Software Engineering (CS- 392)

Section: 171

Workshop: Code Quality [AlahliBank]

Supervisor: Dr.Sultan Alqahtani

Name	ID	Email
Ahmed Alshaalan	440017246	asialshaalan@sm.imamu.edu.sa
Ahmed Othman	440028238	aoaahmed@sm.imamu.edu.sa
Abdulrahman Alenizy	440024383	aomalenizy@sm.imamu.edu.sa
Sultan Almansour	440016632	skaalmansour@sm.imamu.edu.sa

Contents

1 Introduction

Code Quality determines whether the code is good or bad. In this workshop, we will apply the concepts of code quality through static analysis on a bank application and display the results of the analysis (design, safety, etc.) with a mention of the used tools.

2 Purpose

The main purpose of this workshop is to practice the concept of code quality and knowing how to use the tools required to perform static analysis to determine the quality of the code.

3 Goal

The goal of this workshop is to learn one of the most important topics of software engineering "code quality" and apply it in a practical way and gain new experience in the field of code quality and static analysis.

4 Tools

- Docker
- MobSF
- Smart Draw
- Python
- Git
- Qark

5 Steps

- Install Docker Desktop & WSL.
- Install & run MobSF by using Docker hub.
- Download APK for Al-Ahli Bank and use MobSF for Static Analysis.
- Getting report and source code that generated by MobSF too.l
- Used Smart Draw tool to draw class diagram automatically.
- Install Qark tool from GitHub by using git, run the tool and getting the report that generated by the tool.
- We used python script to calculate the number of opacity classes.
- There are some tools we tried to use it for static analysis but we facing problem with it, such as [Embold, Test Automation, SonarQube, AndroBugs].

5.1 Docker Desktop

We will user docker desktop to install Mobsf on windows or mac.

to Install Docker Desktop use the following link :

https://www.docker.com/products/docker-desktop

Docker Desktop required Linux Kernel Update "WSL", to install WSL use the following link:

https://docs.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package

5.2 MobSF

We will use this tool to performs static analysis and getting the source code, this tool required the apk file to do the analysis.

To Download MobSF from Docker Desktop select the appropriate image from the following link:

https://mobsf.github.io/docs/#/docker

after select the image use command line to download and run the tool.

5.3 APK file

We have two requirements for APK to meet the specification:

1. APK from a well-known website for integrity.
2. Getting the last up to date APK.

Finding the website that meets these two specifications was quite hard since these applications are sensitive, so there are a lot of fabricated APK for getting the information from the user, and since most people download APK from the original store (play store, IOS store) there is no benefit for a trusted website to pull APK from the original store and make it public.

So instead of finding websites, we download SNB Al-Ahli Mobile from the play store, then pull the APK from the phone using an APK extractor. With this, we can meet the two specifications that mention above and continue with well trusted APK.

5.3.1 Files walkthrough

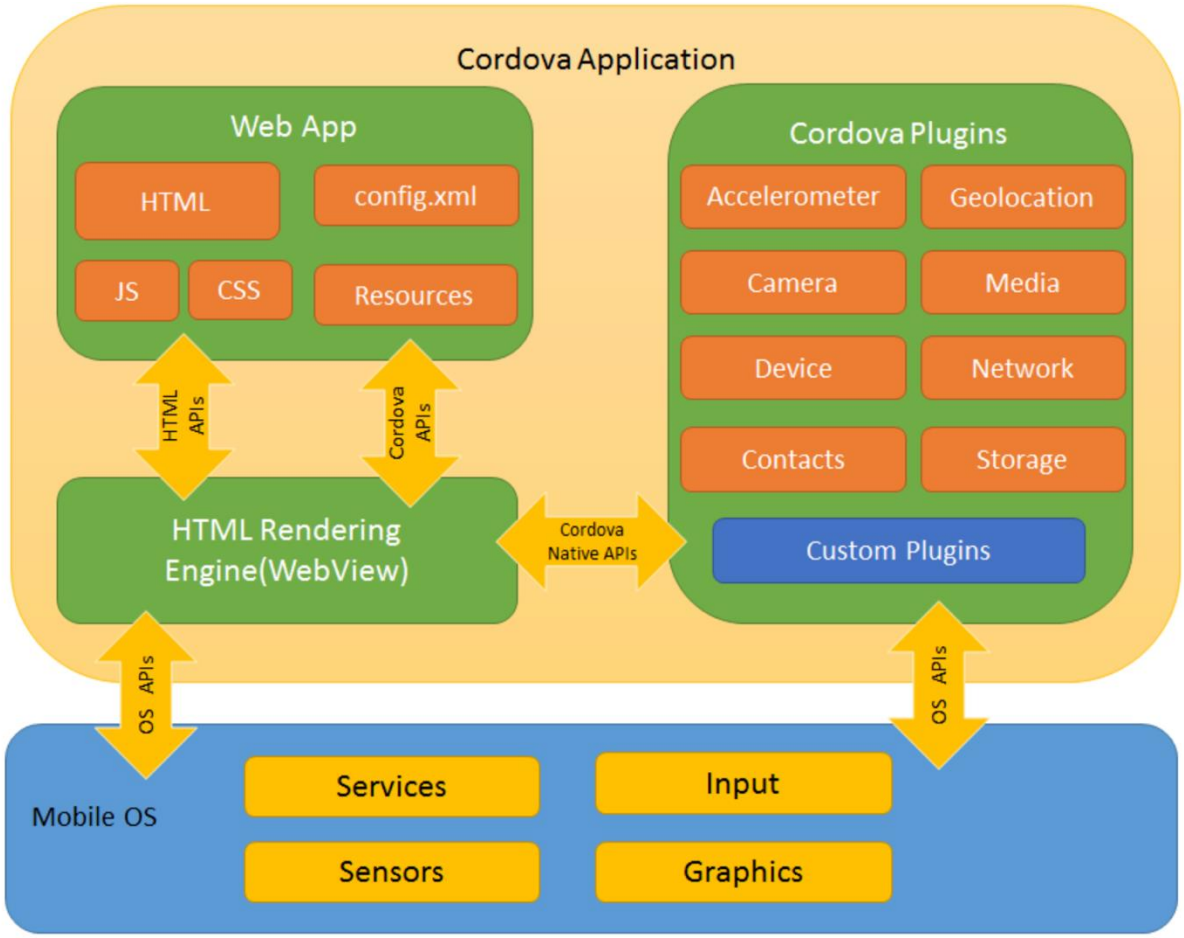
This is the structure of APK that contains the code, to understand how the application developed we need to figure out the purpose of these files. there are only two packages that matter: com, Cordova.com package contains some code of the application, while Cordova is the framework that is used to build the application, the other package is just an extension used for the framework, with android and androidx as exceptions, since these two packages work as API for the OS.

5.3.2 Apache Cordova

Apache Cordova is a framework that enables you to target multiple platforms with one codebase. By using HTML, CSS, and JavaScript, you can build an application that can work on the browser, android, and IOS with a little bit of configuration. Cordova also provides an API that is native for each platform to access different functionality (like camera, sensor, GPS). these APIs can be called in the JavaScript file.

5.3.3 So how does Cordova work?

Since HTML cannot work without an application to render the documents, Cordova provides an engine **WebView** that takes an HTML with the configuration file that has information about the plugin that is used in the application, and what platform that is targeted.



5.3.4 So why all the code is written in java?

Apache Cordova doesn't convert HTML, CSS, and JavaScript to a native language of the targeted platform, it just provides an engine to render and native plugins for the application. So, all the code that is extracted from MobSF is the code of the framework and the plugin that is used to make the application. And that is the reason we cannot find any code that is relevant for the domain problem.

So we need a way to extract the JavaScript code and the HTML, luckily we find a video on how to extract a source code of the Cordova app [made by Joshua Morony: https://www.youtube.com/watch?v=YJN29gIoY6k&ab_channel=JoshuaMorony]. All we need to do is to change the extension from .apk to .zip then unzip the file to get the source code and it's working!

5.4 Class Diagram

By using "Smart Draw" which is a diagram tool on website used to make class diagram, flowcharts, organization charts, mind maps, project charts, and other business visuals automatically.

- Complete tutorial can be watched on YouTube channel [made by Smart draw: https://youtu.be/cYoeqgmvI-M]

5.5 Qark tool

We will use this tool to look for serval security related Android application vulnerabilities.

to download and run this tool, Python language and Git must be exists, to install them use the following links:

Git: https://git-scm.com/downloads

Python: https://www.python.org/about/

after install and setup " Python & Git " start up "Git Bash" and use the following command "git clone https://github.com/linkedin/qark.git" to download the tool , after downloading complete, enter to qark folder "cd qark" and use this command to install tool requirements "pip install -r requirements.txt", after completing requirements use this command to set up the tool "python setup.py install", if setup done run the tool from this command "qark" and pass a source for scanning through either –java or –apk, then it will generate the security analysis report.

5.6 Obfuscate classes

Use the following python script to calculate number of obfuscate classes:

```python
import os
from typing import Tuple

PATH = "path_to_code/com/"


def main() -> Tuple[int, int]:
    number_of_files: int = 0
    number_of_obfuscate_files: int = 0
    for (root, dirs, files) in os.walk(PATH):
        file: str = ""
        for file in files:
            if ".java" not in file:
                continue
            number_of_files += 1
            file_name = file.split(".")[0]
            #  We assume that any file that is less than 5 character are obfuscate
            if len(file_name) < 5:
                number_of_obfuscate_files += 1

    return (number_of_files, number_of_obfuscate_files)


if __name__ == "__main__":
    files_numbers: Tuple[int, int] = main()
    print(f"Total number of classes {files_numbers[0]}")
    print(f"Total number of obfuscate classes {files_numbers[1]}")
    print(
        f"precentage of obfuscate {round(files_numbers[1]/files_numbers[0], 1) * 100}%"
    )
```

5.7 Problem with Embold

We encountered the problem that it asks to upload files in GitHub and the possibility of causing us a security problem as the code is for a large company, we do not have the right to raise it and we did not raise it.

5.8 Problem with Test Automation

This tool allows testing the code, but we can't run it because of the degree of code Obfuscate, so we couldn't perform the performance.

5.9 Problem with SonarQube

This tool requires compiling the code to do the analysis, and because of the obfuscate of the classes, the code cannot be compiled, also there are some libraries used in the classes that have been obfuscate and we cannot access them.

5.10 Problem with AndroBugs

The tool didn't work properly because there is some problem with the tool code implementation.

6 Source Code Quality

6.1 Code obfuscate

By using python script [see page:7] we figure that:

Total number of classes: 1877

Number of obfuscate classes: 925

So, obfuscate classes represent 49% of classes

6.2 Code Structure

- Code use meaningful identifiers.
- Code use coherent indentation.
- Code use coherent identifiers both variable style "CamelCase Identifiers" and "underscores identifiers", it should use one variable style not both.
- Code use standard code structure.
- Code is explicit.

```
public class MainActivity extends CordovaActivity {
    public static final String LOG_TAG = "CordovaActivity";
    public static final double screenDiagonalTabletMin = 6.9d;

    @Override // org.apache.cordova.CordovaActivity
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        if (Build.VERSION.SDK_INT >= 26) {
            disableAutoFill();
        }
        Bundle extras = getIntent().getExtras();
        if (extras != null && extras.getBoolean(PushConstants.START_IN_BACKGROUND, false)) {
            moveTaskToBack(true);
        }
        if (extras != null) {
            sendExtrasToPlugin(extras);
        }
        Uri data = getIntent().getData();
        if (data != null && data.isHierarchical()) {
            AppLinks.sendDataToCallback(getIntent().getDataString());
        }
        StringBuilder sb = new StringBuilder();
        sb.append(this.launchUrl);
        sb.append("?deviceType=");
        sb.append(isTablet() ? "tablet" : "phone");
        loadUrl(sb.toString());
    }
```

- Code doesn’t use vertical alignment.

```
public static final class anim {
    public static final int abc_fade_in = 2130771968;
    public static final int abc_fade_out = 2130771969;
    public static final int abc_slide_in_top = 2130771975;
    public static final int abc_slide_out_top = 2130771977;
```

- Code doesn’t use proper data structures.

```
public static final class anim {
    public static final int abc_fade_in = 2130771968;
    public static final int abc_fade_out = 2130771969;
    public static final int abc_slide_in_top = 2130771975;
    public static final int abc_slide_out_top = 2130771977;
```

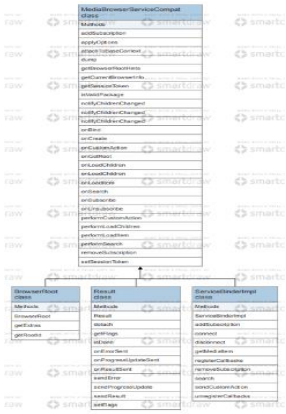
6.3 Code Documentation

We could not find any documents that we could mention in the report. We also tried to find and use tools to extract documents, but we didn't get any to help us with this part, and that is for two main reasons. the first one is the storage since there will be documentation the size of the application will increase. the second reason is security if there is documentation our application won't be secure.

6.4 Design principles

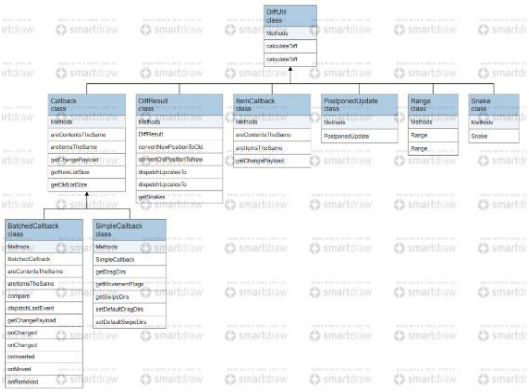
We can figure the design principle from the class diagram that we generated by using Smart Draw [see page:7]:

- Low cohesive.



As we see the classes take too many computation responsibilities, hence it is low cohesive.

- Tight coupling.



As we see the classes it tightly coupled with each other then it is tight-coupling.

6.5 Design Pattern

Some design patterns are used such as:

- Adapter Design Pattern.
- Strategy Design Pattern.
- Observer Design Pattern.

On the following classes: { CompositeGeneratedAdaptersObserver.java, FocusStrategy.java, CursorAdapter.java, MyDataSetObserver.java }

7 Static Analysis

7.1 Security Analysis

We come up with these issues by Qark tool

- WARNING Logging

Vulnerability: logs are detected, this may allow potential leakage of information from Android applications.

To validate this vulnerability: Logs should never be compiled into an application except during development.

- WARNING Webview enables file access

Vulnerability: File system access is enabled in this WebView. If untrusted data is used to specify the URL opened by this WebView, a malicious app or site may be able to read your app's private files, if it returns the response to them.

To validate this vulnerability: load the following local file in this WebView: html/FILE_SYS_WARN.html

- VULNERABILITY Dynamic broadcast receiver found

Vulnerability: Application that register a broadcast receiver dynamically is vulnerable to granting unrestricted access to the broadcast receiver. The receiver will be called with any broadcast Intent that matches filter. (android.content.BroadcastReceiver, android.content.IntentFilter)

- WARNING Webview enables universal access for JavaScript

Vulnerability: JavaScript running in a file scheme context can access content from any origin. This is an insecure default value for minSdkVersion < 16 or may have been overridden (setAllowUniversalAccessFromFileURLs) in later versions.

To validate this vulnerability: load the following local file in this WebView: html/UNIV_FILE_WARNING.html

- WARNING Insecure functions found

Vulnerability: The Content provider API provides a method call. The framework does no permission checking on this entry into the content provider besides the basic ability for the application to get access to the provider at all.

To validate this vulnerability: Any implementation of this method must do its own permission checks on incoming calls to make sure they are allowed. Failure to do so will allow unauthorized components to interact with the content provider.

- WARNING Potientially vulnerable check permission function called

Vulnerability: Be careful with use of Check permission function App maybe vulnerable to Privilege escalation or Confused Deputy Attack. This function can grant access to malicious application, lacking the appropriate permission, by assuming your applications permissions. This means a malicious application, without appropriate permissions, can bypass its permission.

To validate this vulnerability: check by using your application permission to get access to, otherwise denied resources. Use - checkCallingPermission instead.

- WARNING Broadcast sent without receiverPermission

Vulnerability: A broadcast, send Broadcast which does not specify the receiver Permission. This means any application on the device can receive this broadcast.

To validate this vulnerability: You should investigate this for potential data leakage.

- WARNING Webview enables content access

Vulnerability: While not a vulnerability by itself, it appears this app does not explicitly disable Content Provider access from WebViews. If the WebViews take in untrusted input, this can allow for data theft.

To validate this vulnerability: load the following local file in this WebView: html/WV_CPA_WARNING.html

- INFO Hardcoded HTTP url found

Vulnerability: Application contains hardcoded HTTP url: http://localhost/, this request can be intercepted and modified by a man-in-the-middle attack.

To validate this vulnerability: implement HSTS

- VULNERABILITY Empty pending intent found

Vulnerability: A malicious application could potentially intercept, redirect and/or modify this Intent. Pending Intents retain the UID of your application and all related permissions, allowing another application to act as yours.

To validate this vulnerability: the Intent you supply should almost always be an explicit intent that is specify an explicit component to be delivered to through Intent.setClass.

- WARNING External storage used

Vulnerability: Reading files stored on {storage_location} makes it vulnerable to data injection attacks. Note that this code does no error checking and there is no security enforced with these files. For example, any application holding WRITE_EXTERNAL_STORAGE can write to these files.

- WARNING Broadcast sent with receiverPermission with minimum SDK under 21

Vulnerability: A broadcast, sendBroadcast which specifies the receiverPermission, but may still be vulnerable to interception, due to the permission squatting vulnerability in API levels before 21. This means any application, installed prior to the expected receiver(s) on the device can potentially receive this broadcast.

To validate this vulnerability: You should investigate this for potential data leakage.

- WARNING Javascript enabled in Webview

Vulnerability: While not a vulnerability by itself, it appears this app has JavaScript enabled in the WebView: If this is not expressly necessary, you should disable it, to prevent the possibility of XSS (cross-site scripting) attacks.

To validate this vulnerability: load the following local file in this WebView:

html/JS_WARNING.html

- WARNING Webview uses addJavascriptInterface pre-API 17

Vulnerability: This webview uses the "add Java script Interface" method in a pre-API 17 app, which exposes all public methods to Java script running in the WebView. If this webview loads untrusted content or trusted content over plain-text HTTP, this represents a MAJOR issue!

To validate this vulnerability: load the following local file in this WebView:

html/BAD_JS_INT.html

- WARNING Webview enables DOM Storage

Vulnerability: DOM Storage enabled for this WebView, there is a potential for caching sensitive information.

- INFO Phone number or IMEI detected

Vulnerability: Access of phone number or IMEI, is detected. Avoid storing or transmitting this data.

- WARNING Remote debugging enabled in Webview

Vulnerability: Enabling webview remote debugging is insecure.

7.2 Manifest Analysis

We come up with these issues by MobSF tool

- Service protected (High severity):

(com.google.android.gms.auth.api.signin.RevocationBoundService) Protected by a permission, but the protection level of the permission should be checked.

permission:
com.google.android.gms.auth.api.signin.permission.REVOCATION_NOTIFI CATION
[android:exported=true]

- Service not protected (High severity):

There are some services that are not protected such as
(com.innofis.cordova.richpush.FCMService) ,
(com.innofis.cordova.richpush.PushInstanceIDListenerService),
(com.google.firebase.messaging.FirebaseMessagingService),
(com.google.firebase.iid.FirebaseInstanceIdService) and
(com.huawei.hms.support.api.push.service.HmsMsgService)
Which will be shared with other applications on the same device and could be accessed by any other application. So client data wont be secure.

- (broadcast receiver) protection level checking (High severity):

The broadcast receiver in Al-Ahlibank which is (com.google.firebase.iid.FirebaseInstanceIdReceiver) is always protected by a permission but the problem is that the protection level of the permission should be checked.

Permission levels are classified as follow:

a) **Normal protection level:** This type of permission is given when there is no risk to user's privacy or the operation of other apps. In such kind of permission, no permission granting dialog is prompt. User cannot cancel this kind of permission.
b) **Signature Protection Level:** The system allows these app permissions at install time, but only when the app that attempts to use permission is signed by the same certificate as the app that defines the permission.
c) **Dangerous Protection Level:** Dangerous permission is used when where the app wants the data and resources that involve the user's private information. Dialog of this permission is to be accepted by the user unless the app cannot provide functionality that depends on that permissionIn this part the Broadcast Receiver is found to be shared with other apps on the device which can case harm to the application. It is protected by a permission which is not defined in the analyzed application. As a result, the protection level of the permission should be checked where it is defined. If it is set to normal or dangerous, a malicious application can request and obtain the permission and interact with the component. If it is set to signature,only applications signed with the same certificate can obtain the permission. Permission: com.google.android.c2dm.permission.SEND.

- (broadcast receiver) protection level checking (info severity):

The Broadcast Receiver (com.huawei.hms.support.api.push.PushMsgReceiver) is Protected by a permission, but the protection level of the permission should be checked however, in the analysis report the broadcast receiver is found to be exported, but is protected by a permission. However, the protection level of the permission is set to signatureOrSystem. Instead, it is suggested that signature level be used. For most purposes, the signature level should suffice, and it is independent of where the applications are installed on the device. Permission: com.alahli.mobile.android.permission.PROCESS_PUSH_MSG. protectionLevel: signatureOrSystem.

- (Broadcast Receiver) is not Protected(High severity):

(nl.xservices.plugins.ShareChooserPendingIntent) Finding a broadcast receiver that is shared with other applications and thus leaves it handy to any other application on a device Having an intent filter indicates that the broadcast receiver is explicitly exported

- (Content provider) permission level checking (info severity):

Content Provider (com.huawei.hms.support.api.push.PushProvider) is Protected by a permission, but the protection level of the permission should be checked.The export of a Content Provider is discovered, but it is protected by a permission. The permission's protection level, however, is set to signatureOrSystem. Instead, it is suggested that signature level be employed. For most purposes, the signature level should suffice, and it is independent of where the programs are placed on the device. Permission: com.alahli.mobile.android.permission.PUSH_PROVIDER protectionLevel: signatureOrSystem.

- App has a Network Security Configuration(info severity):

(android:networkSecurityConfig=@xml/network_security_config) Configure Network Security The feature allows applications to customize network security settings files in a secure location, and there is no modification to the application code.

- Activity is not Protected. (High severity):

(de.niklasmerz.cordova.biometric.BiometricActivity)and(com.exxbrain.android.biometric.DeviceCredentialHandlerActivity) Find activity to share with other apps on devices so leave it in the hands of other apps.

- Activity is Protected. (High severity):

(com.innofis.cordova.richpush.PushHandlerActivity) Protected by a permission, but the protection level of the permission should be checked. Find activity to share with other apps on devices so leave it in the hands of other apps and be protected so that permission is requested and dealt with and whoever gets the signature can get the permission. Permission: (com.alahli.mobile.android.permission.PushHandlerActivity) [android:exported=true]

7.3 Code Analysis

- CWE-200 Information Exposure(Severity: Warning):

The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.

There are many different kinds of mistakes that introduce information exposures. The severity of the error can range widely, depending on the context in which the product operates, the type of sensitive information that is revealed, and the benefits it may provide to an attacker.

Solution: Typically, issues such as these should be resolved on the server side. However, if necessary the following Content Rule on the LoadMaster can be used to block any internal IP addresses from being exposed.

- CWE-312: Cleartext Storage of Sensitive Information (Severity: Warning):

The application saves sensitive data in cleartext in a resource that might be accessed by a different organization.

Because the data is saved in cleartext, attackers may be able to read it. Even if the data is encrypted in a non-human readable format, some methods could be analyzed to find out which encryption is being used and then decode the data.

- CWE-649: Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking(Severity: High):

The software uses obfuscation or encryption of inputs that should not be mutable by an external actor, but the software does not use integrity checks to detect if those inputs have been modified.

When an application relies on obfuscation or incorrectly applied / weak encryption to protect client-controllable tokens or parameters, that may have an effect on the user state, system state, or some decision made on the server. Without protecting the tokens/parameters for integrity, the application is vulnerable to an attack where an adversary traverses the space of possible values of the said token/parameter in order to attempt to gain an advantage. The goal of the attacker is to find another admissible value that will somehow elevate their privileges in the system, disclose information or change the behavior of the system in some way beneficial to the attacker. If the application does not protect these critical tokens/parameters for integrity, it will not be able to determine that these values have been tampered with. Measures that are used to protect data for confidentiality should not be relied upon to provide the integrity service.

- CWE-250: Execution with Unnecessary Privileges(Severity: High):

Because running with elevated rights, such as root or Administrator, can remove the operating system's or environment's standard security checks, new vulnerabilities can develop. Other defects can become security vulnerabilities if they happen while working with elevated privileges.

Privilege management services might act in unexpected ways, and they have various characteristics depending on the platform. When switching from one non-root user to another, these errors are most noticeable. Signal handlers and spawned processes run with the privileges of the owner process, therefore if a process is running as root when a signal fires or a sub-process is executed, the signal handler or sub-process will run as root.

- CWE-330: Use of Insufficiently Random Values (Severity: Warning):

In a security situation that relies on unpredictable numbers, the software employs insufficiently random numbers or values.

When software generates predictable values in an environment that requires unpredictability, an attacker may be able to guess the next value that will be generated and use that guess to impersonate another user or get access to sensitive data.

For example, This code attempts to generate a unique random identifier for a user's session.

```
function generateSessionID($userID) {

 srand($userID);

  return rand;() }
```

The session ID will always be the same because the seed for the PRNG (Pseudo-Random Number Generator) is always the user's ID. As a result, an attacker may possibly guess any user's session ID and hijack the session.

Computers are predictable machines, which means they can't generate actual randomness. PRNGs use an algorithm to approximate randomness, starting with a seed from which all values are computed.

PRNGs are classified as statistical or cryptographic. Statistical PRNGs have useful statistical qualities, but their output is extremely predictable and forms an easy to duplicate numeric stream, making them unacceptable for application in situations where generated numbers must be surprising. This problem is addressed by cryptographic PRNGs, which generate output that is more difficult to predict. To be cryptographically safe, a value must be impossible or extremely difficult for an attacker to differentiate from a truly random number.

- CWE-327: Use of a Broken or Risky Cryptographic Algorithm (Warning)

The use of a weak or dangerous cryptographic algorithm is an unnecessary risk that could lead to sensitive data being exposed. The use of a non-standard algorithm is risky because a skilled attacker may be able to decode the algorithm and gain access to the data being protected. It's possible that well-known ways exist to defeat the algorithm.

7.4 APIs

Error in the APIs call should not give any information related to the codebase, to prevent any attempt to find any useful information about the codebase. We found some APIs that show pieces of information about the error that happened in the server:

- https://www.alahlimobile.com/api/public/v1/commons/
- https://www.alahlimobile.com/api/public/v1/applications/
- https://www.alahlimobile.com/api/public/v1/promotions/

The error that we got from the server was:

“Error 500: org.springframework.web.util.NestedServletException: Request processing failed; nested exception is java.lang.NullPointerException”.

All these APIs are public APIs that we think cannot do any harm to the domain problem (accounting, security). The important APIs are hidden using environment variables.

7.5 Tokens

We found some hardcoding tokens that may be dangerous to expose in public:

- "google_api_key":"AIzaSyBtG3OBvZGqRas0sR3H8Fpaz4U_DZ1KHlo"
- "google_crash_reporting_api_key":"AIzaSyBtG3OBvZGqRas0sR3H8Fpaz4U_DZ1KHlo"

This key only identifies the project in the Firebase platform, there is no risk in making it public since it cannot do anything other than identify the projects as stated by one of firebase engineering in this post.

- “token=df6586d97d3ecc025149faa487bf2629”

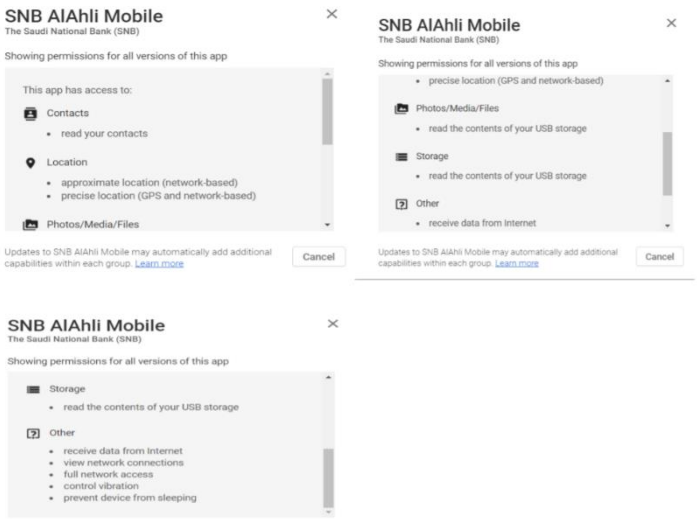
This token is used as a parameter in calling API, we don’t know the usage of this token, so we cannot know if it is dangerous or not.

8 Permissions

Application Permissions: App permissions on Android can allow apps access to your phone's camera, microphone, private messages, conversations, images, and more. App permission requests appear when an app wants access to sensitive technology or data on your phone or tablet for the first time and are usually related to privacy.

Permission types: Permissions on Android are divided into three categories: install-time permissions, runtime permissions, and special permissions. When the system grants your app that permission, the type of that permission indicates the extent of limited data that your app can access and the scope of restricted actions that your app can execute.

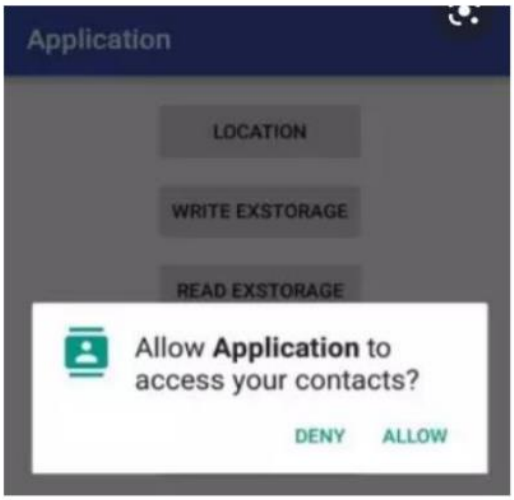
A. **Install-time permissions:** Install-time permissions provide your app with limited access to restricted data and allow it to perform limited activities that have minimal impact on the system or other apps. When you specify install-time permissions in your app, the system grants certain permissions to your app when the user installs it. When a user visits an app's details page, the store displays an install time permission alert, as seen in Figures below (The images have been taken from google play store Al-Ahli bank-permissions section).



We also can classify the Install-time permissions into two types:

1. **Normal permissions:** These permissions give your program access to data and activities outside of its sandbox. The data and activities, on the other hand, pose relatively minimal risk to the user's privacy and the functionality of other apps.
2. **Signature permissions:** If an app declares a signature permission that another app has defined, and both applications are signed by the same certificate, the system grants the permission to the first app after installation. Otherwise, the permission will not be provided to the first app.

B. **Runtime permissions:** Runtime permissions, also known as dangerous permissions, give your app more access to restricted data and let it to take actions that have a greater impact on the system and other apps. As a result, before you can access restricted data or perform restricted actions, you must first obtain runtime rights in your app. The system displays a runtime permission dialog when your program seeks a runtime permission, as seen in Figure.



Many runtime permissions have access to private user data, which is a type of restricted data that can contain potentially sensitive data. Location and contact information are examples of private user data.

C. **Special permissions:** Special permissions are assigned to specific app functions. Special permissions can only be defined by the platform and OEMs (Original Equipment Manufacturers). Additionally, when the platform and OEMs wish to protect access to extremely strong operations, such as drawing over other apps, they normally establish special permissions. A variety of user-toggleable operations can be found on the Special app access page in system settings. Special permissions are used to implement several of these tasks.

8.1 Permissions on Al-Ahli bank

A. Dangerous Permission

1. Read External Storage(android.permission.READ_EXTERNAL_STORAGE).

permissions are always dangerous permissions because they can access the shared external storage. Full read and write access to any location of the volume is protected by two permissions marked as dangerous: READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE. When an app is given storage permission, it has unlimited access to the device's storage. This means it can upload personal files or even remove sensitive data from the device, thus it's recommended not to provide untrusted apps storage permission because it can be dangerous.

2. Read contact(android.permission.READ_CONTACTS).

Allows an application to read all your phone's contact (address) info. This can be used by malicious applications to transfer your data to third parties. However, since the application is not a social media app or calling app it is probably not safe to give this application access to you contact.

3. Access fine location(android.permission.ACCESS_FINE_LOCATION).

Allows the API to determine a position as precisely as possible using available location providers such as the Global Positioning System (GPS), WiFi, and mobile cell data.

4. Access coarse location (android.permission.ACCESS_COARSE_LOCATION).

Allows the API to identify the device's position via WiFi or mobile cell data (or both). The API returns a location with a precision of about a city block.

B. **Normal Permission**

1. Internet(android.permission.INTERNET).

One of the most used permissions for connecting apps to the internet. The "Full Network Access" permission (used by 83 percent of apps) allows an app to connect to any network the device is connected to at the time, whereas the "View Network Connections" permission (used by 69 percent of apps) allows an app to view what networks the device has access to. Any app that requires internet connection to function correctly will require one or both of these permissions. While these two permissions are nearly universal, they do not allow their related apps to directly access any user information.

2. Network state (android.permission.ACCESS_NETWORK_STATE).

Allows an application to view the status of all networks. is needed for accessing ConnectivityManager (mainly for monitoring network connections in general). It is required to check if you are connected to a network, it does not matter of what type it is (Wi-Fi, GPRS, etc.).

3. Use biometric (android.permission.USE_BIOMETRIC).

Allows an app to use device supported biometric modalities. Biometrics are a faster, but potentially less secure, way of verifying your identity with a gadget. Primary authentication (knowledge-factor based modalities such as PIN, pattern, password, and fingerprint) provides the highest level of protection in the tiered authentication scheme. Biometrics are a second-tier authentication method that provides a good blend of convenience and security.

4. Wake lock(android.permission.WAKE_LOCK).

A wake lock is a mechanism to indicate that your application needs to have the device stay on. It's useful when you need to accomplish something even though the device appears to be sleeping, such as downloading data from the internet. Wake locks should only be used when necessary. The reason for this is that they use more battery, and if you have a bug that prevents them from being released when needed, your program will continue to drain the device's battery.

5. Vibration(android.permission.VIBRATE).

This permission will control the vibrating system on your phone. Which could be helpful in the notification massages you received from the application and in the transaction, you made in the app. This permission will be normal since there will be no security harm or data breach.

8.2 Deal with permissions problem

Use permissions

Because Android sandboxes applications from each other, applications must explicitly share resources and data. They do this by declaring the permissions they need for additional capabilities not provided by the basic sandbox, including access to device features such as the camera.

Request permissions

You should minimize the number of permissions that your app requests. Restricting access to sensitive permissions reduces the risk of inadvertently misusing those permissions, improves user adoption, and makes your app less vulnerable for attackers. Generally, if a permission is not required for your app to function, don't request it. If there is a feature that the app can't run without, declare it using a <uses-feature> element in the manifest file.

If it's possible to design your application in a way that does not require any permissions, that is preferable.

Create permissions

Generally, you should strive to define as few permissions as possible while satisfying your security requirements. Creating a new permission is relatively uncommon for most applications, because the system-defined permissions cover many situations. Where appropriate, perform access checks using existing permissions.

If you must create a new permission, consider whether you can accomplish your task with a signature protection level. Signature permissions are transparent to the user and allow access only by applications signed by the same developer as the application performing the permission check.

9 Storage

Because, the most common security concern for an application on Android is whether the data that you save on the device is accessible to other apps.

There are fundamentals ways to save data on the device:

a) Internal storage.
b) External storage.

The following paragraphs describe the security issues associated with each approach.

a) Use internal storage

By default, files that you create on internal storage are accessible only to your app. Android implements this protection, and it's sufficient for most applications.

To provide protection for sensitive data, you can encrypt local files using the Security library. This measure can provide protection for a lost device without file system encryption.

b) Use external storage

Files created on external storage, such as SD cards, are globally readable and writable. Because external storage can be removed by the user and also modified by any application, don't store sensitive information using external storage.

To read and write files on external storage in a more secure way, consider using the Security library, which provides the Encrypted File class.

10 Broadcasts

Broadcast: android apps can send or receive broadcast messages from the Android system and other Android apps, similar to the publish-subscribe design pattern. These broadcasts are sent when an event of interest occurs.

Receiving broadcasts: Apps can receive broadcasts in two ways: through manifest-declared receivers and context-registered receivers.

If you declare a broadcast receiver in your manifest, the system launches your app when the broadcast is sent.

To declare a broadcast receiver in the manifest, perform the following steps:

1. Specify the <receiver> element in your app's manifest.

```
<receiver android:name=".MyBroadcastReceiver"  android:exported="true">
  <intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED"/>
    <action android:name="android.intent.action.INPUT_METHOD_CHANGED" />
  </intent-filter>
</receiver>
```

The intent filters specify the broadcast actions your receiver subscribes to.

2. Subclass BroadcastReceiver and implement onReceive(Context, Intent). The broadcast receiver in the following example logs and displays the contents of the broadcast:

```
public class MyBroadcastReceiver extends BroadcastReceiver {
    private static final String TAG = "MyBroadcastReceiver";
    @Override
    public void onReceive(Context context, Intent intent) {
      StringBuilder sb = new StringBuilder();
      sb.append("Action: " + intent.getAction() + "\n");
      sb.append("URI: " + intent.toUri(Intent.URI_INTENT_SCHEME).toString() + "\n");
      String log = sb.toString();
      Log.d(TAG, log);
      Toast.makeText(context, log, Toast.LENGTH_LONG).show();
    }
  }
```

The system package manager registers the receiver when the app is installed. The receiver then becomes a separate entry point into your app which means that the system can start the app and deliver the broadcast if the app is not currently running.

The system creates a new BroadcastReceiver component object to handle each broadcast that it receives. This object is valid only for the duration of the call to onReceive(Context, Intent). Once your code returns from this method, the system considers the component no longer active.

11 Conclusion

In this project, we managed to know how to use MobSF tool, SmartDraw, and many other tools and how to extract the data from these tools and use this data to get the information we need. This project helps us to understand the source code and how the source code quality is important. It also increases our knowledge about design principles such as cohesion and coupling principles. However, during this project, we encountered many problems and were able to solve them. Finally, this project helped us to get new knowledge about software engineering and improve ourselves in this area.

References

- Android Dev Summit 2021: https://developer.android.com/guide/components/broadcasts#java
- Codegrip: https://www.codegrip.tech/productivity/what-is-code-quality-how-to-measure-and-improve-it/
- SmartDraw: https://www.smartdraw.com/
- MobSF Documentation: https://mobsf.github.io/docs/#/updating
- Docker: https://www.docker.com/products/docker-desktop
- Stack Overflow: https://stackoverflow.com/
- Manual installation steps for older versions of WSL: https://docs.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package
- Python: https://www.python.org/downloads/
- Git: https://git-scm.com/downloads
- APK downloader: https://apps.evozi.com/apk-downloader/
- Youtube channel, Joshua Morony: https://www.youtube.com/channel/UCbVZdLngJH6KOJvpAOO3qTw
- Qark: https://github.com/linkedin/qark
- Free-For.dev https://free-for.dev/#/