); break;      case "android.permission-group.HARDWARE_CONTROLS": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_hardware_24,
null); break;      case "android.permission-group.SYSTEM_CLOCK": d =
ResourcesCompat.getDrawable(getResources(), R.drawable.ic_baseline_access_time_24, null);
break;

    }

return d;

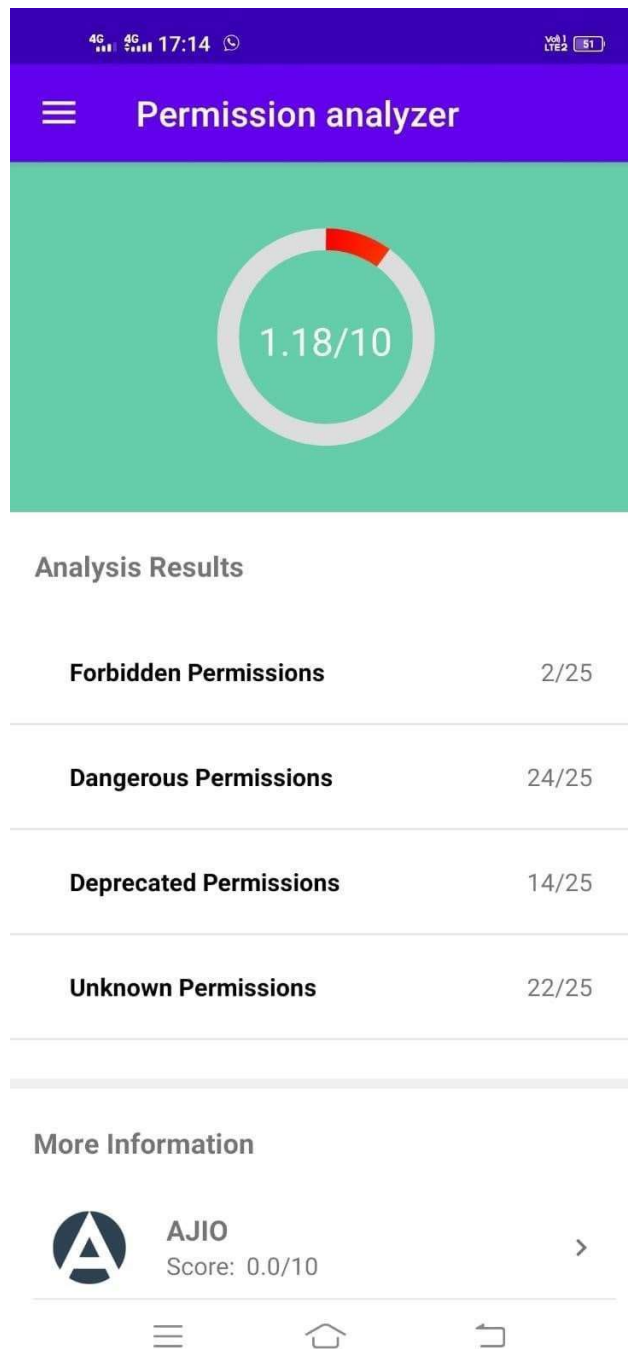
}
}

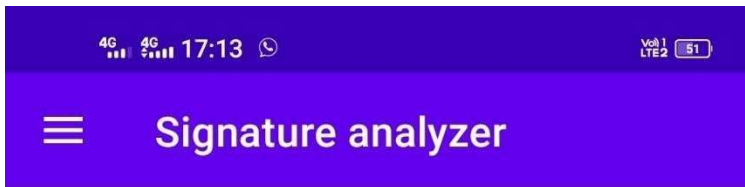
```

6.2 OUTPUT:

7.2. RESULT ANALYSIS:

7.3. SAMPLE OUTPUT:





Signature analysis result

2023-01-10 17:13:31

Analyzed elements:

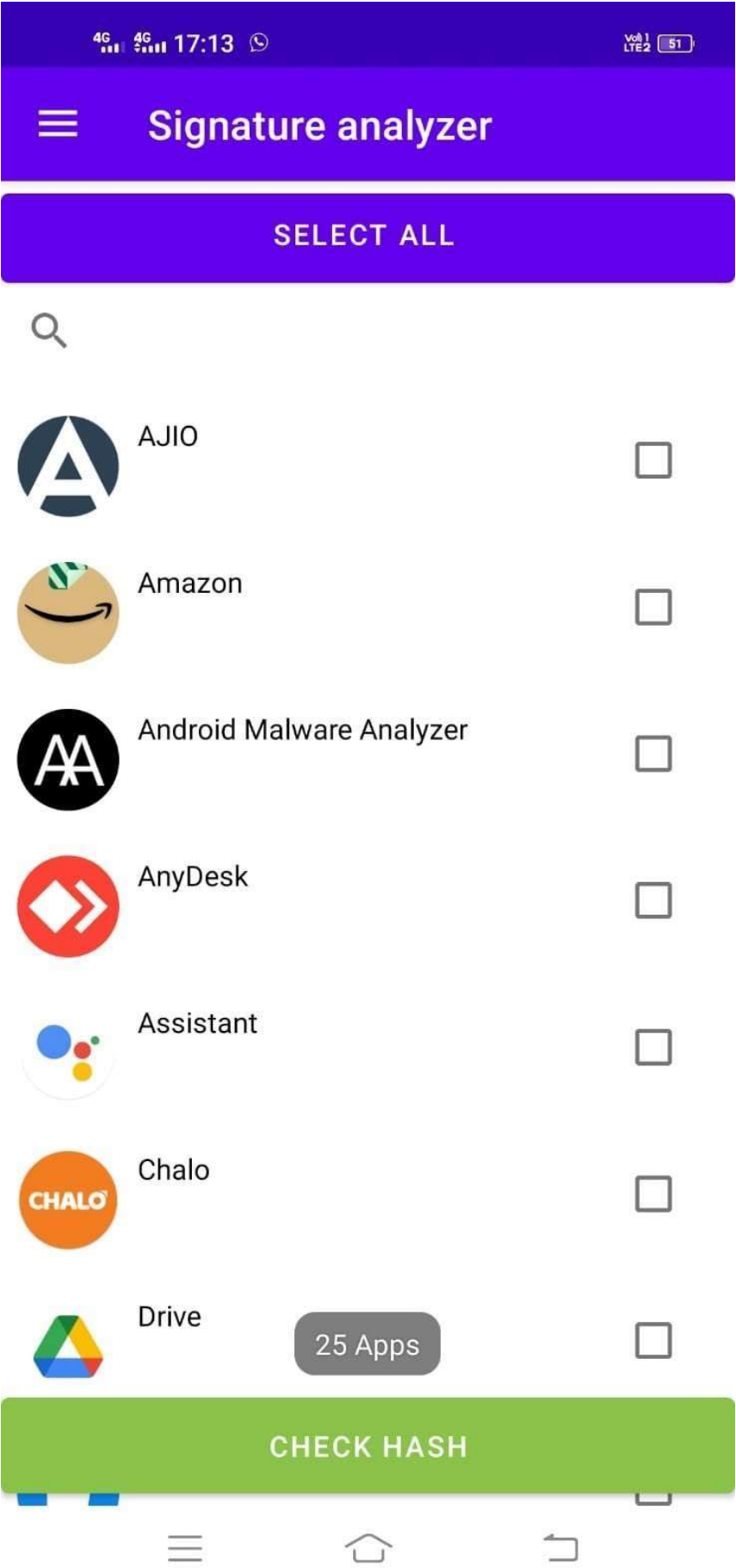


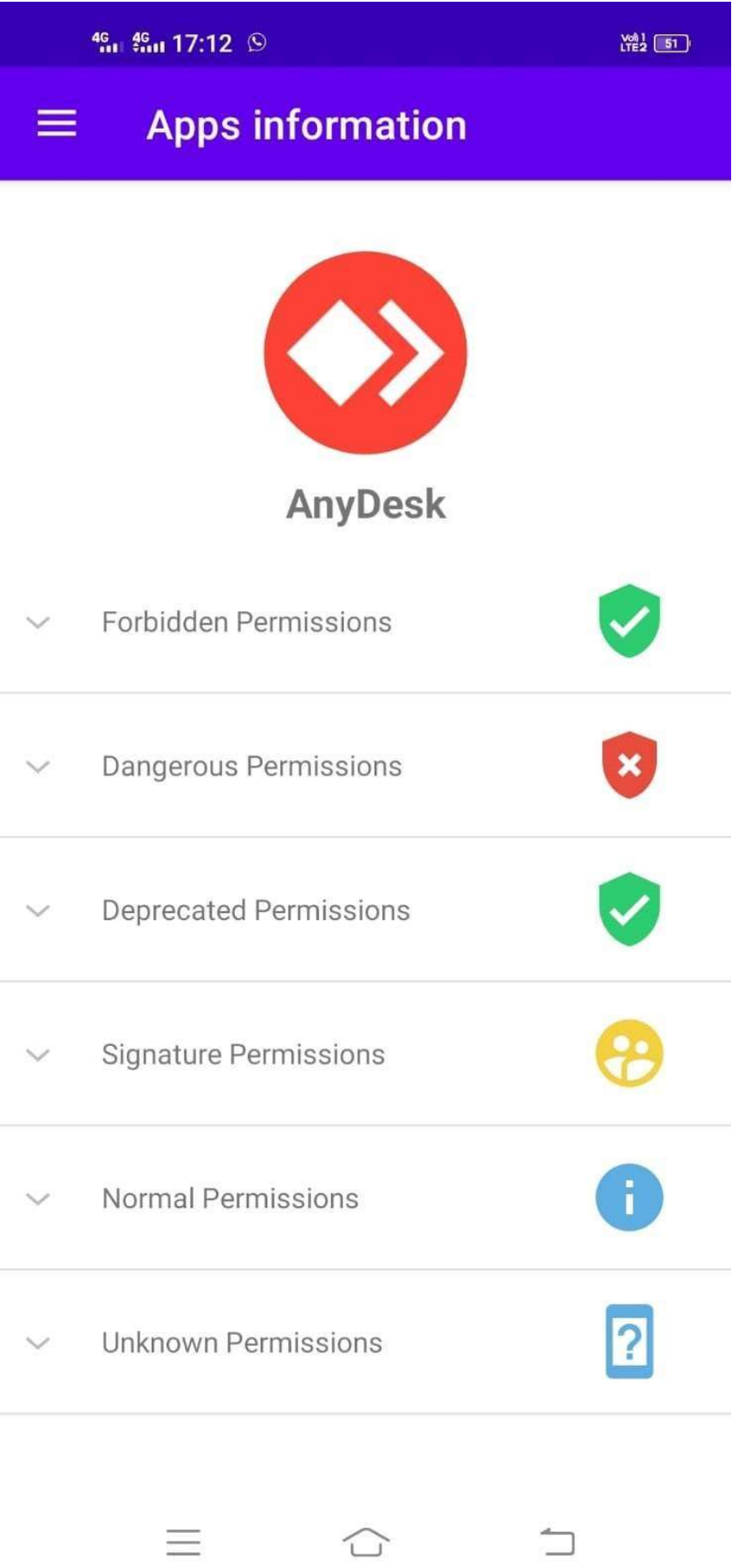
Everything

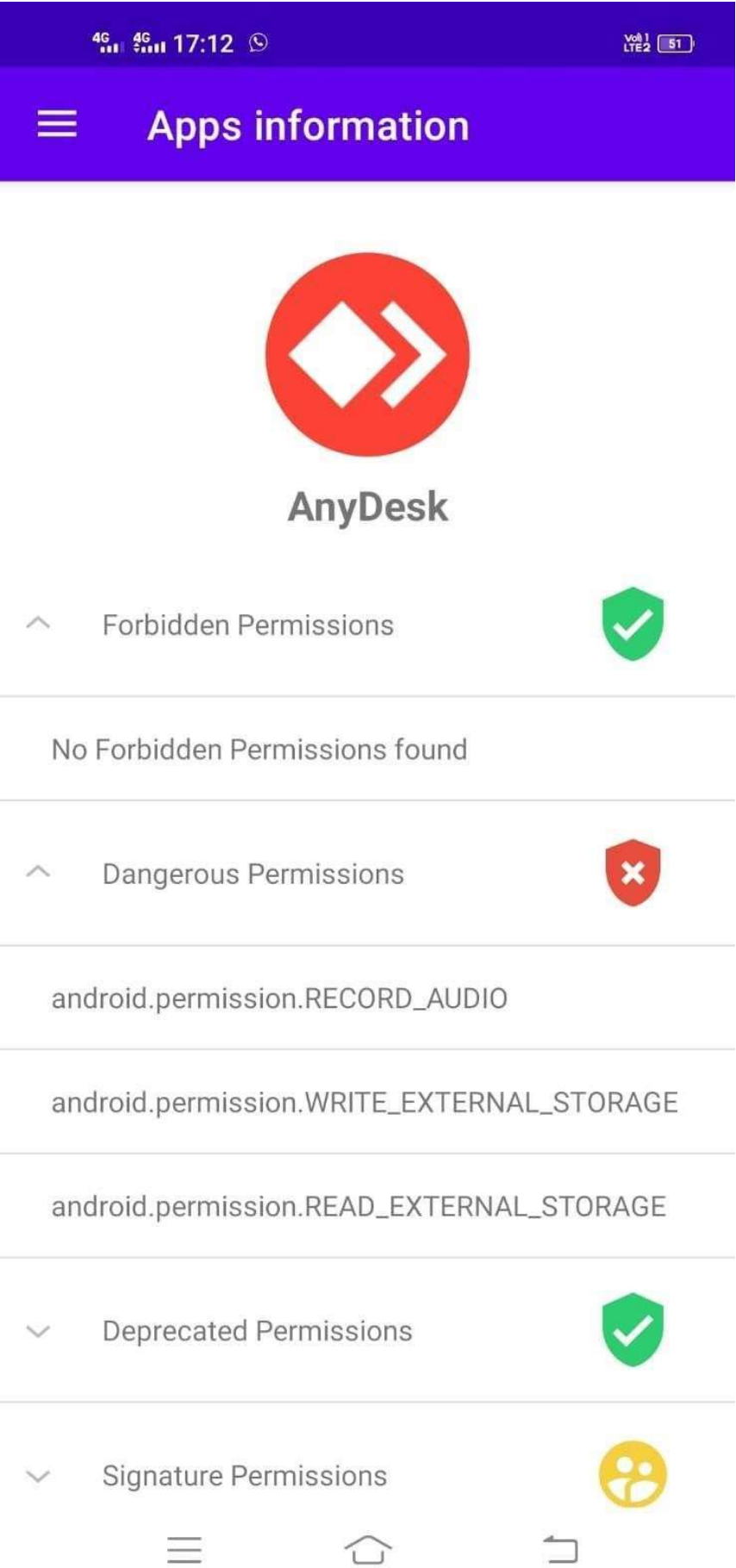
Analysis result:

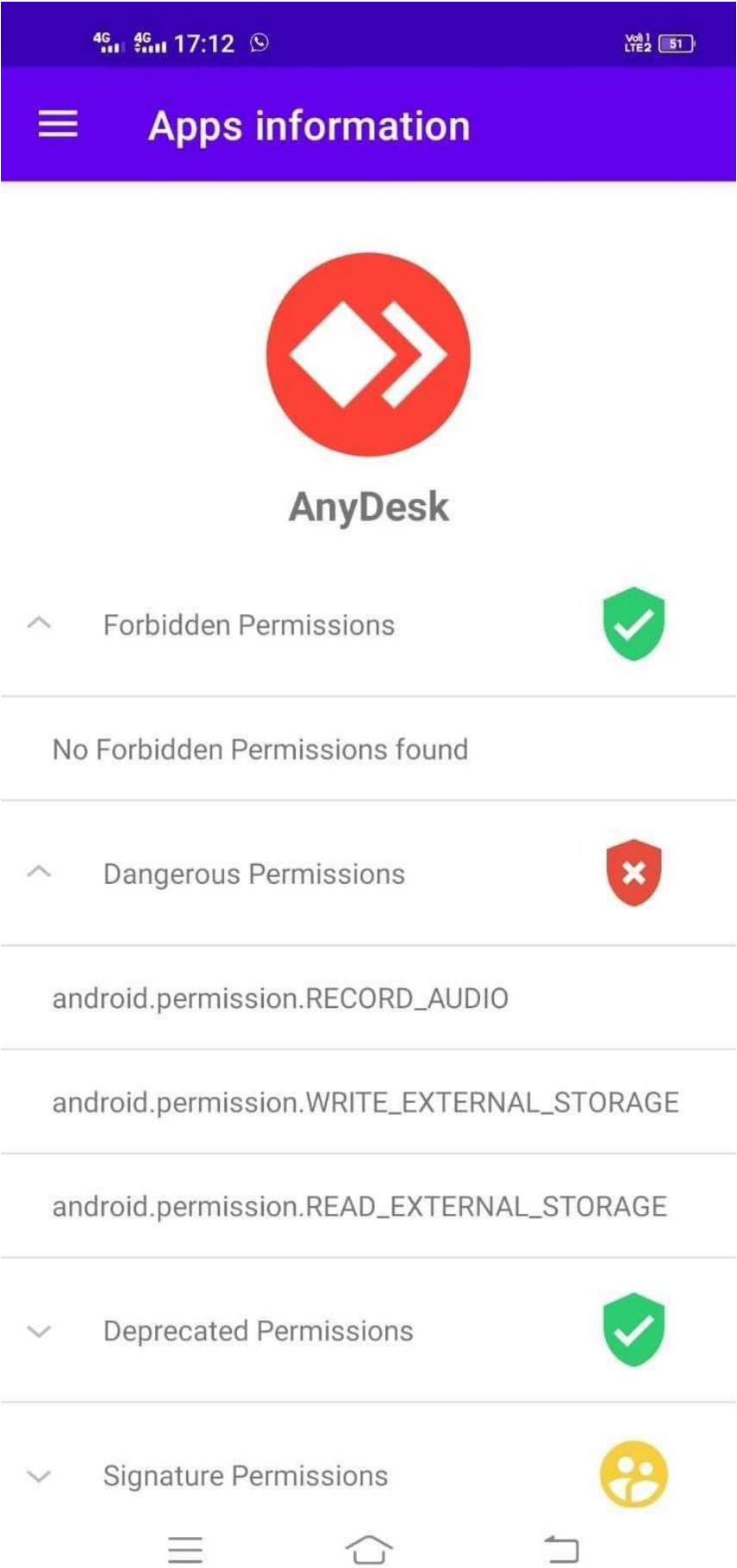
There is no malware found

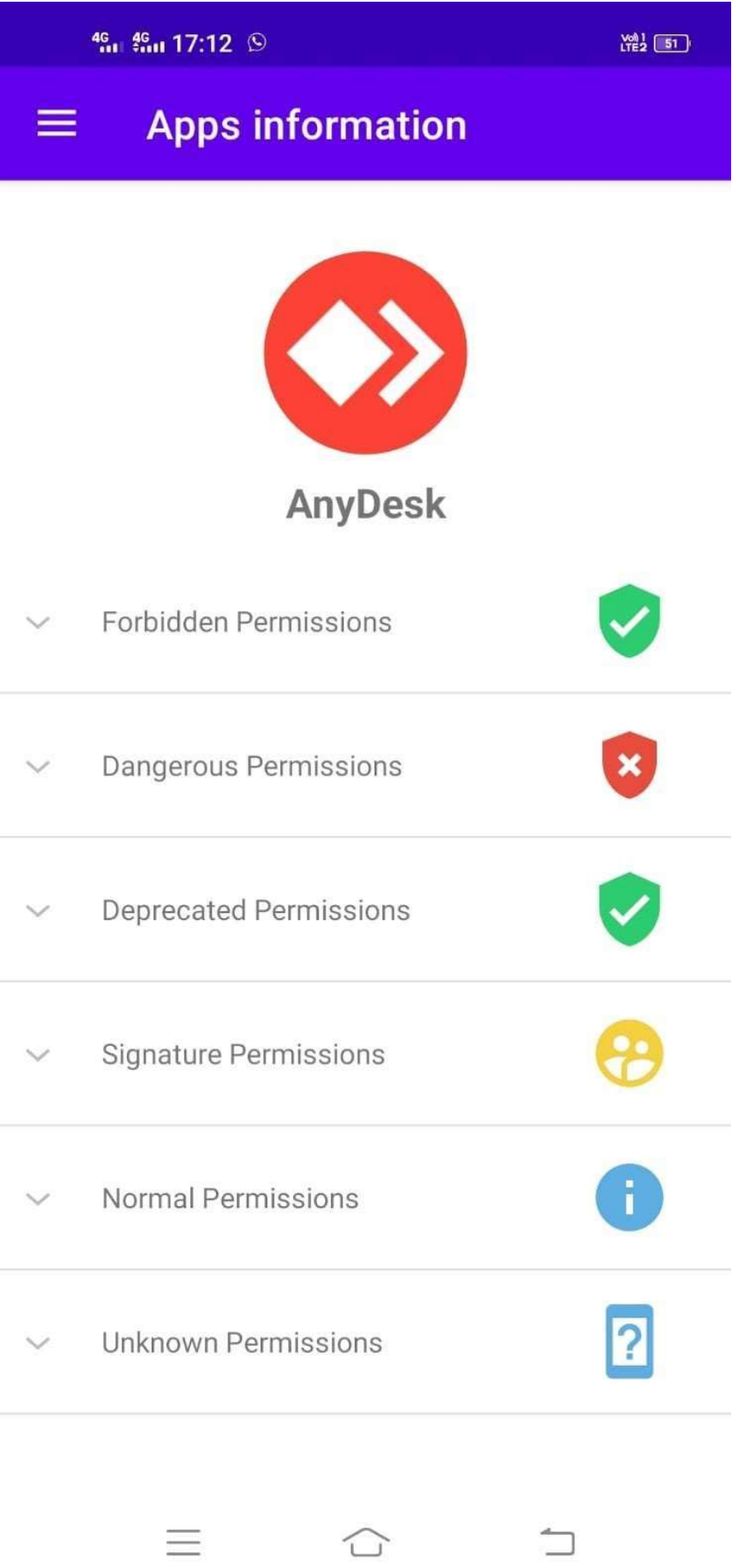


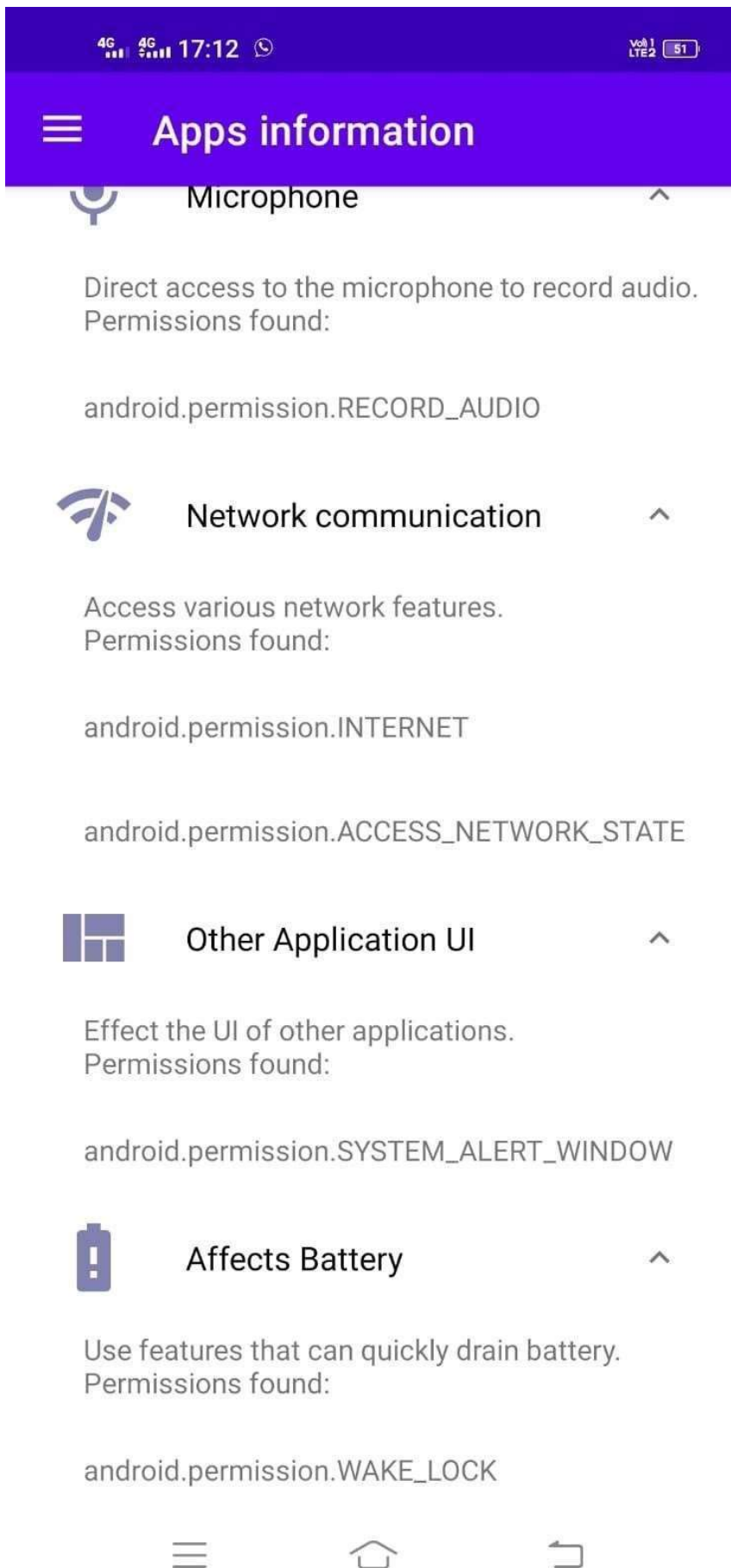


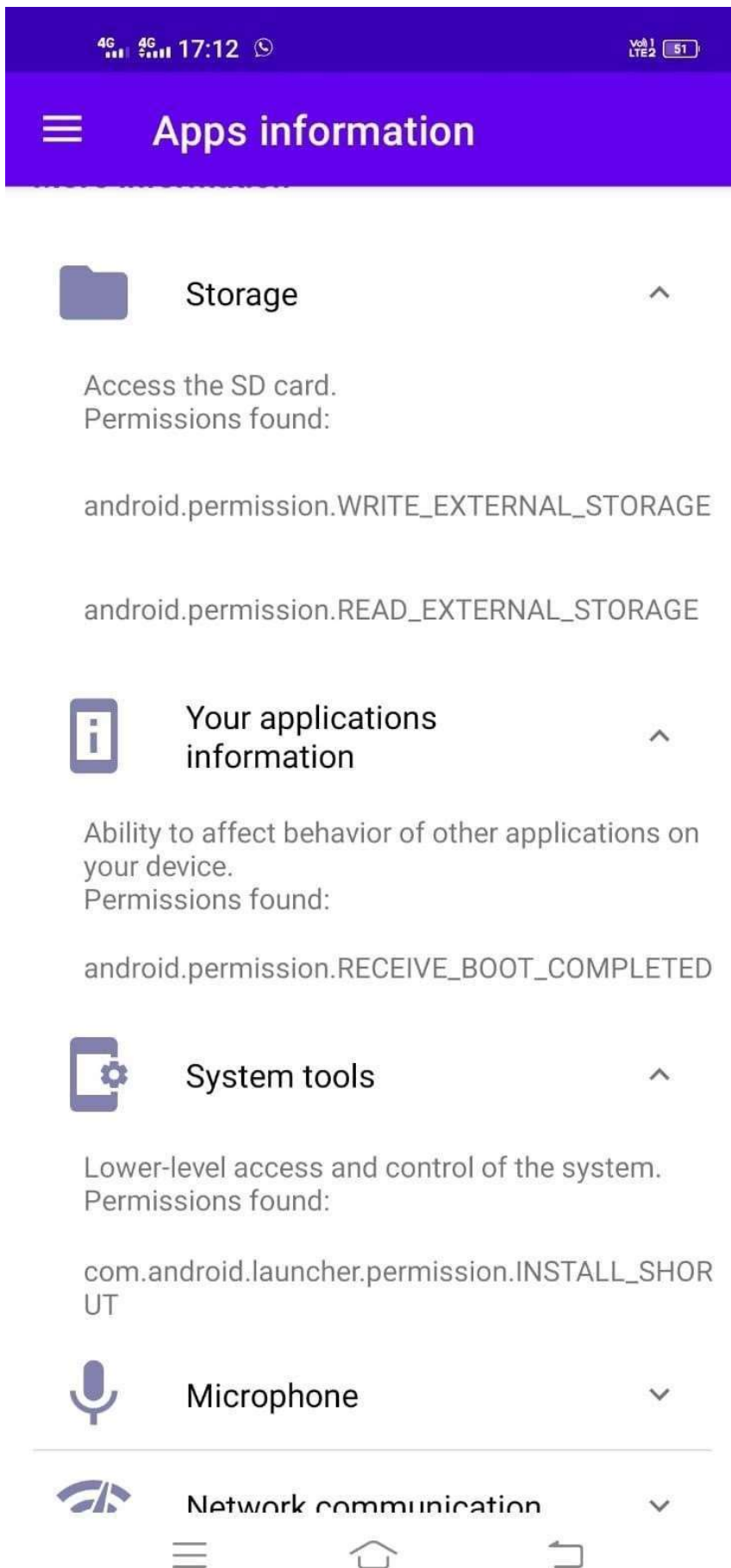


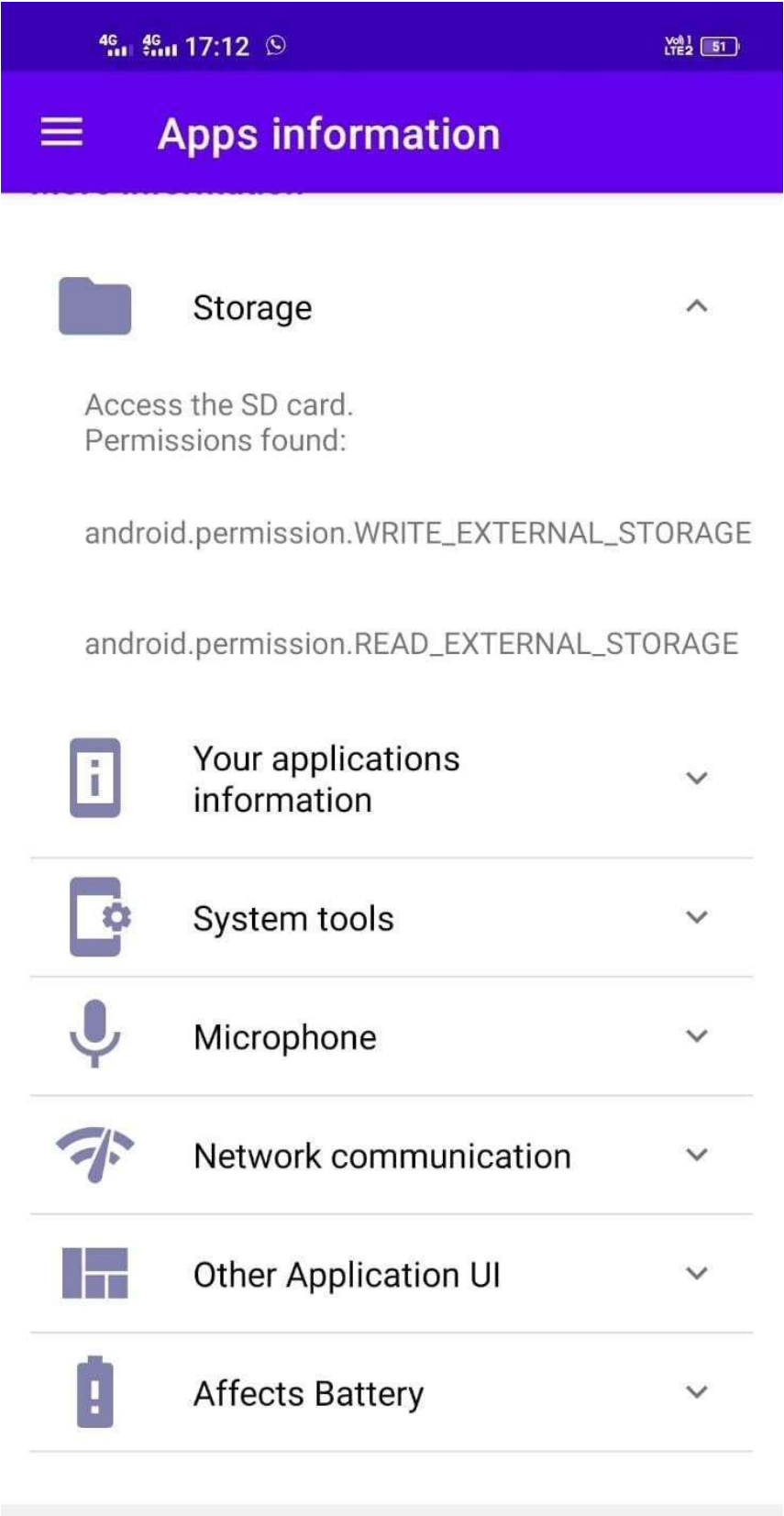






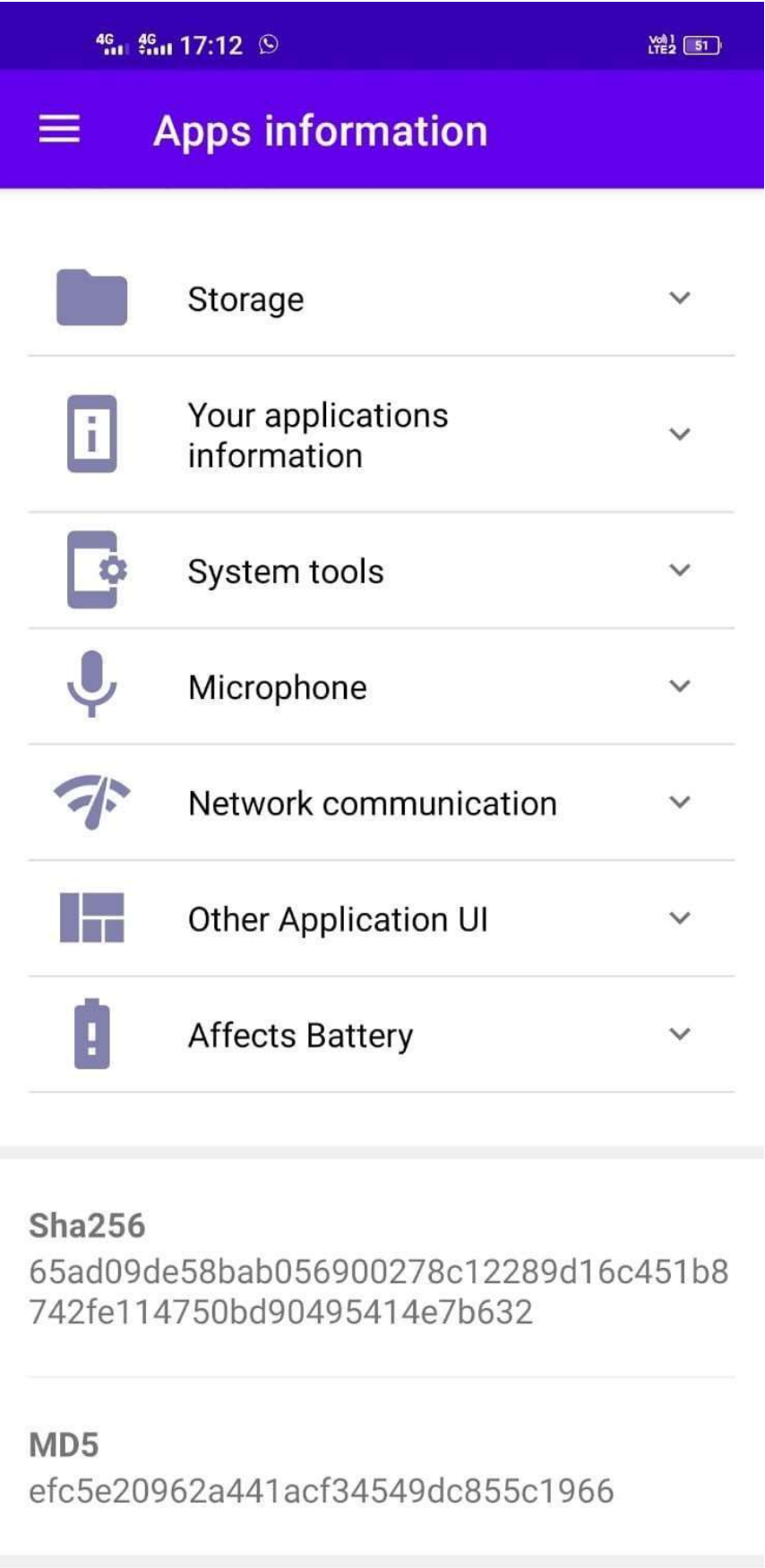


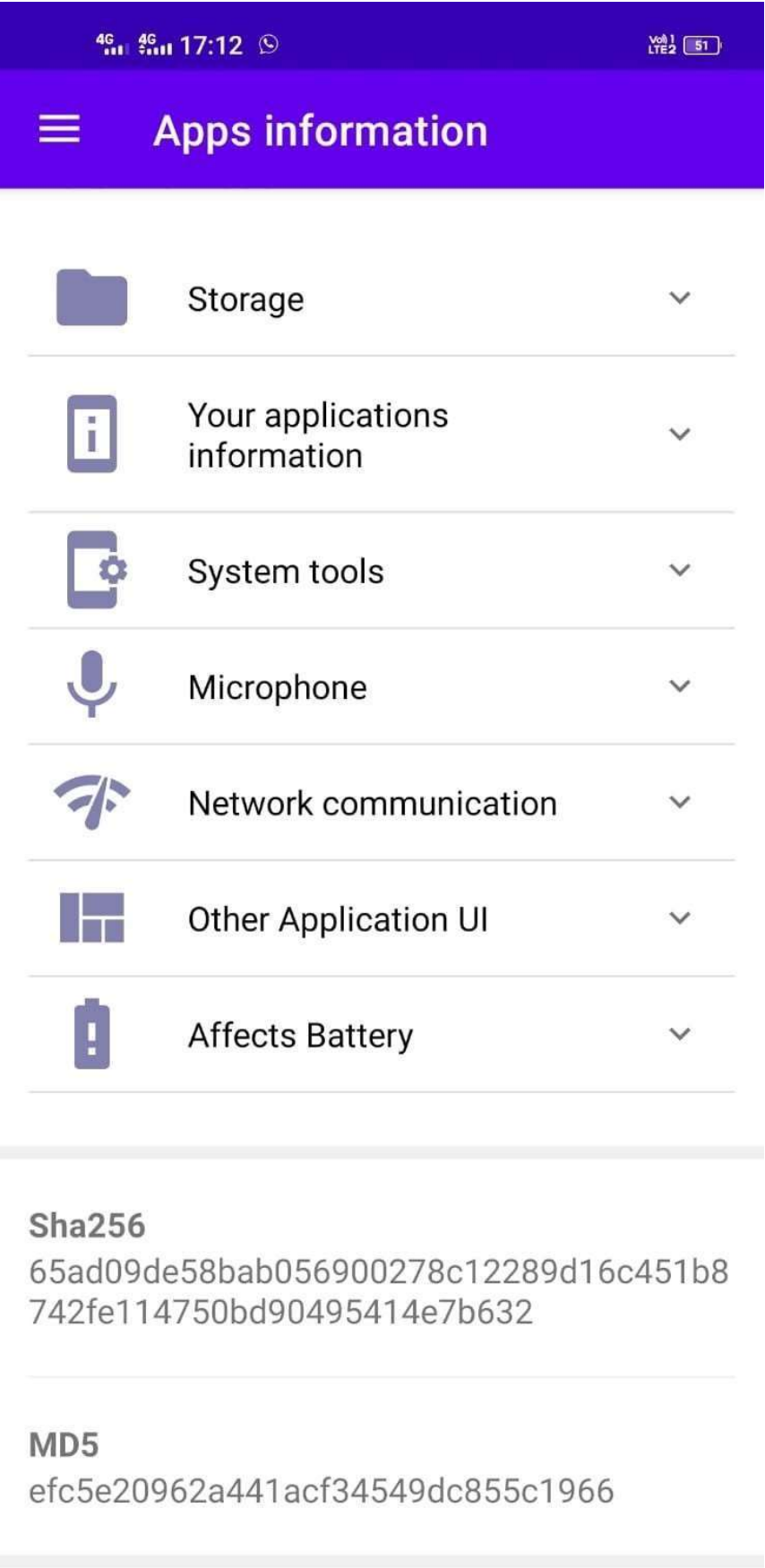




Sha256








4G 4G 17:11

Vol 1 LTE2 51

≡

Apps information



AnyDesk

Versión 6.6.0

Quantity Score

3.2/10

Quality Score

5.88/10

Permissions

>

Category

Productivity

Package

com.anydesk.anydeskandroid

Path

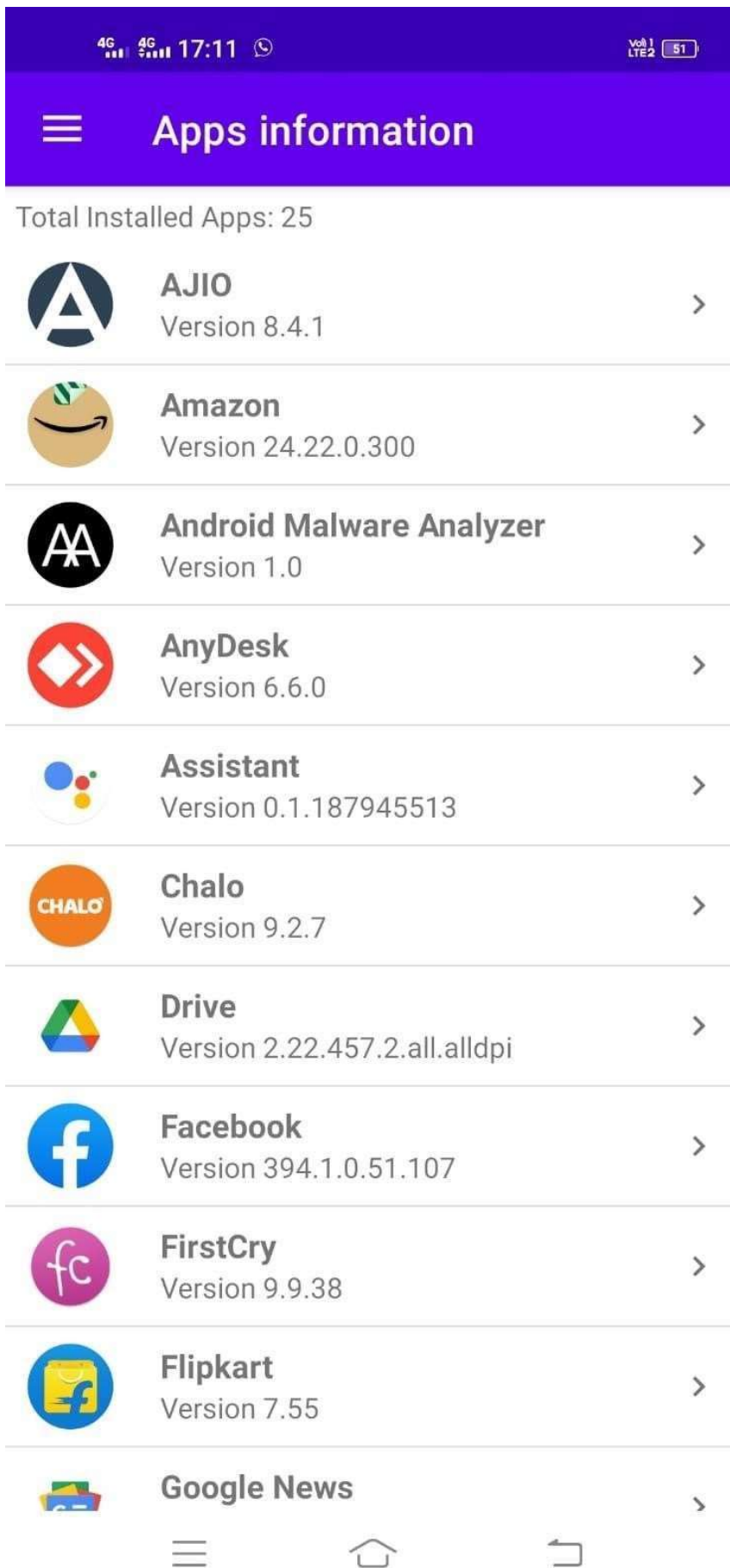
/data/app/~~R2kyakbiSPZnnJceDsKx-A==/
com.anydesk.anydeskandroid-
VBp9AW7_leNilCfqyxjwrw==/
base.apk

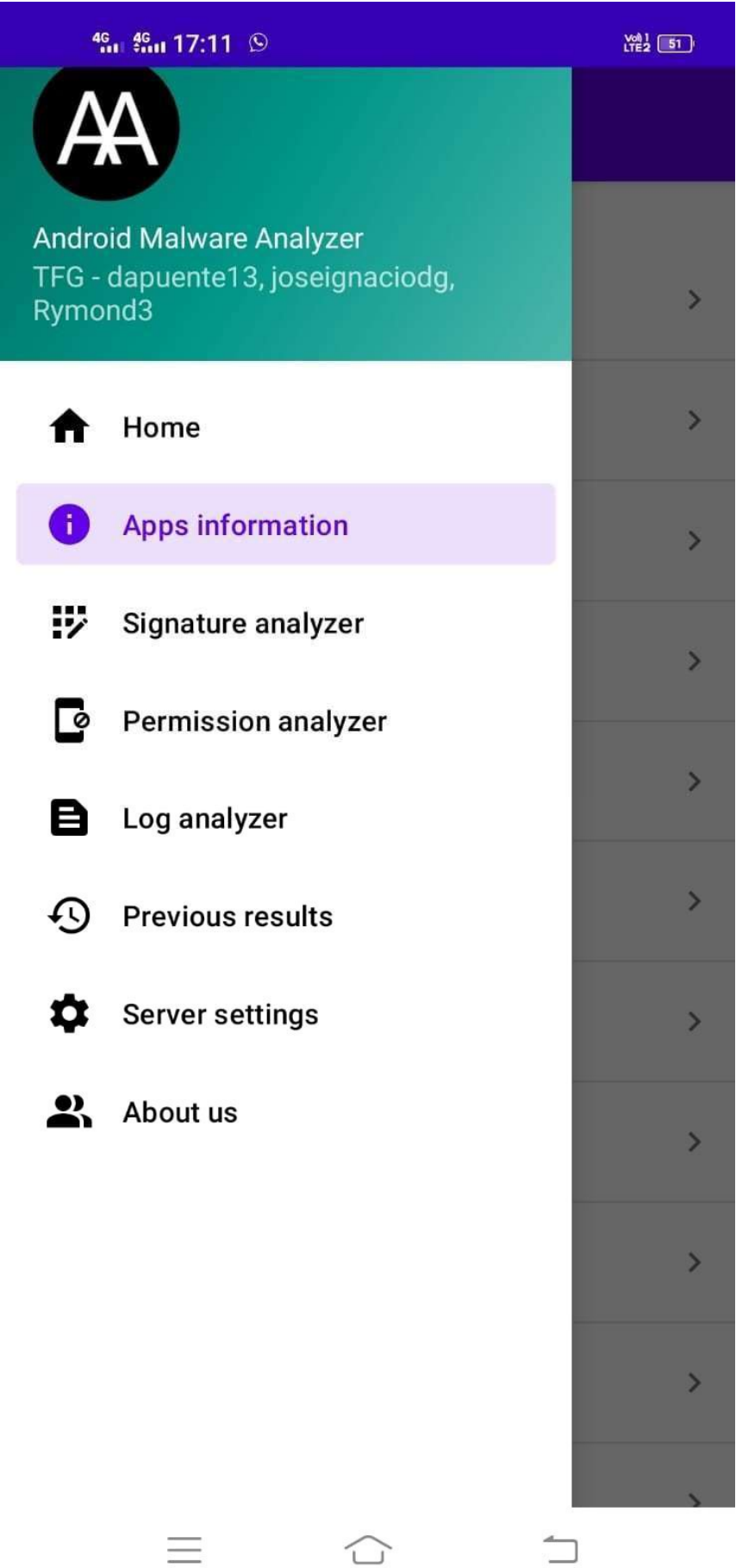
More Information

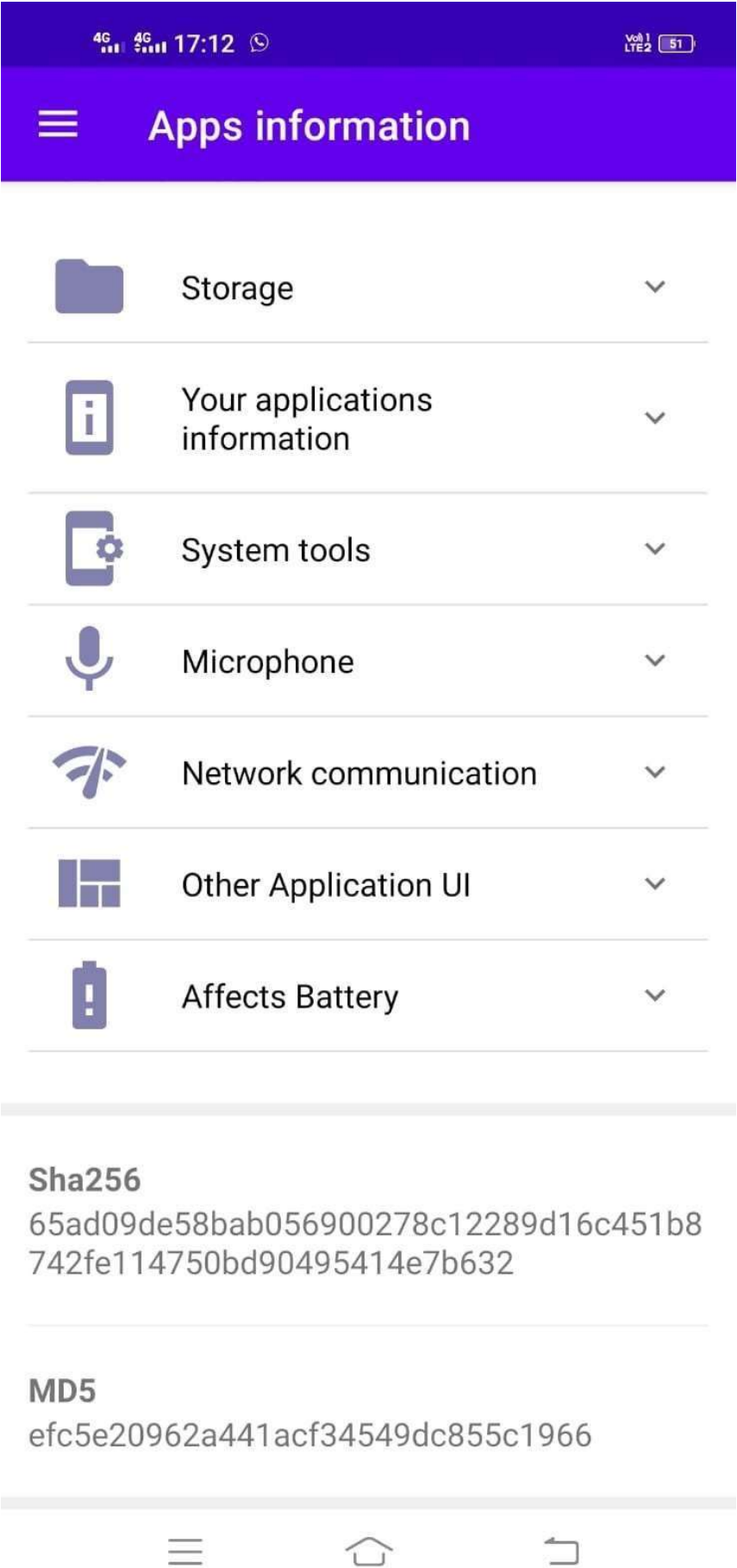
≡

🏠

↩








4G 4G 17:11

Vol 1 LTE2 51

≡

Apps information



AnyDesk

Versión 6.6.0

Quantity Score

3.2/10

Quality Score

5.88/10

Permissions

>

Category

Productivity

Package

com.anydesk.anydeskandroid

Path

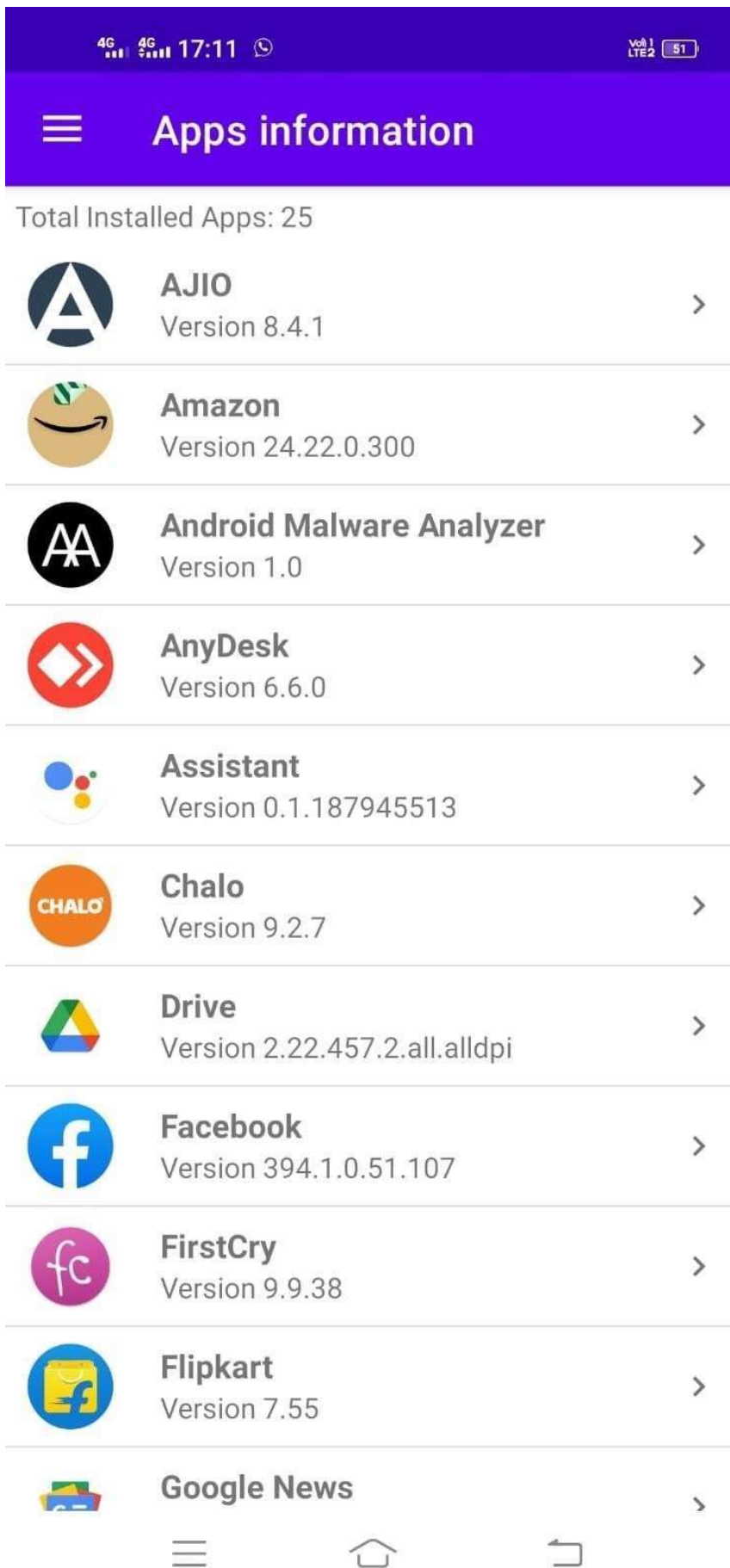
/data/app/~~R2kyakbiSPZnnJceDsKx-A==/
com.anydesk.anydeskandroid-
VBp9AW7_leNilCfqyxjwrw==/
base.apk

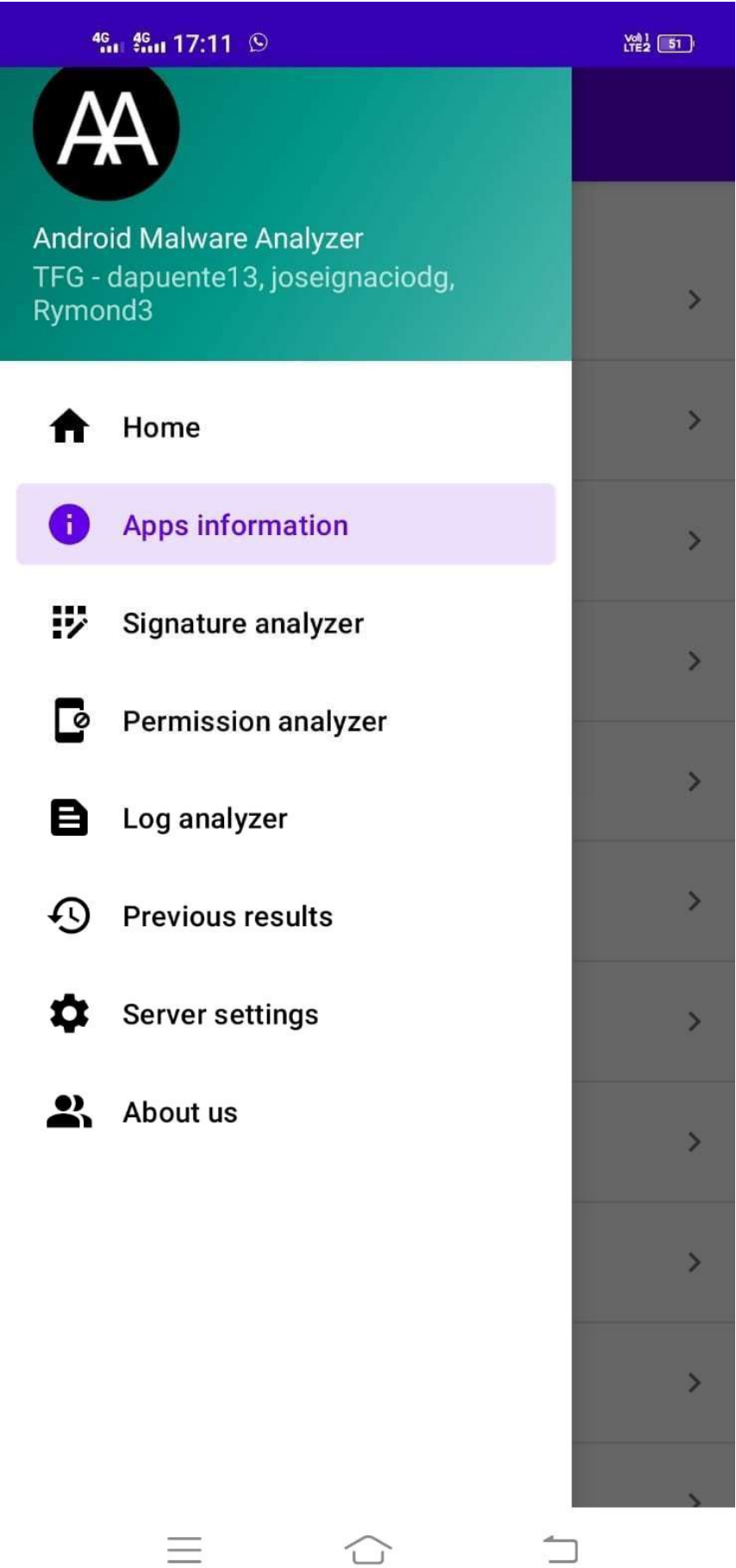
More Information

≡

🏠

↩







Last analysis performed:

No analysis performed yet

Apps whose hash is yet to be analyzed:



AnyDesk



Assistant



Chalo



CONCLUSION

In this system, if your app handles personal or sensitive user data, please also refer to the additional requirements in the "Personal and Sensitive Information" section below. These Google Play requirements are in addition to any requirements prescribed by applicable privacy or data protection laws. We proposed, A user who wishes to install and use any third party app doesn't understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scene.

REFERENCES

- [1] Yerima, S. Y., Sezer, S., & McWilliams, G. (2014). Analysis of Bayesian classificationbased approaches for Android malware detection. *IET Information Security*, 8(1), 25-36.
- [2] Sarma, B. P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., & Molloy, I. (2012, June). Android permissions: a perspective combining risks and benefits. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies* (pp. 13-22). ACM.
- [3] Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012, February). Detecting repackaged smartphone applications in third-party android marketplaces. In *Proceedings of the second ACM conference on Data and Application Security and Privacy* (pp. 317-326). ACM.
- [4] Chia, P. H., Yamamoto, Y., & Asokan, N. (2012, April). Is this app safe?: a large scale study on application permissions and risk signals. In *Proceedings of the 21st international conference on World Wide Web* (pp. 311-320). ACM.
- [5] Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P., & Wang, X. S. (2013, November). Appintente: Analyzing sensitive data transmission in android for privacy leakage detection. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 1043-1054). ACM.
- [6] Zhang, W., Li, X., Xiong, N., & Vasilakos, A. V. (2016). Android platform-based individual privacy information protection system. *Personal and Ubiquitous Computing*, 20(6), 875-884.
- [7] Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011, October). Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security* (pp. 627-638). ACM.
- [8] Lin, J., Amini, S., Hong, J. I., Sadeh, N., Lindqvist, J., & Zhang, J. (2012, September). Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (pp. 501-510). ACM.

- [9] Mann, C., & Starostin, A. (2012, March). A framework for static detection of privacy leaks in android applications. In Proceedings of the 27th annual ACM symposium on applied computing (pp. 14571462). ACM.
- [10] Stevens, R., Gibler, C., Crussell, J., Erickson, J., & Chen, H. (2012, May). Investigating user privacy in android ad libraries. In Workshop on Mobile Security Technologies (MoST) (p. 10).