جامعة الإمام عبدالرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

CYS 507- Mobile & Wireless Security

Academic Year (2025-2026) – Term II

Course Project.

PERMISSIONANALYZER APPLICATION

Section: 9FY02		
Group: 2		
No.	**Student Name**	**ID**
1	Dina Hassan Alamri	2220001993@iau.edu.sa
3	Renad Hashim Alharbi	2220003512@iau.edu.sa
4	Rhmah Ali Alshehri	2220006636@iau.edu.sa
5	Sarah Ayed Alshahrani	2220002351@iau.edu.sa

Instructor: Dr. Abdulrahman Alharby

Table of Contents

Table of Figure

1. Abstract

Users of Android are generally unaware of what permissions their apps have because they do not have sufficient knowledge to understand the security ramifications of granting certain app permissions in order to prevent loss of sensitive information about the user. Apps that want to access your phone's contact list, GPS, Camera, Microphone, etc., can get these accesses by requesting permission from you and while the permission model is intended to protect user privacy, the fact that many users will grant such requests without realizing it, creates serious privacy and security issues for users.

The goal of this project is to develop a Permission Analyzer Application, specifically developed for Android. This system analyzes all applications installed on the user's device statically and determines if there are potentially dangerous uses of declared permissions. The system does this by using the Android PackageManager API to extract all declared permissions, comparing those extracted permissions against a predefined list of high-risk permissions and then classifying each permission usage based on a pre-defined set of rules to determine a risk level for that permission (Low, Medium, High).

The system provides a simple method of determining the risk associated with the permissions requested by each of an individual's apps. It does so by providing a way to easily translate complicated permission structures into a risk-level that the average user can interpret and understand. In doing so, it increases the user's ability to make informed decisions when installing apps and increases the transparency of mobile security. The project shows how simple static analysis methods can be used to increase the transparency of mobile security for users of the Android operating system.

2. Introduction

2.1 Background

Android has a permission-based security framework, which restricts how apps can access sensitive parts of the system. The app developer is required to declare all permissions required for operation of the app and the user is then required to approve those permissions when the app is first installed or at run time. Permission determines what critical system resources an application will have access to, Example: access to GPS, Camera, Microphone, SMS, Call Logs, Contacts, Device Storage.

Android categorizes permissions by two protection levels, normal and dangerous. Normal permissions represent a low-risk threat to user privacy while dangerous permissions allow applications to access sensitive user data or hardware components. Misuse or over-granting of

dangerous permissions could potentially lead to a loss of user privacy and/or unauthorized surveillance.

Although this structured security model exists in Android, many users do not have sufficient technical knowledge to understand the risks associated with permission requests. Therefore, applications requiring extended access rights are frequently installed without a corresponding awareness of security.

2.2 Problem Description

Android displays the requested permissions but does not provide a simple, quantifiable measure of risk. Users are provided with the technical name of the requested permission (READ_SMS, ACCESS_FINE_LOCATION), with no context or summary of risk.

Several problems exist as a result:

- Users can't easily identify if an application is over-privileged.
- There is no direct risk indicator summarizing the extent of exposure from declared permissions.
- Declared permission information is fragmented and requires technical knowledge to interpret.
- User security awareness is completely dependent upon the users' knowledge.

Therefore, there exists a requirement for a system that will analyze the declared permissions and classify the risk of those permissions in a manner that is both easy to understand and accessible to non-experts.

2.3 Objectives of the Project

The main goal of this project is to develop an Android-based Permission Risk Analyzer that will evaluate the applications installed on a device based on the permissions they claim to require.

More specifically, the goals include:

- Obtain the list of applications currently installed on an Android device.
- Obtain a list of declared permissions using the Android PackageManager API.
- Identify and segregate permissions categorized as dangerous.
- Develop a rule-based classifier to determine the level of risk associated with each application.
- Classify each application as being in a Low-Risk, Medium-Risk, or High-Risk category.
- Display the results in a format that is user-friendly and understandable.

Achieving the above objectives should increase user awareness of the potential risks involved with installing various mobile applications and create a simpler process for evaluating the risks associated with mobile applications.

3. System Architecture

3.1 System Overview

The Android Permission Risk Analyzer was developed using a modular design in order to allow for separation of the data collection process, risk assessment, and display of user interfaces. All data processing occurs on the device itself and does not require access to any external servers or the Internet.

The overall system is comprised of 4 key modules:

1. the User Interface Layer
2. the Application Scanner Module
3. the Permission Extraction Module
4. the Risk Classification Engine

Each module serves a purpose in completing the overall analysis workflow.

3.2 Architectural Components

3.2.1 User Interface Layer

The User Interface (UI) Layer is responsible for the following:

- Displaying a list of installed applications
- Displaying the calculated risk level for each application
- Allowing users to interact with the system by selecting an item to see the full list of permissions
- Organizing the display of the full list of permissions for each application

The UI layer is built using Android XML layouts and Kotlin-based Activity classes. There are 2 main views that will be displayed in the UI layer:

- MainActivity - Display a list of installed applications along with their respective risk level
- DetailsActivity - Display the full list of permissions for the selected application

3.2.2 Application Scanner Module

The Application Scanner Module is used to retrieve a list of installed applications on the device. This is done using either of the following methods provided by the PackageManager class:

PackageManager.queryIntentActivities()

PackageManager.getInstalledPackages()

To improve both the performance and relevancy of this process, the system will filter out applications that do not meet one or both of the following conditions:

- Are not visible as launchers
- Are part of the system (can be optionally filtered)

As such, the analysis will focus solely on user installed applications.

3.2.3 Permission Extraction Module

Using the package name, the Permission Extraction Module will extract all declared permissions that were specified when the application was published. This can be achieved using either of the following methods:

PackageManager.GET_PERMISSIONS

packageInfo.REQUESTEDPERMISSIONS

If no permissions were declared, the system will assign an empty list to the variable representing the permissions for the application.

This module is designed to perform only static permission inspections (no runtime monitoring).

3.2.4 Risk Classification Engine

The Risk Classification Engine will analyze the extracted permissions and classify them into risk categories based upon a predefined set of sensitive permission keywords.

The classification logic will:

- Count how many times a dangerous permission appears in the extracted permissions
- Apply decision-making rules based upon a defined threshold
- Determine a risk level for the application (Low/Medium/High)

Unlike other systems that use machine learning to evaluate risk, this engine uses deterministic logic and heuristic evaluations.

3.3 System Workflow

The operational workflow of the system will follow the following steps:

1. Application will request all available applications.
2. For each application:
    - Get the package name.

- Get the declared permissions.

3. Compare the declared permissions to the predefined list of dangerous permissions.
4. Count the number of matches to the dangerous permissions.
5. Apply decision-making rules based upon a defined threshold.
6. Store the result in an AppInfo object.
7. Display the results in a RecyclerView.
8. When a user clicks on a particular application → Open the detailed permission screen for that application.

3.4 Design Considerations

The system had many constraints for operation and design which were put in place to make sure it would be easy to use and reliable. It was made to run completely on the device (not on an outside server) to keep the user's information private and to keep all data local to the user. No information about the user's permissions are sent from the device to anywhere else. To prevent using too much device resources and to allow the app to run on as many different Android devices as possible, the system needed to have low computational complexity. In addition to making the system simple, transparent and reproducible, the system also needed to be able to display risk levels in a way that could easily be understood by non-technical users. Overall, the architecture of the system emphasized the need to prioritize these factors above the need to create complex algorithms.

4. Methodology

4.1 Development Environment

The Permission Risk Analyzer has been developed by means of the following tools and environments:

- Android Studio as the Integrated Development Environment (IDE)
- Kotlin as the main development language
- Android SDK with an appropriate API level setting
- Android Emulator for the first functional tests
- Real Android devices for verification and for real world tests

All the operations that involve processing are executed locally on the device without need of Internet connection nor remote API access. The development testing were made in both emulator and real device for ensuring the functional correctness, the compatibility and the performance homogeneity of the application under the different execution context.

4.2 Data Collection Method

Data collection for this project occurs through direct data retrieval from the Android operating system.

Process consists of:

1. Getting installed apps via Android PackageManager
2. Filtering visible applications to increase performance
3. Using PackageManager.GET_PERMISSIONS to retrieve declared permissions
4. Saving collected app data in structured objects (AppInfo data class).

Since all data utilized for risk evaluations comes from the Android OS and is based on the declared permissions of each individual application; there is assurance that the data accurately represents each app's permissions.

4.3 Risk Classification Algorithm

The method of classifying applications using this system is based on a deterministic approach that uses a set of pre-defined "dangerous" permission categories in order to produce transparent, consistent, and repeatable results. The system has a pre-defined list of high-risk (“dangerous”) permission categories which include:

- Permissions related to SMS
- Permission to access a contact
- Access to use the camera and/or microphone
- Permission to use location services
- Permission to access call logs
- Permission to access external storage

For each application that is installed on the device, the system will perform a static analysis of the declared permissions to compare the declared permissions with the pre-defined dangerous permission categories. The number of declared permissions that are considered dangerous will be counted and compared against a pre-defined set of threshold values in order to determine the risk level of the application.

Let:
D = Number of dangerous permissions requested by an application.

Then,
The risk level R is defined as:
High Risk = $D \geq 5$
Medium Risk = $2 \leq D \leq 4$
Low Risk = $D \leq 1$

The threshold values were chosen to clearly represent the amount of privilege an application had to expose to the user, and thus the potential risk to their privacy and/or security. For example, an application that requests multiple high-risk privileges such as SMS, contacts, camera and location may be viewed as having a greater risk than one that does not request these types of privileges. Thus, counting the number of dangerous privileges that an application requests provides a direct and efficient way to assess the risk of an application.

4.4 Testing and Validation

The Permission Risk Analyzer has been validated in the first instance through testing on an Android Emulator for the validation of the functionality of the tool, which demonstrated that the tool performed correctly in the emulation environment. Upon completing these tests successfully the tool was then deployed to a physical Android device to test its reliability under normal operating conditions.

Tests have been conducted to validate the performance of the tool's ability to extract permissions accurately, to ensure that the risk classifications are consistently applied to applications with differing permission sets, to evaluate the responsiveness of the User Interface (UI), and to ensure the application continues to function when it is operating in an offline environment.

The Permission Risk Analyzer has been evaluated against applications with the following characteristics:

- Applications that request a minimum number of permissions
- Applications that request multiple permissions in various categories.

In both the emulator and physical device environments the Permission Risk Analyzer has produced consistent classification results; this demonstrates the deterministic aspect of the risk evaluation model and confirms that the implementation of the Permission Risk Analyzer will produce stable and predictable results across different execution contexts.

5. Implementation

5.1 Application Structure

The Permission Risk Analyzer app was developed with an Android multi-activity architecture. All the main features of the app are as follows:

- **MainActivity** — displays all the applications that have been installed and their calculated risk level.
- **DetailsActivity** — provides users with detailed information about permissions for a selected application.

- **AppsAdapter** — manages RecyclerView data binding.
- **AppInfo** — is a class to store application name, package name, and a list of permissions.

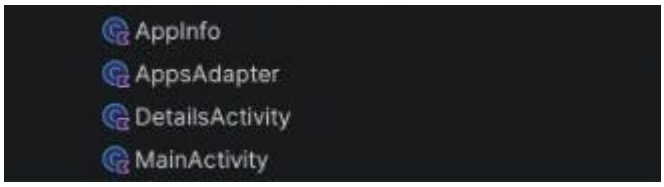

Figure 1 Android Application Class Structure

The modular design allows the developer to separate concern regarding how the data is processed from how the data is presented in the user interface.

5.2 The MainActivity Implementation

MainActivity is the Application's main activity that represents the entry point of the application and it performs a few functions:

1. It loads the RecyclerView.
2. It invokes the permission scanning method.
3. It instantiates the adapter.
4. It handles click events to call the details screen.

loadInstalledApps():

- Loads all installed apps.
- Gets the declared permissions of each app.
- Creates an AppInfo object for each app.
- Calls the RecyclerView adapter with the app information.

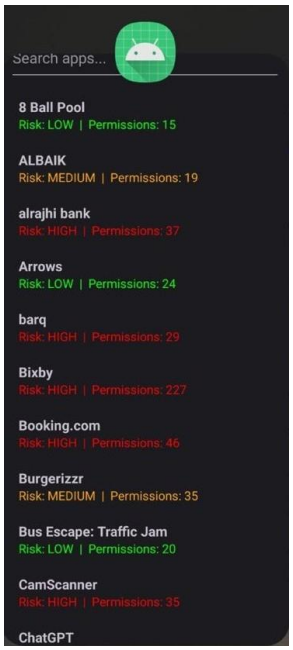

Figure 2 PermissionAnalyzer Application

5.3 RecyclerView and Adapter Design

RecyclerView is used in the app to present the installed apps in an efficient manner.

AppsAdaptor Class is responsible for:

- Displaying each application's information as a single item in the List.
- Application Name Display
- Calculation & Display of Application Risk Level
- Color Coding:
    - Red → High Risk
    - Orange → Medium Risk
    - Green → Low Risk

5.4 Risk calculation function

The system evaluates dangerous permissions by using a keyword matching method.

Step-by-Step Process:

- Creating a database of risky keywords for permissions.
- Process each application's declared permissions.
- Counts the number of matching permissions.

- Apply rule thresholds.

Classification results are shown as a result with the name of the app.

Deterministic evaluation and transparent logic will be achieved with this implementation.

5.5 DetailsActivity Implementation

Upon selection by a user of an Application in the Main List:

1. A new Intent will be created.
2. Information regarding the selected Application (name of application, package name, and list of requested permissions) will be passed to DetailsActivity as extras.
3. DetailsActivity will receive this information and display it to the user.

The details view will show:

- Name of Application
- Package Identifier for Application
- All Declared Permissions for Application

If an application does not declare any permissions, the details view will indicate "No permissions requested".

5.6 Data Flow Summary

describes the data flow throughout the application:

Figure 3illustrates the operational workflow of the Permission Risk Analyzer system.

Figure 2 displays the operational workflow for Permission Risk Analyzer System. Data will be initiated when the user executes the application, where the MainActivity will be activated as the primary entry point into the system.
Following the execution of the MainActivity, the system will use the Android PackageManager API to scan through all applications that have been installed on the device and that are visible to the Launcher.

Once the applications have been scanned, the declared permissions associated with each application will be extracted and provided to the risk evaluation function.
The function will evaluate the amount of dangerous permissions assigned to the application and assign a corresponding risk level to the application based upon the number of dangerous permissions that were identified (Low, Medium, High), and once the calculated risk level has been determined, the risk level will be stored in AppInfo objects and presented in the RecyclerView interface.

If a user selects an application from the list of applications that was presented in the RecyclerView interface, an Intent will be generated and executed to open the DetailsActivity, which will provide a detailed view of the application that the user selected.
In addition to providing a detailed view of the application, the details activity will also display the metadata and permission list of the application, so that the user may review the complete list of permissions that the application requests access to.
As illustrated by the linear structure of the above-mentioned workflow, it allows the separation of the scanning layer, analysis layer, and presentation layer to be clearly defined and separate from one another, which can contribute to maintaining the architecture of the application and its overall maintainability.

The linearity of the data flow provides for a clear and well-structured architecture that lends itself to maintenance.

5.7 Performance Optimization Considerations

In order to achieve good execution times with the least amount of resource utilization possible and to avoid additional CPU load, the following constraints have been defined for the operation of the system:

- Only visible applications launched by the user will be analyzed.
- Applications that belong to the system are always excluded from the analysis if they do not need to be made available to an application.
- The analysis is purely static; therefore it can only analyze declared permissions.
- Background services as well as continuous monitoring of the system are not employed.

The described measures help reduce the overhead caused during the analysis process, the required memory is reduced, and the UI remains responsive. To confirm that the system works reliably and performs acceptably in practical environments, the system has been tested using the Android Emulator and a physical Android-based device.

6. Results and Evaluation

6.1 Observed System Behavior

The system was tested on a test Android device to observe how static inspection of permissions can be used to analyze the installed apps on the device.

The system performed the following actions upon execution:

1. It retrieved the list of all installed applications from the PackageManager.
2. It extracted the declared permissions from each application.
3. It evaluated each app's permissions based on a predefined set of rules to determine its associated risk level.
4. It displayed the installed applications in the User Interface grouped by their respective risk levels.

Figure 4 Interface on Physical Android Device Showing Risk Classification Results

Observations:

- Apps which did not declare any permissions were properly identified as Low Risk.
- Apps requesting high-risk permissions (i.e. location, camera, SMS) were consistently rated as High Risk.
- Processing time did not increase significantly when evaluating many applications at once.
- The system was able to operate entirely offline without needing to use any additional services or APIs.

These observations support the functionality of the permission extraction and risk evaluation components of the system.

6.2 Example Classifications

Several applications were analyzed and tested for their classification logic.

EXAMPLE 1: ALRAJHI BANK APPLICATION

- **Application Name**: alrajhi bank
- **Declared Permissions** (some examples): Camera; record audio; access fine location; access coarse location; read contacts; write contacts; read external storage; read phone state; use biometric; nfc
- **Total Declared Permissions Requested** : 37
- **Risk Level Classification**: high
- **Justification**: Alrajhi Bank app has requested a large number of declared permissions that are sensitive. Examples of such include access to your camera, microphone ability to record audio, location service, contacts, biometrics, and even your mobile device state. Many of the above permission types may have functional justification to use them as part of banking functions (i.e. using a fingerprint reader for authentication purposes, using the camera to scan documents). However, when you add up the total amount of high risk declared permissions (the D > 5) then this results in an overall classification of High Risk under the deterministic rules based model. The reason why this example illustrates how the system identifies apps with excessive access to sensitive resources is because the system does not rely on the app's reputation or the app's intended use case.

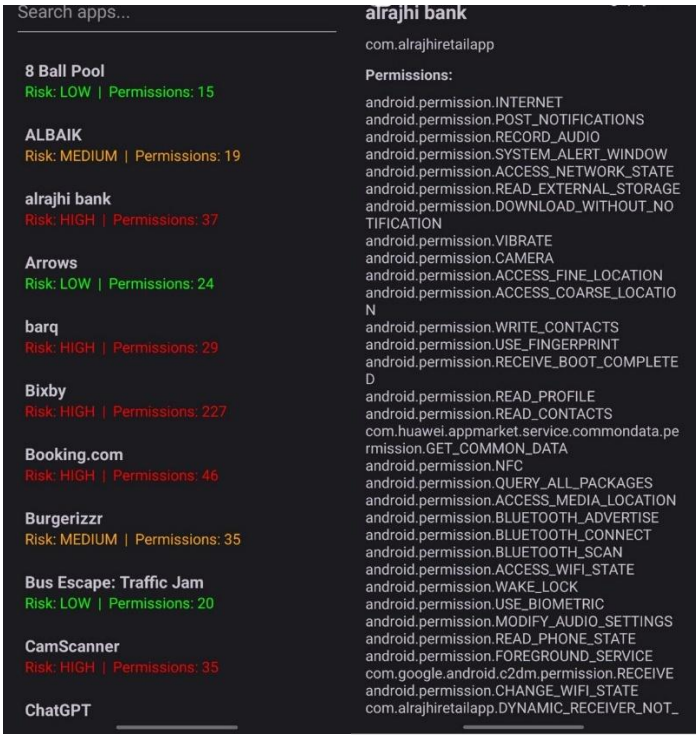

Figure 5 High Risk Classification of Alrajhi Bank Application (Main Screen View)

Figure 6 Detailed Permission View of Alrajhi Bank Application

EXAMPLE 2: GOOGLE CALENDAR APP

- **Application Name**: Google Calendar
- **Declared Permissions** (some examples): READ_CALENDAR, READ_CONTACTS, CALL_PHONE, READ_SYNC_SETTINGS, WRITE_SETTINGS
- **Total Permissions Requested:** 26
- **Risk Level Classification**: Medium
- **Justification**:
  This application is requesting a lot of permissions, such as READ_CALENDAR, READ_CONTACTS, CALL_PHONE, READ_SYNC_SETTINGS, WRITE_SETTINGS, etc., which provide it with access to calendar data, your contacts, the capability to call from the app, and all sync settings in your device. These are potentially justified because they are functional for scheduling and reminder purposes; however, the quantity of dangerous permissions identified by the rule-based model falls into the Medium risk level (D = 2–4), which shows the system can identify moderately privileged apps versus apps that have access to many high risk-sensitive resources.

Figure 7 Medium Risk Classification of Google Calendar (Main Screen View)

Figure 8 Detailed Permission View of Google Calendar Application

EXAMPLE 3: 8 BALL POOL APPLICATION

- **Application Name**: 8 Ball Pool
- **Declared Permissions** (some examples): WAKE_LOCK, ACCESS_NETWORK_STATE, VIBRATE, INTERNET, BILLING, POST_NOTIFICATIONS, FOREGROUND_SERVICE
- **Total Declared Permissions Requested**: 15
- **Risk Level Classification**: Low
- **Justification**:
  The 8 Ball Pool application has requested many operational permissions (network access, advertisement, billing, etc.) in addition to requesting service level permissions to run in the background, however, it is not requesting any permissions defined as dangerous by the predefined rules based model (SMS, contacts, audio recording, GPS tracking, call history, etc.). Therefore, the number of dangerous permissions (D), which defines the risk level classification as low ($D \leq 1$) was determined. The difference here is that the system classifies the total amount of declared permissions versus the total amount of permission types.

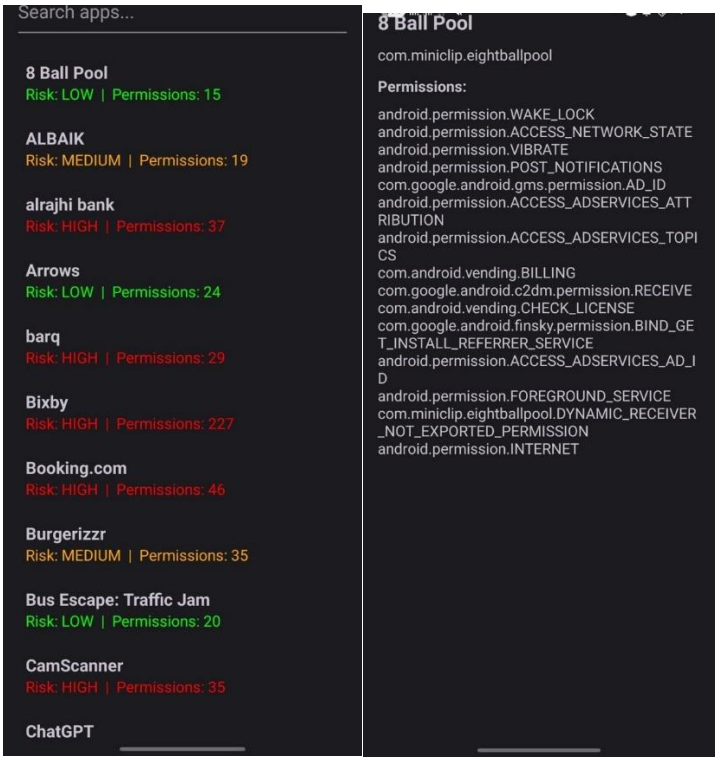

Figure 9 : Low Risk Classification of 8 Ball Pool Application (Main Screen View)

Figure 10 : Detailed Permission View of 8 Ball Pool Application

6.4 Security Implications

There are numerous observations in the data about privacy concerns with regard to the most common applications on modern Android devices. Many of the most common applications have requested access to very personal and private information about the Android device, including its location, all of a user's contacts, a user's camera, and much more about the state of the user's device. Although many of these requests would likely be necessary to allow an application to perform its intended functions, each additional request that an application makes to access user data increases the amount of user data that is exposed by the application.

Results from this study show that static permission analysis provides a transparent and cost-effective way to identify applications that could have too many permissions and therefore expose user data to a higher degree than is needed. Static permission analysis allows users to clearly see what risks exist when they install an application based on the permissions required by the application.

Static permission analysis does not analyze how an application will behave at runtime; therefore, the identified risks represent potential risks to the user, but do not necessarily represent actual malicious behavior by the application. This is discussed further in the limitations section of this report.

In general, the study has contributed to raising the level of awareness of users as to which types of applications have the greatest need for various forms of data from a user's Android device and to assist users in making more informed decisions about whether or not to download an application and manage their privacy settings.

6. Limitations

The PermissionAnalyzer uses only static methods for analyzing declared application permissions. Therefore the analyzer will provide an assessment of a possible permission exposure instead of determining if a permission has been used maliciously, exploited or abused at runtime. The Permission Risk Analyzer utilizes a classification model that performs deterministic keyword matching and threshold-based counting to classify risk, these characteristics ensure that the results are repeatable however can be overly simplistic in assessing risks as the analyzer will assign equal risk weighting to all dangerous permissions.

In addition, the Permission Risk Analyzer does not assess whether the justification for declaring certain permissions is appropriate based on the intended function of the application. As such the analyzer would not make differentiations between legitimate usage scenarios (i.e., using biometric authentication in banking apps) and excessive requested access rights. The model also

does not utilize behavioral analytics, anomaly detection, runtime monitoring, or external threat intelligence feeds.

The decision to implement this design was made in order to maintain both transparency and interpretability of the Permission Risk Analyzer, while maintaining minimal computational overhead. However, this design limits the Permission Risk Analyzers' ability to identify complex or stealthy behaviors.

7. Future Work

Several ways to add to the Permission Risk Analyzer application's ability to analyze risks are possible. An addition that should be made is a weighted-risk scoring method for assigning dangerous permission severity weights. This will allow a much finer level of detail as well as a much more accurate analysis of an application's overall risk compared to the current model which has a single (uniform) threshold for determining when an application is "at-risk".

Future versions of this tool could also include runtime monitoring to track how an application uses its permissions and detects anomalies in an application's use of APIs that could potentially represent abuse/misuse of those permissions. Also, the predictive capability of the tool can be improved by using machine learning methods trained on labeled data sets of known good (benign) and known bad (malicious) applications.

In addition, the tool could perform some form of contextual analysis of whether or not an application's requested permissions match up with what type of application it is supposed to do. The ability to identify applications that have been granted more than enough permissions to perform their intended functions (over-privileged applications), will be much easier than it is now and will help in identifying applications that are more likely to represent a risk to an application's security posture.

Finally, improvements in visualizing the risks associated with each application, as well as improving the tools for guiding the user through interpreting these risk indicators, will assist non-technical users in making educated decisions about whether or not they want to install an application.

8. Conclusion

This project presented the design and implementation of an Android-based Permission Risk Analyzer designed to enhance users' awareness of the risks associated with the permissions requested by the applications they install. The system leverages the AndroidPackageManager API to extract the permissions declared by applications that are currently installed on the device and applies a deterministic, rule-based classification method to categorize the applications into

one of three risk levels: Low, Medium, or High. The modular architecture allows the system to operate efficiently, off-line, and provides a clear separation of responsibility among the three main modules of the system: data extraction, risk evaluation, and user interface. While the system does not analyze the behavior of the application or determine if the application intends to act maliciously, this project clearly demonstrates that basic static analysis methodologies can easily convert complex permission structures into understandable indicators of risk. Furthermore, the project demonstrates the need for transparency in mobile security and illustrates that simple heuristic models can contribute greatly to the awareness of security in the Android ecosystem.

9. Reference

[1] Android Developers, “Security Overview,” Android Developers Documentation. [Online]. Available: https://developer.android.com/guide/topics/security/overview

[2] Android Developers, “App Permissions Overview,” Android Developers Documentation. [Online]. Available: https://developer.android.com/guide/topics/permissions/overview

[3] Android Developers, “PackageManager,” Android SDK Reference Documentation. [Online]. Available: https://developer.android.com/reference/android/content/pm/PackageManager

10. Appendix

Appendix A – Source Code Repository

The complete source code of the *Permission Risk Analyzer* project is publicly available on GitHub: sarahalshahranicyber/PERMISSIONANALYZER-APPLICATION