



Al-Imam University  
Computer Science Department  
College of Computer and Information Sciences



## Workshop 1

SABB Mobile (3.4.0)



### Teams Members

**NAME**

**ID**

**Ghada Al-Harbi**

**440021008**

**Norah Al-Marul**

**440021196**

**Raghad Al-Fares**

**440020499**

**Shatha Al-Suwaid**

**440023707**

Instructor: Dr. Lamees Alhazaa  
Course: CS392  
Section: 372  
Thursday, November 4, 2021

## Index

1. Project Description.....	3
1.1 Introduction.....	3
1.2 Lessons learned.....	3
1.3 Problems .....	3
1.4 References.....	3
2. Mobile App analysis results.....	4
3. Analysis results .....	5
3.1 Code analysis.....	5
3.2 Code structure.....	6
3.2.1 Documentation analysis.....	6
3.2.2 Meaningful identifiers.....	6
3.2.3 Vertical alignment .....	7
3.3 Design .....	7
3.3.1 Design principle .....	7
3.3.1.1 Low coupling.....	7
3.3.2 Security design .....	8
3.3.2.1 Documentation.....	8
3.3.2.2 Meaningful identifiers.....	8
3.3.2.3 Distribution.....	8
4. Discussion about results.....	9
4.1 Code analysis.....	9
4.2 Code structure.....	9
5. Conclusion .....	10

# 1 Project Description

## 1.1 Introduction

The purpose of this workshop is to present studies on the Saudi British Bank (SABB) and evaluate the quality of its application from: source code and its structure, security, its complexity, its services to the beneficiary.

## 1.2 Lessons learned

- 1.2.1 Gain the skills of analysis mobile app data (codes, docs, designs, etc.).
- 1.2.2 Increase knowledge about Docker github.
- 1.2.3 Understand how to extract mobile app data analyses.
- 1.2.4 Learn about Common Vulnerability Scoring System (CVSS).

## 1.3 Problems

We face many problems during project:

- 1.3.1 Initially downloading The MobSF with no docker version.
- 1.3.2 The size of the program was large and some of us couldn't install it.
- 1.3.3 The selected application to analysis required the android version and the experience of the members with android was weak.

## 1.4 References

Website Name	Website Link
--------------	--------------

<b>Docker</b>	<a href="https://www.docker.com/">https://www.docker.com/</a>
<b>github</b>	<a href="https://github.com/">https://github.com/</a>
<b>Wikipedia</b>	<a href="https://en.wikipedia.org/wiki/Common_Vulnerability_Scoring_System">https://en.wikipedia.org/wiki/Common_Vulnerability_Scoring_System</a>

## 2 Mobile App analysis results

Steps that we follow for extract mobile app data:

STEPS	DESCRIPTION	PREDETERMINED DATE
STEP 1	Install docker.	3/10/2021
STEP 2	Install MobSF and learn how to use it.	3/10/2021
STEP 3	Extract mobile app (codes, docs, designs).	7/10/2021
STEP 4	Learn about CVSS and its ratings.	7/10/2021
STEP 5	Analysis the code	14/10/2021
STEP 6	Analysis the code structure	15/10/2021
STEP 7	Identify the pros and cons of the code design	19/10/2021

### 3 Analysis results

#### 3.1 Code analysis

NO.	ISSUE	STANDARD	DESCRIPTION
1	The app logs information.	CVSS V2: 7.5 (high) CWE: CWE-532	The app stores user information in log files which makes it vulnerable to attack from hackers and viruses.
2	Files may contain hardcoded sensitive information like (username, password, keys etc.).	CVSS V2: 7.4 (high) CWE: CWE-312	Sometimes memory analysis enables hackers to access sensitive information, so if the application does not use the encryption algorithm, the hacker can determine the username and password of the user.
3	The App uses an insecure Random Number Generator.	CVSS V2: 7.5 (high) CWE: CWE-330	The RNG algorithm was not used to generate fake numbers to encrypt the user's real numbers.
4	MD5 is a weak hash known to have hash collisions.	CVSS V2: 7.4 (high) CWE: CWE-327	This application uses a weak encryption algorithm "MD5" and developers must periodically check the efficiency of the encryption because over time the encryption algorithms that were previously considered secure become weak.
5	The App uses the encryption mode CBC with PKCS5/PKCS7 padding. This configuration is vulnerable to padding oracle attacks.	CVSS V2: 7.4 (high) CWE: CWE-649	The efficiency of the encryption algorithm is not periodically checked.

## 3.2 Code structure

### 3.2.1 Documentation analysis

There is poor documentation issue in most of the classes because of that we could not relate classes between each other and their outputs.

```
12
13 public class CordovaPlugin {
14     static final /* synthetic */ boolean $assertionsDisabled = (!CordovaPlugin.class.desiredAssertionStatus());
15     public CordovaInterface cordova;
16     protected CordovaPreferences preferences;
17     private String serviceName;
18     public CordovaWebView webView;
19
20     public final void privateInitialize(String serviceName2, CordovaInterface cordova2, CordovaWebView webView2, CordovaPreferences prefere
21     if ($assertionsDisabled || this.cordova == null) {
22         this.serviceName = serviceName2;
23         this.cordova = cordova2;
24         this.webView = webView2;
25         this.preferences = preferences2;
26         initialize(cordova2, webView2);
27         pluginInitialize();
28         return;
29     }
30     throw new AssertionError();
31 }
32
33 public void initialize(CordovaInterface cordova2, CordovaWebView webView2) {
34 }
35
36 /* access modifiers changed from: protected */
37 public void pluginInitialize() {
38 }
39
40 public String getServiceName() {
41     return this.serviceName;
42 }
```

*Example of poor documentation.*

### 3.2.2 Meaningful identifiers

The variables and classes name's unclear and incomprehensible, so we could not understand its purpose.

```
3 public interface cp {
4     public static final o a;
5     public static final o b;
6     public static final o c = a.a("2.2");
7     public static final o d = a.a("2.3");
8     public static final o e;
9     public static final o f;
10    public static final o g;
11    public static final o h;
12    public static final o i = g.a("2");
13    public static final o j = g.a("3");
14    public static final o k = g.a("4");
15    public static final o l = g.a("5");
16    public static final o m = g.a("6");
17    public static final o n = g.a("7");
18    public static final o o = g.a("8");
19    public static final o p = g.a("9");
20    public static final o q = g.a("10");
21    public static final o r = g.a("11");
22    public static final o s = g.a("12");
23    public static final o t = g.a("13");
24    public static final o u = g.a("14");
25 }
```

*Example of unclear identifiers.*

### 3.2.3 Vertical alignment

The purpose of vertical alignment to facilitate understanding of the assignment.

```
47 public class SocialSharing extends CordovaPlugin {
48     private static final String ACTION_AVAILABLE_EVENT = "available";
49     private static final String ACTION_CAN_SHARE_VIA = "canShareVia";
50     private static final String ACTION_CAN_SHARE_VIA_EMAIL = "canShareViaEmail";
51     private static final String ACTION_SHARE_EVENT = "share";
52     private static final String ACTION_SHARE_VIA = "shareVia";
53     private static final String ACTION_SHARE_VIA_EMAIL_EVENT = "shareViaEmail";
54     private static final String ACTION_SHARE_VIA_FACEBOOK_EVENT = "shareViaFacebook";
55     private static final String ACTION_SHARE_VIA_FACEBOOK_WITH_PASTEMESSAGEHINT = "shareViaFacebookWithPasteMessageHint";
56     private static final String ACTION_SHARE_VIA_INSTAGRAM_EVENT = "shareViaInstagram";
57     private static final String ACTION_SHARE_VIA_SMS_EVENT = "shareViaSMS";
58     private static final String ACTION_SHARE_VIA_TWITTER_EVENT = "shareViaTwitter";
59     private static final String ACTION_SHARE_VIA_WHATSAPP_EVENT = "shareViaWhatsApp";
60     private static final String ACTION_SHARE_WITH_OPTIONS_EVENT = "shareWithOptions";
61     private static final int ACTIVITY_CODE_SENDVIAEMAIL = 3;
62     private static final int ACTIVITY_CODE_SENDVIAWHATSAPP = 4;
63     private static final int ACTIVITY_CODE_SEND__BOOLRESULT = 1;
64     private static final int ACTIVITY_CODE_SEND__OBJECT = 2;
65     private static final Map<String, String> MIME_Map = new HashMap();
66     private CallbackContext _callbackContext;
67     private String pasteMessage;
68
69     private abstract class SocialSharingRunnable implements Runnable {
70         public CallbackContext callbackContext;
71
72         SocialSharingRunnable(CallbackContext cb) {
73             this.callbackContext = cb;
```

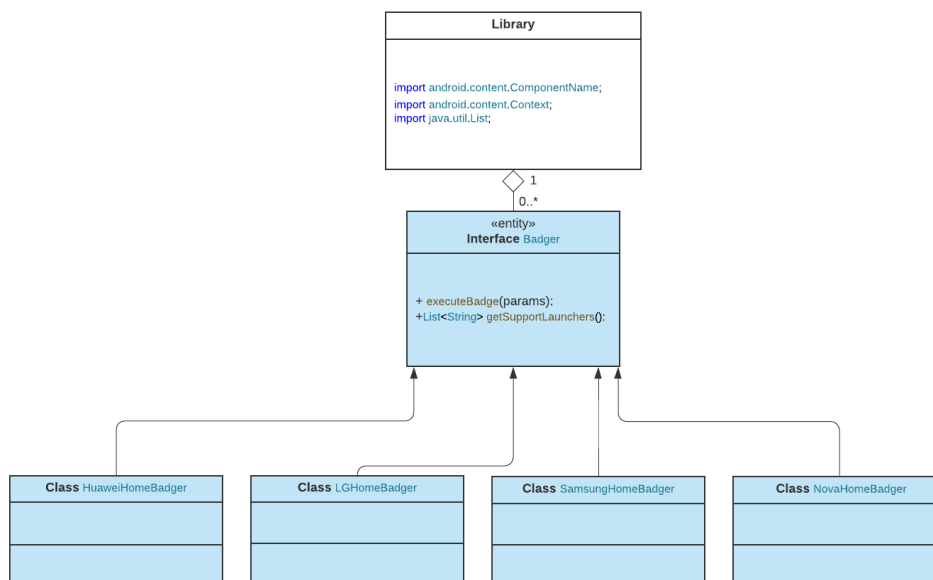
Example of not using vertical alignment.

## 3.3 Design

### 3.3.1 Design principle

#### 3.3.1.1 Low coupling

There are positive aspects in some classes that there is low coupling between them.



Example of low coupling.

### **3.3.2 Security design**

Security design also has to take into account the purpose, criticality, and operational environment of the application. So, through the analysis we did on the code, we discovered its lack of high quality, and after our study of the security design, we found that this helps to increase security.

#### **3.3.2.1 Documentation**

We found in SABB java classes has poor documentation which increase the security.

#### **3.3.2.2 Meaningful identifiers**

We notice that the code has ambiguous identifiers which limit the visibility of information in a program for security reasons.

#### **3.3.2.3 Distribution**

This system is an application on phones, it uses a high distribution, and this affects security.



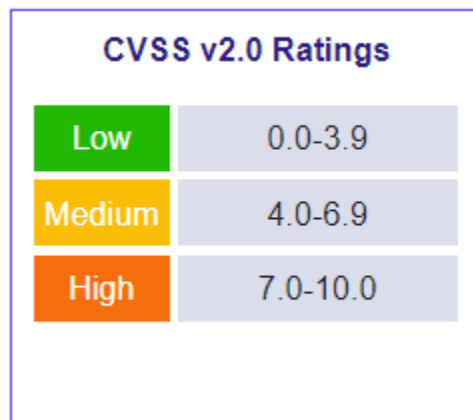
## 4 Discussion about results

### 4.1 Code analysis

We found from code analysis that user information is vulnerable because it stored the user information in a log file (user information contains username, password, keys etc.). Also, they use old encryption algorithm (MD5) which is easy to be hacked.

The CVSS standard rate them in range of (7.4 ~ 7.5) which considered high rate.

CVSS: The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of computer system security vulnerabilities.



A table titled "CVSS v2.0 Ratings" showing three levels of vulnerability severity: Low (0.0-3.9), Medium (4.0-6.9), and High (7.0-10.0). The table is enclosed in a purple border.

CVSS v2.0 Ratings	
Low	0.0-3.9
Medium	4.0-6.9
High	7.0-10.0

CVSS rates

We suggest new encryption algorithm for example, AES (Advanced encryption standard) to protect user sensitive information and do not store the information in a log file.

### 4.2 Code structure

We noticed there is poor documentation, even if there is a documentation it is not in javaDoc style. In addition, there is meaningless variables name. And we think from our study that all due to security reasons.

## 5 Conclusion

We conclude documentation, classes and variables names should be explicit, but if it is an open source such as Android, so we might dispense it because of the users especially the attackers in order to not be used to find its vulnerabilities.

We learned that there is a standard to help us rate the quality of the code like CVSS.

We found low extendibility between classes witch it reduces the coupling.

The expected future work that we imagined is to not include sensitive user information in backup copies because it makes it easier for the attacker to extract it, also expecting to use a strong and sufficiently random number generators for example, Random Number Generator (RNG).

We also expect to check the integrity of encryption of Security-Relevant Inputs and use internal data storage for sensitive information because even if the user decides to delete the application, his information in the application will not be deleted and that will be a good chance for the attacker to take advantage from this vulnerability.

We expect to improve the code structure from (coherent indentation, vertical alignment, high cohesion classes).