Imam Mohammad Ibn Saud Islamic University
College of Computer and Information Sciences
Computer Science Department

Project Title:	**Workshop 1**
Section	**171**

Course Title:	Software Engineering 2
Course Code:	CS 392
Course Instructor:	Sultan S. Alqahtani

Group 1

Student Name	Student ID
Turki Mohammed Alqahtani	440016263
Abdullah Mohammed Alqahtani	440018317
Faisal Mansour Alkhalifah	440025849
Abdullah Mohammed Alrasheed	439027348

11-4-2021

Table of Contents

Introduction 4
Sections of the report 4
Security 4
Code Quality 5
Bugs 5
The tools that we are going to use 5
mobSF: 5
PMD: 6
Spot bugs: 6
Snyk: 6
Our Goal 7
STC Pay 7
API 7
Notifications for Android 7
Notification Manger: 7
Notifications Channels : 8
Android Content Provider 8
Base64 8
Base64 encoding 9
Base64 decoding 9
what is manifest analyses 9
manifest analyses (problems) 9
DataBase 11
Security 11
Total cost and service stability 11
Source code Security Analysis 12
Warnings: 12
SQL Injection(CWE-89): 12
Use of insecure random(CWE-330): 13
Incorrect Default Permissions(CWE-276): 13
Method that exposes the software(CWE-749): 14
Log of sensitive information(CWE-532): 15
Good: 16
Secure Sockets Layer(SSL): 16
Root checker: 16

Informational : 17
Using clipboard:..... 17
Clean code.....18
UnusedFormalParameter.....18
UnusedPrivateField.....18
UnusedPrivateMethod.....19
Self comparison: 19
Unclosed InputStream:.....19
Self-assignment:.....20
Unnecessary method call: 20
Unnecessary if statement: 20
Unreachable code:.....21
Bugs: 21
Bad attempt to compute absolute value of random integer : 21
Code documentation 22
Discussions 23
Conclusion 23
Message.....23

Introduction

In this workshop, we are going to inspect and analyze an android bank application. We are going to take a deep dive into multiple aspects of this bank application such as Security, Code quality,implementation and documentation.The whole idea behind SoftWare Engineering is that you will be working in a shared environment (in a team), as a team member you will have to follow certain rules and standards. The output Product and its implementation and functionality will reflect upon these rules and standards.We are going to look at the bank application app from different perspectives.

Sections of the report

In the introduction we are going to take a look at each section of the report and the purpose of that section and its part of making the application.

- Security
  - APIS
  - DataBase
  - Manifest analysis
  - Source code security analysis

- Code Quality
  - Clean code
  - bugs
  - Code documentation

Security

The source code of a project is the core on which every aspect of the project stands on , which makes it vulnerable to cyber attacks . Therefore the level of security must be at a high level and ensures security measurements and protocols are meeting high standards.

Code Quality

One of the topics that we are going to analyze in this project is code quality. Code quality is a set of standards and rules that a programmer should follow in order to achieve quality in their code, this will significantly improve reusability,and will make the maintenance process easier and overall will make the code easy to read for all parties included in the development.

There are methods of code analysis, including static and dynamic code analysis.

Static analyzes the code without executing it, and its task is to discover errors and problems in the code

In dynamic, you analyze the code using real-time data

Bugs

In order to have a fully functional bank application or any kind of application the number of bugs are needed to be as minimum as possible.In software projects there are always more bugs and the Term that we are going to use “bug free” does not necessarily mean that the bank application has zero bugs, It means that the application was tested thoroughly within the time constraints of this report.

The tools that we are going to use

mobSF:

The main tool that will be used to analyze and view the source code is Mobile Security Framework. Mobile Security Framework is an open source framework that is used to analyze mobile applications, it can run malware analysis and is also capable of performing static and dynamic analysis. The tool will also provide us with the source code on which we can base our report on .

PMD:

it's a source code analyzer, used to find unused code, Bugs, Errors, Design problems, etc we are going to use it to analyze STCPay source code, for example, we run the tool on a small sample from the source code and we got several problems some of them could be positive errors like naming a class. in this problem, we have several things like where the problem begins and where it ends, description and the priority of the problem, and link for more information.

```
{
    "beginline": 276,
    "begincolumn": 11,
    "endline": 277,
    "endcolumn": 9,
    "description": "Avoid empty catch blocks",
    "rule": "EmptyCatchBlock",
    "ruleset" : "Error Prone",
    "priority": 3,
    "externalInfoUrl": "https://pmd.github.io/pmd-6.39.6/pmd_rules_java_errorprone.html#emptycatchblock"
},
```

```
268.public boolean hasValue() {
269.    boolean z;
270.    boolean z2 = this.g.getKeyAndEncryptedValue() != null;
271.    try {
272.        if (this.g.getUnencryptedValue() == null) {
273.            z = false;
274.            return !z2 || z;
275.        }
276.    } catch (CryptException unused) {
277.    }
278.    z = true;
280.    if (!z2) {
281.    }
282.}
```

Spot bugs:

Spot Bugs is an extension used for static code analysis to find bugs, logical errors and useless (functions, conditions, variables).

Snyk:

Find and automatically fix vulnerabilities in your code, open source dependencies, containers, and infrastructure as code all powered by Snyk's industry-leading security intelligence.

Our Goal

Our goal is to output a fully well documented report about the STC Pay application. The report will contain multiple sections as mentioned . The status that will be given to the application will be unbiased.Every step that we took towards analyzing the application will be well thought out and documented. In general our approach is to play the role of the receiving end, and ask ourselves was this report useful? This approach will help us understand and achieve our goal for this report.

STC Pay

STC Pay is a digital wallet to complete daily operations, after the Cabinet approved the transfer of STC Pay to an approved digital bank to become one of the first digital banks in the Kingdom of Saudi Arabia.

The STC Pay digital wallet is designed to make payments, whether personal or local or international money transfers.

One of the main advantages of STC Pay is that it is able to better connect traders with their customers through a secure digital wallet, enabling both sides to complete their transactions quickly, easily and securely.

API

API is a set of definitions and protocols that allow applications to access data with another software.

In our application analysis we found multi APIs and libraries that have been used to improve the application. We will mention some of them and give a brief explanation of their purpose:

Notifications for Android

Notifications are short messages that inform the users of information from your app such as cashback offers. There are types of notifications have been used in STC Pay:

Notification Manger:

To create a new notification and add content into notifications and manage the channels. Also can set a color, the icon for your application by NotificationCompat.

Notifications Channels :

If you have different types of notifications in your app some of the notifications the user is interested in and some of them they didn't want to receive. By the Notifications Channels, the user can turn off the category that controls some of the notifications.

Android Content Provider

In the android system, every application has its own database and files which are stored in the application folder in the system. In android another application cannot access this folder unless the folder is stored in the

content provider, other applications can easily access and make use of this data.

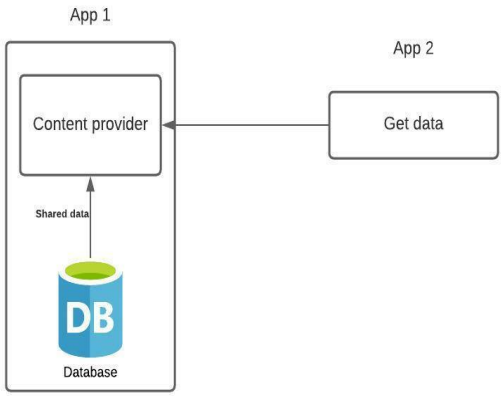
By content provider you can:

- fetch data from a content provider.
- insert, update, or delete data.

Security issue :

If your application shares sensitive data in the content provider may put you at risk.

Applications that are shared in the content provider must grant permission to access data by other applications.



Base64

Base64 is considered as an extra layer of security that helps in encoding string text to Bytes and vice versa and there are two main methods which is Base64 encoding and Base64 decoding

Base64 encoding

When it comes to encoding binary data that needs to be saved and transferred across ASCII-compatible media, Base64 encoding is used. This is done to ensure that the data is not tampered with during transit. Base64 is widely used in a variety of applications, including email and storing complex data in XML.
for example

```
import base64
encoded_data = base64.b64encode("Encode this text")
print("Encoded text with base 64 is")
print(encoded_data)
```

Base64 decoding

it's basically decode encoded a text string into bytes. for example

```
import base64
decoded_data = base64.b64decode("RW5jb2RlIHRoaXMgdGV4dA==")
print("decoded text is ")
print(decoded_data)
```

what is manifest analyses

Manifest analysis is a new form of technique used to dedicate android melicons behaviors by analyzing the manifest file. It mostly focuses on permissions in the applications.

manifest analyses (problems)

- androidx folder under activity folder every class imports"jxbx.x", which is an import call to allow usage from other apps(intent) however these classes don't protect the intent. severity is High

```
import jxbx.C1459;
public final class ActivityViewModelLazyKt {
   @MainThread
   public static final /* synthetic */ <VM extends ViewModel> Lazy<VM> viewModels(ComponentActivity
componentActivity, Function0<? extends ViewModelProvider.Factory> function0) {
       Intrinsics.checkParameterIsNotNull(componentActivity, C1459.m3865(8622));
       if (function0 == null) {
           new ActivityViewModelLazyKt$viewModels$factoryPromise$1(componentActivity);
       }
       Intrinsics.reifiedOperationMarker(4, C1459.m3865(8623));
       throw null;
   }
}
```

solution : by making the intent value =false , however this kind of solution does not make sense because in order for this application to work it needs to take orders from other kinds of apps. so we believe the optimal solution is to make the intent value = True and add an intent filter to filter the orders of this intent.

- Broadcast receiver: is an android API that allows sending and receiving android application events such as (valid payment, notify the host). with that being said. Each class that implements this API does not check for permission(AdjustReferrerReceiver) this means that a malicious application can take control of this component and can cause threats to the application. severity is High.

```
public class AdjustReferrerReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        String stringExtra = intent.getStringExtra(C1459.m3865(7564));
        if (stringExtra != null) {
            Adjust.getDefaultInstance().sendReferrer(stringExtra, context);
        }
    }
}
```

- in folder diagnostics : in the class “DiagnosticsReceiver” that extends Broadcastreceiver class which is responsible for dumping out useful diagnostics information.The problem here is the same as the problem mentioned in number two there is no permission checking when creating the instance therefore can be a vulnerability. severity is High

```
public class DiagnosticsReceiver extends BroadcastReceiver {
    private static final String TAG = Logger.tagWithPrefix(C1459.m3865(19280));

    public void onReceive(@NonNull Context context, @Nullable Intent intent) {
        if (intent != null) {
            Logger.get().debug(TAG, C1459.m3865(19281), new Throwable[0]);
            try {
                WorkManager.getInstance(context).enqueue(OneTimeWorkRequest.from(DiagnosticsWorker.class));
            } catch (IllegalStateException e) {
                Logger.get().error(TAG, C1459.m3865(19282), e);
            }
        }
    }
}
```

DataBase

STC Pay uses a FireBase. A firebase is a cloud hosted database that allows you to store data and sync it with your users in real time.In general the dissection of picking a certain database for your application is very crucial and it breaks down to multiple aspects.

- Operation and maintenance cost
- service stability
- security

Security

FireBase has no default implementation of security measures.It relies on the development team to implement the security aspect of the Database. however it provides default authentication that includes email and password and as a development team you can add more custom authentication to your database.

Total cost and service stability

In general, a firebase database is not expensive, it does not have a fixed price.Looking at it from a stability standpoint Firebase is a very powerful service that can help with developing applications quickly without having to implement new components or modules. Firebase was built for scalability and performance to put it into perspective the application can go from 1 user to 1million users without having to change any code related to the database.

Source code Security Analysis

In this section, we will talk about the source code. It has some good attributes and some bad security problems that may occur if not addressed; some of these may come under code quality.

Here are some of those problems we found in STC pay app:

Warnings:

CWE: or common weakness enumeration is a list of common software weakness types or errors in software code or design that could result in software vulnerability if it was attacked, the CWE list is community developed list that addresses the issues in the source code.

SQL Injection(CWE-89):

SQL databases hold sensitive data and SQL injection allows the attackers to insert a query to make changes to sensitive data that they are not normally allowed to view or belong to some other users.

```
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public final void onCreate(SQLiteDatabase sQLiteDatabase) {
sQLiteDatabase.execSQL(C1459.m3865(18583));
}
```

Found in (lookout, dynatrace)

In the example, the app uses `sQLiteDatabase.execSQL(String)` method that executes SQL statements to the database

Prevention: Since the input field is the main gate for the attacker to send these queries we recommend developers to use:

- Input validation.
- Use firewalls.

Use of insecure random(CWE-330):

Software sometimes needs to generate random numbers to use it in any form but when that random number is predictable it is not useful anymore and the attackers can predict or generate that number and access sensitive information.

```
import java.util.Random;

public static void m4717(String str, String str2, Context context) {
   m4713(context, null, str, str2, C1459.m3865(9839), new Random().nextInt(19901994));
}
```

Found in (lookout, dynatrace)

In this example, the app uses `Random().nextInt()` method that generates a random integer which only takes 2^{48} attempts to break whereas secureRandom take 2^{128} attempts

Prevention: we recommend minimizing the use of random numbers but if it is a must try to use secure randoms.

Incorrect Default Permissions(CWE-276):

The software reads and writes to external storage which is accessible by any other software, the problem is that the software may get modified, that problem may occur during the installation and corrupt the data or even worse misuse of the software.

```
import android.os.Environment;

public void run() {
       ArrayList arrayList = new ArrayList();
       for (File file : this.f8318) {
File file2 = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES), C1925.m5153(this.f8317));
            if (!file2.exists()) {
                file2.mkdirs();
            }
            File file3 = new File(file2, file.getName());
            try {
                file3.createNewFile();
                C1925.m5147(file, file3);
                arrayList.add(file3);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        C1925.m5151(this.f8317, arrayList);
}
```

Found in (lookout, huawei)

In this example, the app uses `Environment.getExternalStoragePublicDirectory()` method which gets the directory from the external storage this makes it vulnerable to misuse from external storage

Prevention: apply after installation checks to check the safety of the software.

The software creates temporary files, temp files may contain sensitive information these information should not be in temporary files.

```
import java.io.File;

public static Uri writeTempStateStoreBitmap(Context context, Bitmap bitmap, Uri uri) {
       boolean z = true;
       if (uri == null) {
           try {
                uri = Uri.fromFile(File.createTempFile(C1459.m3865(12294), C1459.m38
65(12295), context.getCacheDir()));
           } catch (Exception e) {
               Log.w(C1459.m3865(12296), C1459.m3865(12297), e);
               return null;
           }
       } else if (new File(uri.getPath()).exists()) {
           z = false;
       }
       if (z) {
           writeBitmapToUri(context, bitmap, uri, Bitmap.CompressFormat.JPEG, 95);
       }
       return uri;
}
```

Found in (theartofdev)

In this example the app uses `File.createTempFile()` method, that method creates a temp file and store it in a given directory

Prevention: if the files are important or sensitive they must be saved in an organized file or developers must delete these temp files.

Method that exposes the software(CWE-749):

The software uses insecure webview implementation which has exposed method that the attacker can use to exploit to penetrate and get sensitive data or use the software improperly Some methods in webview are never intended to be accessible by any user or to be exposed at all, depending to the behavior of the method that could lead to several critical problems.

```
public View onCreateView(@NonNull LayoutInflater layoutInflater, @Nullable ViewGroup viewGroup, @Nullable {
...
       if (this.f5093) {
           m3965();
           this.f5090.setWebViewClient(new C2262(this, null));
           this.f5090.getSettings().setJavaScriptEnabled(true);
           this.f5090.getSettings().setDomStorageEnabled(true);
           this.f5090.getSettings().setBuiltInZoomControls(false);
           this.f5090.setLongClickable(false);
           this.f5090.setWebChromeClient(new C1790(this));
           this.f5090.addJavascriptInterface(new C3408(this), C1459.m3865(16250));
       }
...
```

In this example the app uses `File.getSettings().setJavaScriptEnabled()` using `setJavaScriptEnabled()` may introduce XSS (Cross-Site Scripting) vulnerabilities that allow the hacker to compromise the communication between the app and the site by executing malicious code to the user browser

Prevention: software developers must examine these methods before using them in their software.

Log of sensitive information(CWE-532):

While logging information is helpful for the maintenance and the development of the software, some of the information is highly sensitive and should never be put in log folders.

```
import android.util.Log;

public void observe(@NonNull LifecycleOwner lifecycleOwner, @NonNull Observer<? super T> observer) {
    if (hasActiveObservers()) {
            Log.w(C1459.m3865(39615), C1459.m3865(39616));
        }
    super.observe(lifecycleOwner, new C1820(this, observer));
}
```

The app is logging info and that's is a vulnerability if not handled correctly

Prevention: consider setting logging levels appropriately to prevent exposing sensitive users or server data.

Good:

Secure Sockets Layer(SSL):

The software uses Secure Sockets Layer(SSL) security protocol to maintain a secure connection and encrypt the communication between the user and the server, which prevents several attacks like man-in-the-middle attack.

```
import java.security.KeyStore;
import javax.net.ssl.SSLSession;

private X509TrustManager m9176(Context context) throws GeneralSecurityException {
Collection<? extends Certificate> generateCertificates = CertificateFactory.getInstance(C1459.m3865(36330)).ge
nerateCertificates(context.getResources().openRawResource(C1841.f7814));
       if (!generateCertificates.isEmpty()) {
           char[] charArray = C1459.m3865(36331).toCharArray();
           KeyStore r1 = m9175(charArray);
           int i = 0;
           for (Certificate certificate : generateCertificates) {
               int i2 = i + 1;
               r1.setCertificateEntry(Integer.toString(i), certificate);
               i = i2;
           }
           KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm()).init(r1, charArray);
           TrustManagerFactory instance = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgori
thm());
           instance.init(r1);
           TrustManager[] trustManagers = instance.getTrustManagers();
           if (trustManagers.length == 1 && (trustManagers[0] instanceof X509TrustManager)) {
               return (X509TrustManager) trustManagers[0];
           }
           throw new IllegalStateException(C1459.m3865(36332) + Arrays.toString(trustManagers));
       }
    throw new IllegalArgumentException(C1459.m3865(36333));
    }
```

The app uses `KeyStore.setCertificateEntry(alias, certificate)` method to assign the certificate to the alias which assign SSL certificate to secure the connection between the browser and the server

Root checker:

The software uses a method that checks if the android device is rooted or not, which makes the android device capable of doing a lot more than what it was intended to do.

```
import com.dynatrace.android.agent.RootDetector;

private void init() {
       this.manufacturer = Utility.trimString(Build.MANUFACTURER, 250);
       this.deviceRooted = RootDetector.isDeviceRooted();
       this.operatingSystem = C1459.m3865(21858) + Build.VERSION.RELEASE;
       this.cpuInformation = Utility.getCPUInfo();
       this.modelId = Build.MODEL;
       this.userLang = formatUserLanguage(Locale.getDefault());
       updateVerboseMetrics();
       updateBasicMetrics();
       updateCommonMetrics();
```

In the example the app uses `RootDetector.isDeviceRooted()` method, that method returns the boolean value if the android device was rooted or not

Informational :

Using clipboard:

Users of the software can copy text to clipboard which is shared by other applications. Some attackers may misuse this feature to their own intentions, the software has security checks that contain this problem but developers must be careful about this problem.

```
import android.content.ClipboardManager;

public final void m9520(Context context, String str, Function0<Void> function0) {
       Intrinsics.checkNotNullParameter(context, C1459.m3865(39511));
       Object systemService = context.getSystemService(C1459.m3865(39512));
       if (systemService != null) {
           ((ClipboardManager) systemService).setPrimaryClip(ClipData.newPlainText(str, str));
           if (function0 != null) {
               function0.invoke();
               return;
           }
           return;
       }
       throw new NullPointerException(C1459.m3865(39513));
   }
```

Here the app is using methods to set and get from the clipboard of the android device.

Clean code

Code is clean if it can be understood easily by everyone on the team. Clean code can be read and enhanced by a developer other than its original author.
so when writing in a weird way that only the developer will understand, that's not clean code.

Here are some of those example we found in the code:

(examples below were fetched out by using PMD tool)

UnusedFormalParameter

arguments in the method or the constructor parameter that aren't used to intialze somthing or computing it, dVar, aVar, cVar, bVar2 aVar2, hVar aren't used in the constructor and might confused other developer

```
private AndroidCrypt(d dVar, b bVar, a aVar, c cVar, b bVar2, a aVar2, DatastoreEncryptionAdapter
datastoreEncryptionAdapter, h hVar, KeyInfo keyInfo) {
    this.b = bVar;
    this.g = datastoreEncryptionAdapter;
    this.i = keyInfo;
}
```

UnusedPrivateField

private attribute that is initialized and assigned when it isn't used, doesn't clarify that this is a clean code

```
public class Manifest {
    private static final Logger i = LoggerFactory.getLogger(Manifest.class);
```

UnusedPrivateMethod

a private method with a defined body that isn't used by any other classes in the whole project

```
private static BasicScannableFile a(String str, MediaType mediaType) {
    if (mediaType.equals(MediaTypeValues.APK)) {
        try {
            return new AndroidApkFile(str);
        } catch (ApkException e) {
            throw new ScannerException(C1459.m3865(26034).concat(String.valueOf(str)), e);
        }
    } else {
        return mediaType.equals(MediaTypeValues.ZIP) ? new ContainerFile(new File(str), mediaType) :
(mediaType.equals(MediaTypeValues.THREEGPP) || mediaType.equals(MediaTypeValues.THREEGPP2) ||
mediaType.equals(MediaTypeValues.MP4)) ? new IsoMediaFile(new File(str), mediaType) :
mediaType.equals(MediaTypeValues.MP3) ? new Id3TagFile(new File(str), mediaType) : new BasicScannableFile(new
File(str), mediaType);
    }
}
```

(Example below were fetched out by using SpotBugs tool)

Self comparison:

This method will calculate the ratio of screen

In the if statement i7 and i6 compares with itself may cause logic error

```
private void computeInsetRatio(int[] iArr, int i, int i2, int i3, int i4, float f, int i5) {
    if (i8 <= i6 && i7 <= i7) {
        iArr[0] = i8;
        iArr[1] = i7;
    } else if (i6 <= i6 && i9 <= i7) {
        iArr[0] = i6;
        iArr[1] = i9;
    }
}
```

(Examples below were fetched out by using Snyk tool)

Unclosed InputStream:

`FileInputStream()` must be closed using `.closed()`, since you opened it, it is your responsibility to close it

```
public static void m5152(File file, File file2) throws IOException {
    m5148(new FileInputStream(file), file2);
}
```

Found in C1925.java

Self-assignment:

Assigning the same value to itself have no effect on the code, and should be removed as it may cause the code to be unclear

```
columnIndexOrThrow16 = columnIndexOrThrow16;
columnIndexOrThrow18 = columnIndexOrThrow18;
columnIndexOrThrow9 = columnIndexOrThrow9;
columnIndexOrThrow11 = columnIndexOrThrow11;
columnIndexOrThrow = columnIndexOrThrow;
columnIndexOrThrow22 = columnIndexOrThrow22;
columnIndexOrThrow3 = columnIndexOrThrow3;
```

Found in WorkSpaceDao_Impl.java

Unnecessary method call:

`.getClass()` have no effect unless it is assigned to a variable

```
private final void zzo(String str) {
    str.getClass();
    this.zzd |= 4;
    this.zzap = str;
}
```

Found in zzcd.java

Unnecessary if statement:

`i5` is assigned then used in an if statement, the statement will always be false and the condition must be removed

```
if (zzh2.zza(DataLayer.EVENT_KEY, 40, str2)) {
    i5 = 0;
    if (i5 != 0) {
        zzq().zzg().zza("Invalid public event name. Event will not be logged (FE)", zzn().zza(str2));
        this.zzy.zzh();
        String zza2 = zzkx.zza(str2, 40, true);
        if (str2 != null) {
            i4 = str2.length();
        }
        this.zzy.zzh().zza(i5, "_ev", zza2, i4);
        return;
    }
}
```

Found in zzhe.java

Unreachable code:

The code under return can not be reached, and should be removed

```
…
        }
    }
    zzq().zzh().zza("Failed to read events from database in reasonable time");
    return null;
    cursor = null;
    zzq().zze().zza("Error reading entries from local database", e);
    this.zzb = true;
    if (cursor != null) {
    }
    if (sQLiteDatabase2 == null) {
    }
    i3++;
}
```

Found in zzet.java

Bugs:

Bad attempt to compute absolute value of random integer :

This code generates a random integer and then computes the absolute value of that random integer. If the number returned by the random number is too large the method should throw an Arithmetic Exception for this value.

```
private String getNonce() {
    return String.valueOf(System.nanoTime()) +String.valueOf(Math.abs(RAND.nextLong()));
}
```

Code documentation

Before we talk about code documentation we would like to talk about code ambiguously(obfuscation).After looking at the source code of STC Pay we noticed that most of the classes are filtered out including the methods and the variable names, some classes don't even have a single line of code in them.Of Course we understand that this is a method of protecting the source code of the application and its called code obfuscation.But this leaves little to no room to inspect the code documentation.

Examples of code obfuscation

```
package io.card.payment;

public class CameraUnavailableException extends RuntimeException {
}
```

```
package kotlin.text;

public class StringsKt__IndentKt extends StringsKt__AppendableKt {
}
```

```
public void onLayout(boolean z, int i, int i2, int i3, int i4) {
    if (z && getChildCount() > 0) {
        int i5 = i3 - i;
        int i6 = i4 - i2;
        int i7 = this.mPreviewHeight;
        int i8 = i5 * i7;
        int i9 = this.mPreviewWidth;
        if (i8 > i6 * i9) {
            int i10 = (i9 * i6) / i7;
            this.mSurfaceView.layout((i5 - i10) / 2, 0, (i5 + i10) / 2, i6);
            return;
        }
        int i11 = (i7 * i5) / i9;
        this.mSurfaceView.layout(0, (i6 - i11) / 2, i5, (i6 + i11) / 2);
    }
}
```

our initial plan was to say that the quality of the code documentation is very low and it does not meet the Code Documentation standards however, this assumption is wrong.From a security perspective it does not make any sense to make the code obfuscated but leave the documentation that describes the function/class and what it receives and what output does it produce on display.We have come to the conclusion that the development team have also obfuscated the code documentation as well, thus we can't give a status to the code documentation.

Discussions

We generally have around 4 meetings a week where we talk about the report requirements and our understanding of it and to show our progress and have multiple perspectives and opinions. These meetings were extremely helpful as it gives an outside perspective of the work and it allows for “Team Work” which is the whole point of software engineering, Although the meetings that were conducted were only 4 people meetings we believe it was a perfect practice to a real world environment.

Conclusion

In light of these informations, we have covered every aspect of the STC Pay bank application. We have talked about the Security aspect that includes Database, Source Code, API and manifest analysis with each of these subsections we have provided our understanding and problems that we generated and their solutions (if available). We have also taken a deep dive into the Quality code and have divided it into three sections Clean code , Code documentation and bugs.

Message

We would like to end our report with a message.”Tell me and I forget. Show me and I remember . Involve me and I understand”. This workshop report was quite the journey, yes the topics that we have covered here were mentioned in the course slides but to have to apply it in a real world example truly makes our understanding of these topics excel.