

**Supervisor name:**

Dr. SULTAN S. ALQAHTANI

Quality Assessment for Local Banking Apps

|  |  |  |
| --- | --- | --- |
| **TEAM MEMBERS** | | |
| **Name** | **Email** | **ID** |
| Yousef Khalid Alturki | yktaltuki@sm.imamu.edu.sa | 440013537 |
| Ziyad Mohammed Al-Enizy | zunzizi@sm.imamu.edu.sa | 440018676 |
| Mohammed Omar Alsharef | Moalsharif@sm.imamu.edu.sa | 440013633 |

**Table of content**

24

21

20

18

16

15

13

13

12

11

11

9

8

7







67

5

4

3

1

8

CONCLUSION

7

READABLE

6

SECURITY

DOCMENTATION

UML DIAGRAM

5

4.7 WEBVIEW ENABLED CONTENT ACCESS

4.8 LOG FILE

4.9 DYNAMIC CODE LOADING

4.10 CLEARTEXT TRAFFIC

4

3

2

4.6 DATA STORGE

4.5 INSUFFICIENTLY RANDOM VALUES

4.4 SQL-INJECTION

4.3 INTENTS

4.2 TEMPORARY FILE

4.1 PERMISSIONS

INTRODUCTION

APPLICATION OVERVIEW

CHALLENGES AND LESSONS

**INTRODUCTION**

The purpose of the workshop is to analyze the quality of the source code of local bank applications using MobSF (Mobile Security Framework) as the main tool and some other additional tools that help us analyze the application to work on AlRajhi Bank - Android version -.

The goals are to analyze the application and find errors in the quality of the source code to improve and correct errors that may be performance or security issues of the code quality using some tools that help us achieve the goals.

The tools make it easy to check the quality of the code, discover weaknesses in the application and try to improve it by conducting a comprehensive analysis of the application and preparing a report explaining the most important points and negatives and show the strengths and weaknesses of the program and show if the program meet the important requirements that the program must maintain in terms for security and the privacy of the end user, this analysis is facilitated for programmers who will develop and improve the quality of code and solve the problems discovered.

**APPLICATION OVERVIEW**

|  |  |
| --- | --- |
| **APPLICATION NAME** | ALRAJHI |
| **VERSION** | 4.2 |
| **DATE OF ANALYSIS** | 23/9/2021 |
| **MOBILE OPERATING SYSTEM** | Android |
| **STATIC TOOLS** | MOBSF  Qark  AndroBugs  Smartdraw |

CHALLENGES AND LESSONS

In this workshop, we learned many things about analysis tools, and the differences between them, and how they work in real applications in Android Java versions.  
First, we tried to download several analysis tools ( such as SonarQube, Embold, Owasp zap), we did not know the difference between the tools and tried to run the tools in Alinma bank using the Windows environment and the tools did not work, we thought maybe these tools may not support the Windows environment, so we installed the tools in the MACOS environment and tried to run it in Alinma Bank. After the tools did not work, we thought Alinma bank has some issues or the tools do not support kotlin language, so we did download AlRajhi bank and try to run the tools, but we got the same thing in Alinma bank.

After we read about these tools and why they did not work in these applications, we discovered that there are different types of tools, the tools we try to run in the applications and did not work are dynamic tools that depend on applications are running and try to find bugs in working applications by testing inputs to cover almost all possible outputs and behavior of the program  
Bank applications do not work on dynamic analysis because they have multiple communications with other servers, so we use static analysis instead of dynamic analysis to obtain the analysis result of bank applications, static analysis is the testing and evaluation of an application by examining the code without executing the application, which works fine on applications that have other communications to other servers.

Security

Secure coding involves writing code in a high-level language to prevent potential vulnerabilities (which could expose data or cause harm within a targeted system).

Security needs to be a priority as you develop code,secure coding is more than just writing, compiling, and releasing code into applications, should become familiar with the techniques and tools that help code become secure.

The steps of analysis secure coding:

**Step 4: use sub tools**

Use the sub tools we have to see if give the same issues in MOBSF or MOBSF lead to false positive test.

**Step 1: run MOBSF Tool**

Upload android apk file to the tool to give us static analysis of the code and source code.

**Step 2: download source code**

Download source code to computer and open it with IDE.

**Step 5: search about issues**

MOBSF and sub tools gives some security issues and we are search about issues to get more information about it.

**Step 6: find solutions**

After we get more information about the issue, we try to get solution for the issue.

**Step 3: discover source code**

Discover the source code to find security issue.

* **Permissions**

Permissions can give apps control of your phone and access to some features using it or may possibly run in the background.

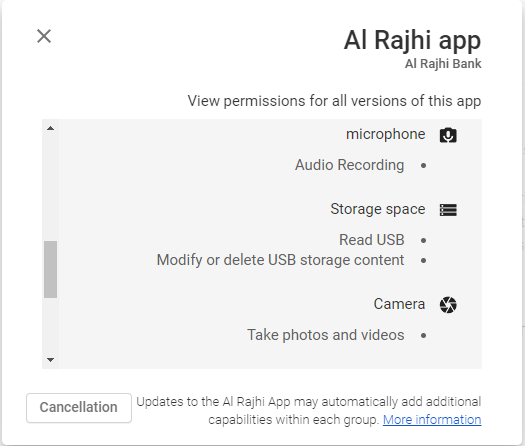
There are some permissions in the application that are not necessary to access them such as camera, audio, and location, these permissions are called Runtime permissions or dangerous permissions and these permissions are not dangerous but have the potential for misuse.

Some examples of dangerous permission misuse:

* **Camera**: the app can secretly turn on your camera and record what’s going on around you.
* **Location**: the app can secretly track your location to build a profile on your daily habits
* **Audio**: the app can secretly record what’s going on around you, including private talks
* **Storage**: the app can secretly read, change, and delete any of your saved documents, photos, and other files.

All these permissions AlRajhi bank asks for it, and should keep the private user data-sensitive information protected and do not ask about these permissions because there is a privacy concern for something that should be just a Bank app.

Some examples of permissions in AlRajhi Bank:



* **Temporary file (CWE-276**)

As the name suggests, temporary files are the kind of files that store temporary data that is created by the programs.

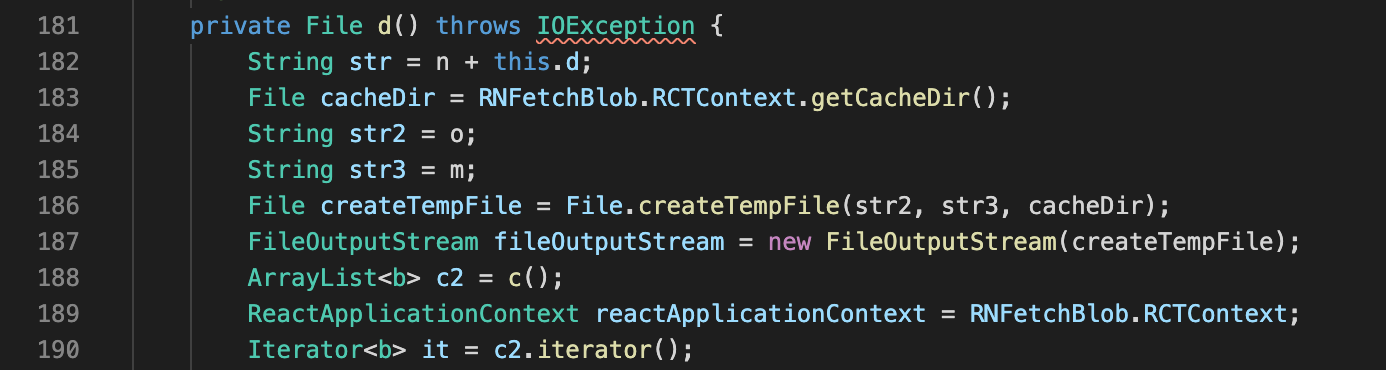
The MOBSF tool shows there is unsecured to written sensitive information in the temporary file, but this leads to false positives test because there are no security issues and no sensitive information written in temporary file, Alrajhi app uses OpenSSL which used to secure credit card transactions, data transfer, and logins.

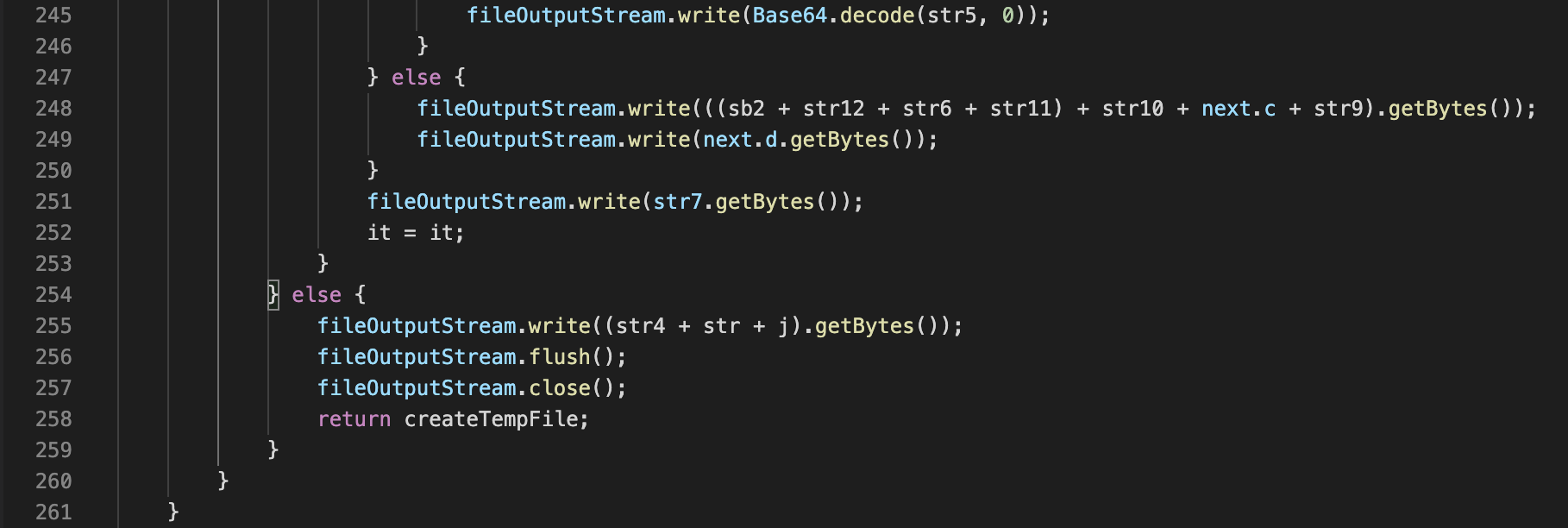
The reason for MOBSF to show this in secured issues for the temparary file the OpenSSL is creating a temporary file for temporary keys used for one purpose, so it is difficult to decrypt past messages because the key was temporary and discarded (once the session terminated) and throw it into a temporary file.

AlRajhi Bank should use SSL certification to keep the tractions data encrypted and make the high privacy for users.

The Qark and AndroBug tools do not show any security issues in temporary file.

The snapshots in source code for creating temporary file example:





Path: com/RNFetchBlob/a.java

* **Intents ( CWE-927)**

Intents is a messaging object you can use to request an action from another app component. Intent use for activities, services, and Delivering a broadcast.

There are two types of intents:

* **Explicit intents:** communication between two activity inside the application by specific the target component. You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example: when click button will show the photo in the same app.
* **Implicit intents:** communicates between two activities of different application, Implicit intents do not name a specific component.For example: when you want to share a photo will show a pop that contains the apps that support the sharing photo, these apps show by OS.

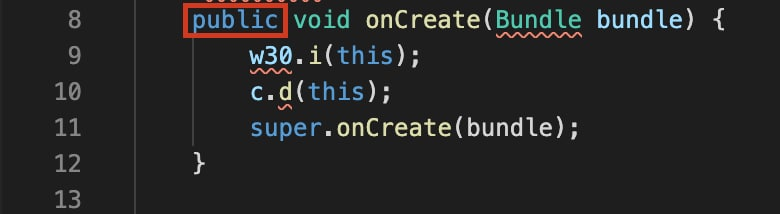
hackers can leverage the intent messaging to extract personal data or to call components without credentials by sending malicious intents to components.

Implicit intent does not specify a component that receives a message and insecure ways of using implicit intents may allow malicious applications to intercept or forge intents.

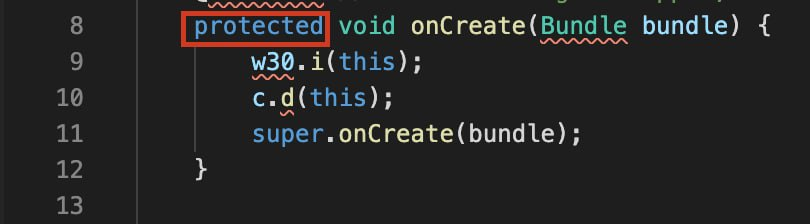
For sensitive data of users do not use implicit intent because the message when shared maybe some applications can interrupt the message.

In the source code we notice some intents use public method instead of protected method, should use protected methods because cannot be accessed from outside class to protect sharing with other apps on the device.

This is an example of one activity method issue:



this method should be protected like this:



Path:com/alrajhiretailapp/MainActivitydark.java

* **SQL Injection (CWE-89)**

A code injection technique that might destroy the database, SQL injection is one of the most common web hacking techniques and it is a powerful attack because it provides access to application databases.

MOBSF shows that there is a dangerous issue in the SQL database, but this leads to false positives test because we ran AlRajhi application into two tools (Qark, AndroBug), which shows that there are no issues in the SQL database.

We also tried to use the vulnerability in the application and it did not appear as expected from the vulnerability, which means that the AlRajhi application has prevented this vulnerability. In addition, we searched the source code to check if vulnerability was present, but vulnerability did not appear in the source code.

Path: com/reactnativecommunity/asyncstorage/e.java

* **Insufficiently Random Values (CWE-330)**

When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated and use this guess to impersonate another user or access sensitive information.

Graphical user interface, text, application

Description automatically generated

Path: defpackage/f30.java

Recommended to use a cryptographically strong random number generator "java.security.SecureRandom"

Graphical user interface, text

Description automatically generated

* **Data storge (CWE-276)**

Android provides several options for you to save persistent application data. The solution you choose depends on your specific needs, such as whether the data should be private to your application or accessible to other applications and the user and how much space your data requires.

External storage is the memory space such as SD card can also store application data, there’s no security enforced upon files you save to the external storage. All applications can read and write files placed on the external storage and the user can remove them.

For AlRajhi Bank App a sensitive data shouldn't store on external storage it can be modified or read by other apps installed on the device, it is better to use internal storage is for save sensitive data to which other apps and users cannot access.

To avoid modifying or reading data in external storage, you should:

* Encrypt any sensitive data that it writes to external storage.
* Perform input validation on any data that it reads from external storage.
* Requires the user to grant their permission to access the resource, it is possible for apps to read and write the private files that belong to other apps after being granted permission by the user.
* **Webview Enables Content Access**

Android WebView is a system component that lets Android apps display web content inside them without opening a dedicated browser. Android WebView uses webkit engine to display web page.

WebView file access is enabled by default. If the WebViews take in untrusted input, this can allow for data theft. The app does not disable Content Provider access from WebViews.

For example, setAllowUniversalAccessFromFileURLs enabled for WebView this setting removes all same origin policy restrictions and allows the webview to make requests to the web from the file which is normally not possible. Because of that attacker can read local files and send them across the web.

If the WebView is exported, this behavior can be very dangerous because it can allow the attacker to read arbitrary files which may be private to the application.

* **Log file (CWE-532)**

log file is data file that contains information about usage patterns, activities, and operations within an application. Log files are especially useful for developers while analyzing data processing inside apps. Adding data to logs makes it easy for the developer to

make sure the results are correct and keeping track of crashes and errors, logs may also store sensitive data like passwords, names or numbers. It is dangerous because logs are stored locally on devices.

Using a log file in Al Rajhi Bank to record events, transfers, deposits, loans, customer and employee information, numbers of credit cards, passwords, messages, and communications between communication software applications.

Should be more careful about sensitive data about employees and customers, so that attackers do not eavesdrop on the information by hacking the log file and obtaining sensitive data.

Examples of sensitive data in AlRajhi Bank:

* **Names, email addresses, and phone numbers** are all these sensitive data that attackers can use by modifying it or deleted.
* **Personally Identifiable Information (PII):** that could be misused for identity theft Social Security numbers, credit card numbers, and bank account numbers.

When the app saves sensitive data in log files (like Authorization access, user login, user password) attackers can drop the database

and have access to all the needed information, so it will be easy to access sensitive data.

We can save sensitive data by Encryption keys but to encrypt any kind of data we need to use encryption keys. The big mistake when saved encryption keys on device. This is unacceptable because in fact it makes all the encryption efforts pointless, the attacker can decrypt the database of sensitive data and get everything he needs.

The solution for this problem uses tool KeyStore. This tool is part of Android security tools, it is designed for handling and storing encryption keys with most common and secure algorithms. Keystore

contains Keystore passwords to put passwords that are readable by the specific group and contains private and sensitive information

* **Dynamic Code Loading:**

After using AndroBugtool, shows the Al-Rajhi Bank app use Dynamic Code Loading**,** and show it high issue.

Dynamic code loading may be misuse by attackers, may load source code into a free library, and run malicious applications by loading dynamic code.

And the benefit of dynamic Code Loading allows an app developer to load additional code from free library at runtime, and Al-Rajhi bank App can use DCL to upgrade specific parts of app without letting the user know. Al-Rajhi bank App may show different content to a person depending on if they use the free or premium version, Dynamic code loading can display the correct content.

But there are some dangerous uses by dynamic code loading that may cause problems for Al-Rajhi bank. Dynamic code loading may

not cause a direct problem to the Al-Rajhi application, but the hacker could use a flaw in the Al-Rajhi Bank application to pull and steal the application's source code from free libraries that store some of the code.

The proposed solution to avoid hacking free libraries and obtaining the important codes for the Al Rajhi Bank application doesn’t save the important codes in the libraries.

* **Cleartext traffic (CWE-319)**

Is information that is stored or sent in an unencrypted form, and it lacks security, reliability, and protection against tampering. It means that an attacker able to see traffic from the network and can read, modify or corrupt the transported content.

There are some examples of Cleartext trafficissues**:**

* **No Encryption of Sensitive Data:** When the application does not encrypt sensitive information before storage or
* transmission, it is easy for an attacker to tamper with the information.
* **Transfer passwords in cleartext**: when the app transmits passwords in clear text, allowing attackers to take away the organization passwords.

Al-Rajhi app enable the cleartext, and it may cause many problems for the organization or customers, and one of the commune problems it causes during the transfer of the customer’s data, whether it is his accounts or sensitive information, when sent via clear text makes it easier for attackers to eavesdrop or modify information.

Some issues caused by cleartext traffic to the app:

When transferring sensitive data of organization or customers, the attackers may be eavesdrop to the information and modify or delete the sensitive data.

**Sensitive data for organization:​**

Sensitive information about the organization, whether it is loans or the number of employees and their personal information or password numbers for accounts, all of this information is sensitive information that harms the company when hacked by attackers.​

**Sensitive data for customer:​**

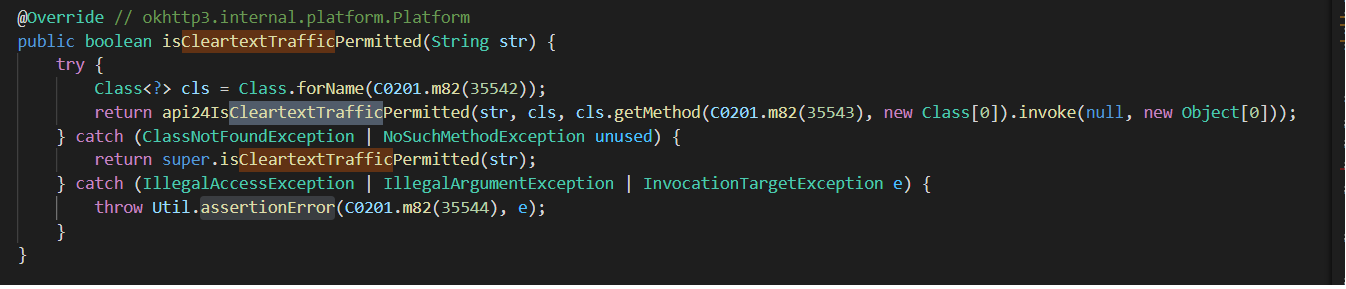
* Delete or modify the account number.​
* Taking the customer’s numbers, and the numbers can be taken advantage of by defrauding him through messages.
* Freezing customer account. ​

One of the proposed solutions when transferring sensitive data, whether it is to the customer or the organization, is to transfer it via protocols encrypted, reliable and difficult to hack. SSL is one of example for these protocols.

Secure Sockets Layer (SSL) is a protocol for transmitting private documents. SSL uses an encryption system that uses two keys to encrypt data a public key known to everyone, and a private key known only to the recipient of the message.

AlRajhi uses **isCleartextTrafficPermitted** which allows checking whether cleartext network traffic is permitted for all network communications from this process or not.

When cleartext network traffic is not permitted, the platform's components will refuse this process's requests to use cleartext traffic.



Path of the code: okhttp / internal / platform / AndroidPlatform.java

Here will return boolean value (true or false), whether cleartext network traffic (for example HTTP, DownloadManager, and MediaPlayer) is permitted for communicating with (str) or not.

DOCMENTATION

Code with good documentation is far more valuable than code without documentation. Good documentation makes the review go more smoothly and makes it more likely that the review will do its intended purpose. Good software documentation, helps programmers and testers understand the code quickly and save time.

Documentation helps software development to keeps software quality at high levels by makes it easy to transfer projects and make the code comments clean, readable, and familiar.

Steps for Check the documentation:

**Step 1: discover code**

will see all the functions in the code.

**Step 3: improve**

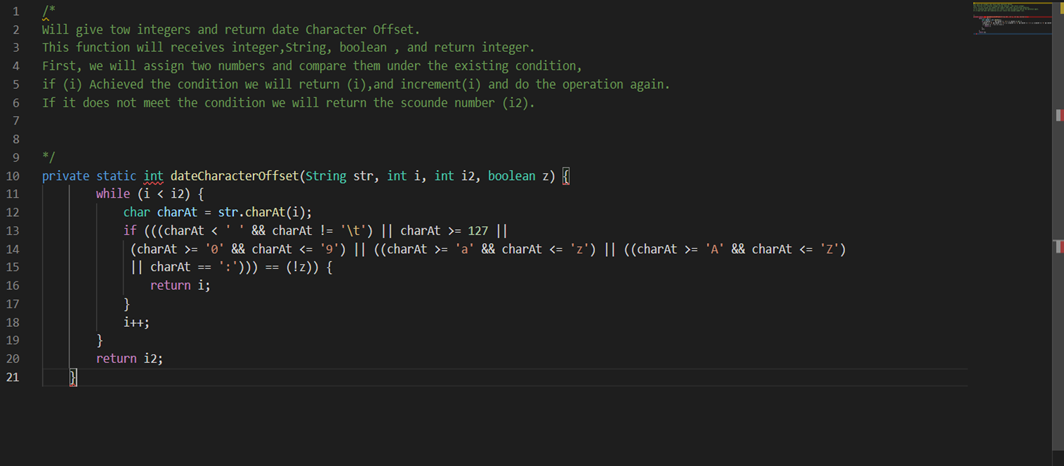
if the code is not documented, we will improve the code as possible to be documented.

**Step 2: check the documentation**

check if the code meets the documentation requirements in all its respects

In this class used letters instead of names, and this makes the class difficult to understand and deal with, moreover, there is no comment to explain the work of the class.

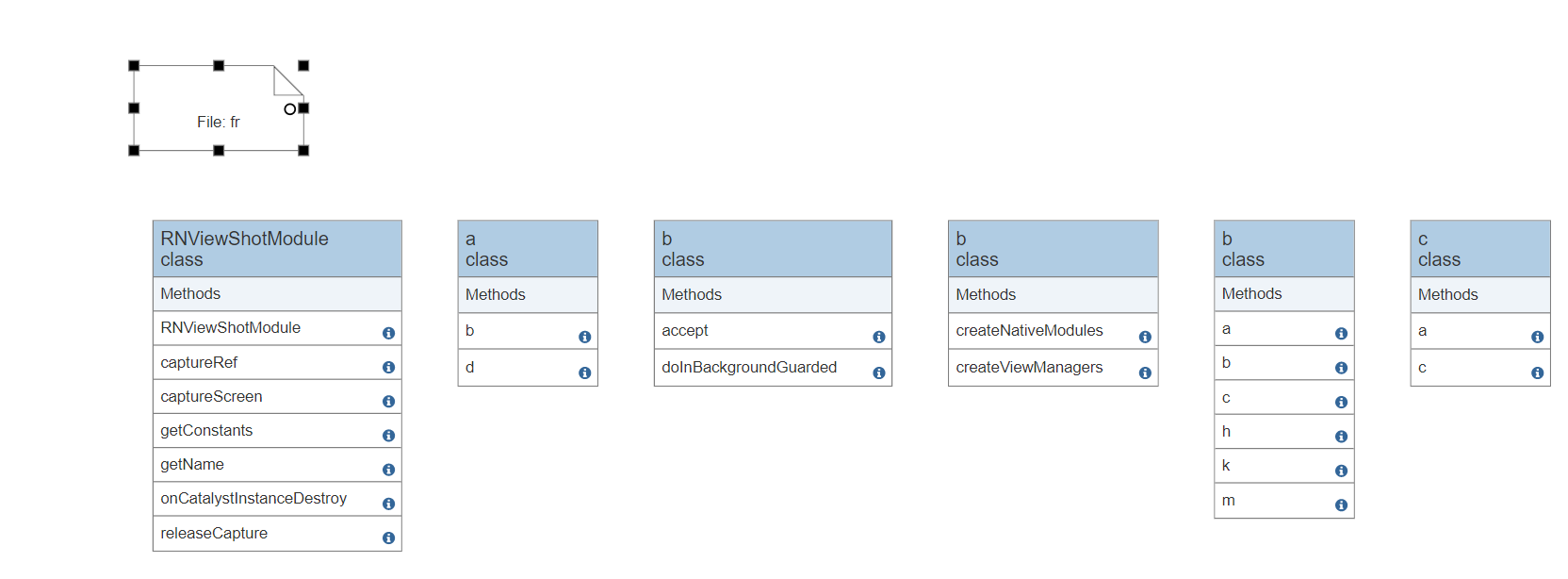


In this case, the class don't have documentation and I will improve it to be documented.

After reading the documentation, it becomes clear to us that the code works and understands it faster, and here lies the value of the documentation.

UML DIAGRAM

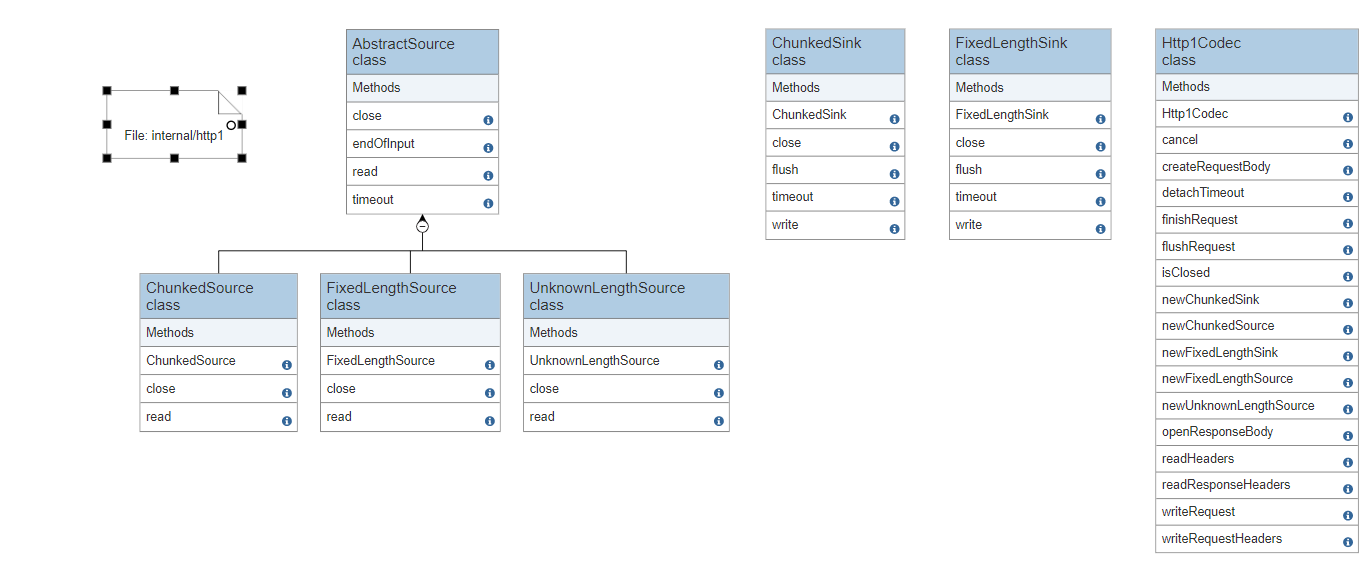
We use class diagram to explain data models for information systems, no matter how simple or complex the code. Better understand the general relationship between classes**.**



After see the code and implementation class diagrams on code, I see a lot of class extend(inheritance) form libraries, and the libraries have advantages and disadvantages. It is mistakes to depend on libraries because they may be hacked by attackers and may cause stored information to be lost and leaked, and it is very expensive and complex.

**Another example for Class Diagram:**

Here the class have inheritance from each other inside the class:



The problem here is that if the superclass has a problem or an error, this causes a problem to all its subclass and may stop working completely due to a problem in the superclass. And this problem called Single Point of Failure.

READABLE

Source code readability is a property that influences how easily a given code can be read and understood. Since source code readability can affect software quality, especially maintainability, then programmers must have a good sense of writing readable code.

Here are some practices for writing readable code:

* Commenting and Documentation
* Consistent Indentation
* coherent identifiers
* DRY Principle
* Meaningful identifiers

Steps we followed to get to analyze the application:

**Step 3: Download the application**

Download the Al-Rajhi application in the form of APK file

**Step 1: Choosing application**

Choose the bank application to be analyzed

**Step 4: Analyze**

Analyze the application and correct some errors

**Step 2: Download the tools**

Download the MobSF tool and its requirements to analyze the application

Source code has been used coherent indentation and coherent identifiers by choosing English language only and using CamelCase Identifiers style, in another hand they didn't use vertical alignment, for example:

Graphical user interface, text

Description automatically generated with medium confidence

Path: com\avishayil\rnrestart\ReactNativeRestart.java

Text

Description automatically generated

Path: com\agontuk\RNFusedLocation\RNFusedLocationModule.java

We noticed that source code application has a lot of abbreviations in the names, for example, writing the first letters of the word only, which can cause some errors in changing some variables or modifying them, for example:

صورة تحتوي على نص

تم إنشاء الوصف تلقائياً

Path: androidx\core\app\c.java

To avoid this from happening, it is better to **Use meaningful identifiers**, for example:

صورة تحتوي على نص

تم إنشاء الوصف تلقائياً

Conclusion

In this work shop, we worked on studying and analyzing the Al-Rajhi application to try to know the extent of the quality and safety of the application, We found some errors and problems whose risks range from high to low risks, which could cause penetration of the application and access to sensitive information, and then we worked on analyzing it and finding some solutions to reduce risks using security assessment tools such as (MobSF, Qark, AndroBug) that helped us reach the problems by understanding the report and source code, and through it we were able to know the percentage of protection and security of the application and determine its reliability.

With some problems that can be solved, Al-Rajhi application represents a high level of protection and security with the presence of data encryption that helps protect it from penetration.