ASSIGNMENT-2 FIT-5003

Leveraging static analysis tools to pinpoint security issues of software applications

Submitted by : **SUYOGYA TANEJA(28961897)**

CONTENTS

INTRODUCTION

The code review is one of the oldest and useful methods to detect defects. The ratio benefits make the static analysis quite a useful method, applied by many companies.

STATIC CODE ANALYSIS

Static code analysis is used to test the quality of the code by scanning the source code.

Static code analysis is required due to the following reasons:

1. It enhances the code review quality
2. It does cost saving.

The report analyzes GitHub source code and an apk file to do code analysis using static analysis tools. The tools used for the analysis are Androbugs and Findbugs. The tools have been explained in depth and explored in various aspects. Later, various projects were analyzed and complex source code and Android application were chosen. Further, the tools were used to find the existing security issues in both the projects. The detailed explanation of the experimental setup has been mentioned in the report. The static analysers have also been compared in depth for in-depth understanding. The results generated by the tools were analyzed and the appropriate solutions and recommendations for the same have been given in the conclusion.

OVERVIEW OF THE REPORT

The following diagram gives an overview of the report.

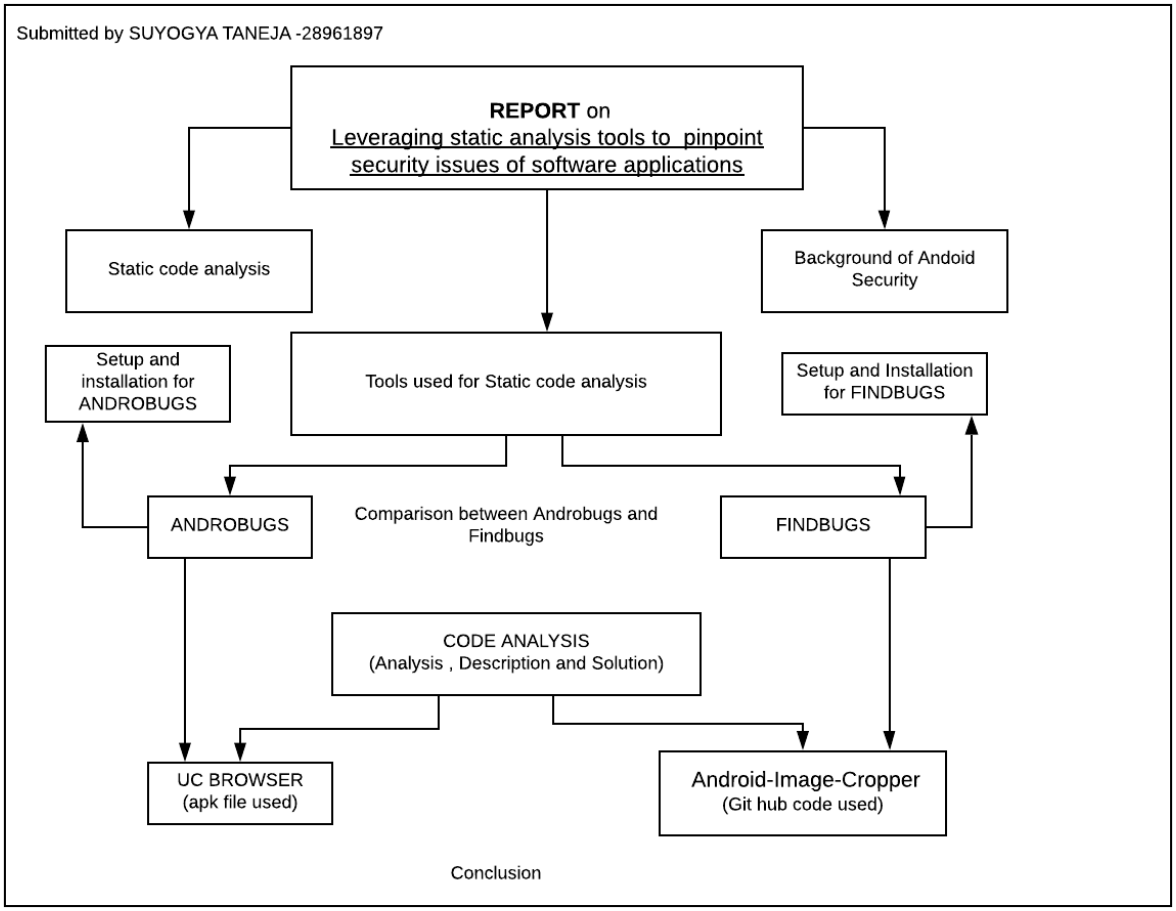

Figure 1: Overview of the report

STATIC ANALYSIS TOOL

The static analysis tool is selected on the basis of the language in which the software or code is written. The code may be written in many languages such as C++, JAVA, JAVASCRIPT, Python and many more. The tools used for code static analysis in this report are:

1. Androbugs
2. Findbugs

BASIC BACKGROUND OF ANDROID SECURITY

Every Android application operates in a sandbox by default. Hence, any application in android cannot access the services outside the sandbox without permission. Therefore, no android application can affect the other application, user, or operating system without getting permission granted for that. If two android applications want to share any data with each other than they are required to specify the same **"Android:sharedUserId"** in the **AndroidManifest.xml** and must sign with the same certificate.

Each Android application runs with its UID in its own Dalvik virtual machine. It provides CPU protection, memory protection. Android application announces the permission requirement.

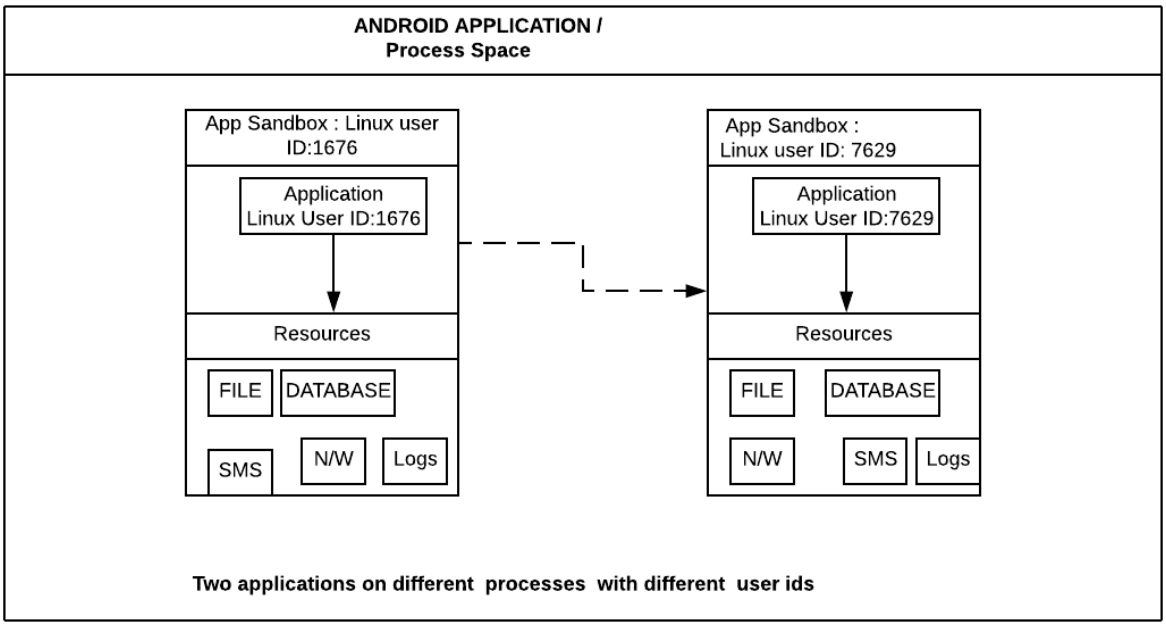

Figure 2: The above figure shows how two android application processes interact with each other.

The Android platform takes advantage of the Linux user-based protection to identify and isolate app resources. This isolates apps from each other and protects apps and the system from malicious apps. To do this, Android assigns a unique user ID (UID) to each Android application and runs it in its own process.

Difference between Android and iOS Sandbox:

ANDROID	iOS
Android application announces permission requirement.	iOS applications have same permissions.
Users can approve permissions at installation time.	Users approval is required for the first time of application installation.
The application may have more powerful permissions.	The application may have limited permissions.

Hence, it is easier to exploit an Android application as compared to the iOS application because of its permissions i.e android applications have more powerful permissions as compared to the limited permissions given to an iOS application.

HOW TO CHOOSE A STATIC ANALYSIS TOOL

The most common approach used for choosing a static analysis tool for code analysis is to choose that tool which reports the maximum number of violations. Static program analysis is the analysis of a particular software which is performed without actually executing the programs.

WHY STATIC CODE ANALYSIS:

1. It is usually important as it doesn't require the code to be executed in order to find the bugs in the source code.
2. It also helps in detecting the potential bugs in the initial stage of the development cycle, therefore the cost to fix the code reduces.

ANDROBUGS

Androbugs Framework is an Android vulnerability analysis system that is used to find potential security vulnerabilities in ANDROID applications. It is written in Python language. It doesn't have a great graphical user interface but is really efficient and accurate to help the developers in finding the possible security vulnerabilities.

The features are as follows:

1. It helps in finding the security vulnerabilities in an Android application
2. It checks whether the code is written missing the best practices.
3. It checks the dangerous shell commands.
4. It collects information from millions of android applications.
5. It also checks application security protection.

Androbugs classifies the bugs on the basis of the severity level as following:

Severity Level	Description
Critical	A confirmed security vulnerability that should be resolved. (except for the testing code)
Warning	Androbugs framework is not sure whether this is a security vulnerability or not. This needs to confirmed by the developers manually.
Notice	Low priority issue or Androbugs framework tries to let you know some additional information.
Info	No security issue detected

STEPS TO INSTALL AND SETUP ANDROBUGS

A flowchart is used to explain the installation steps for ANDROBUGS in windows.

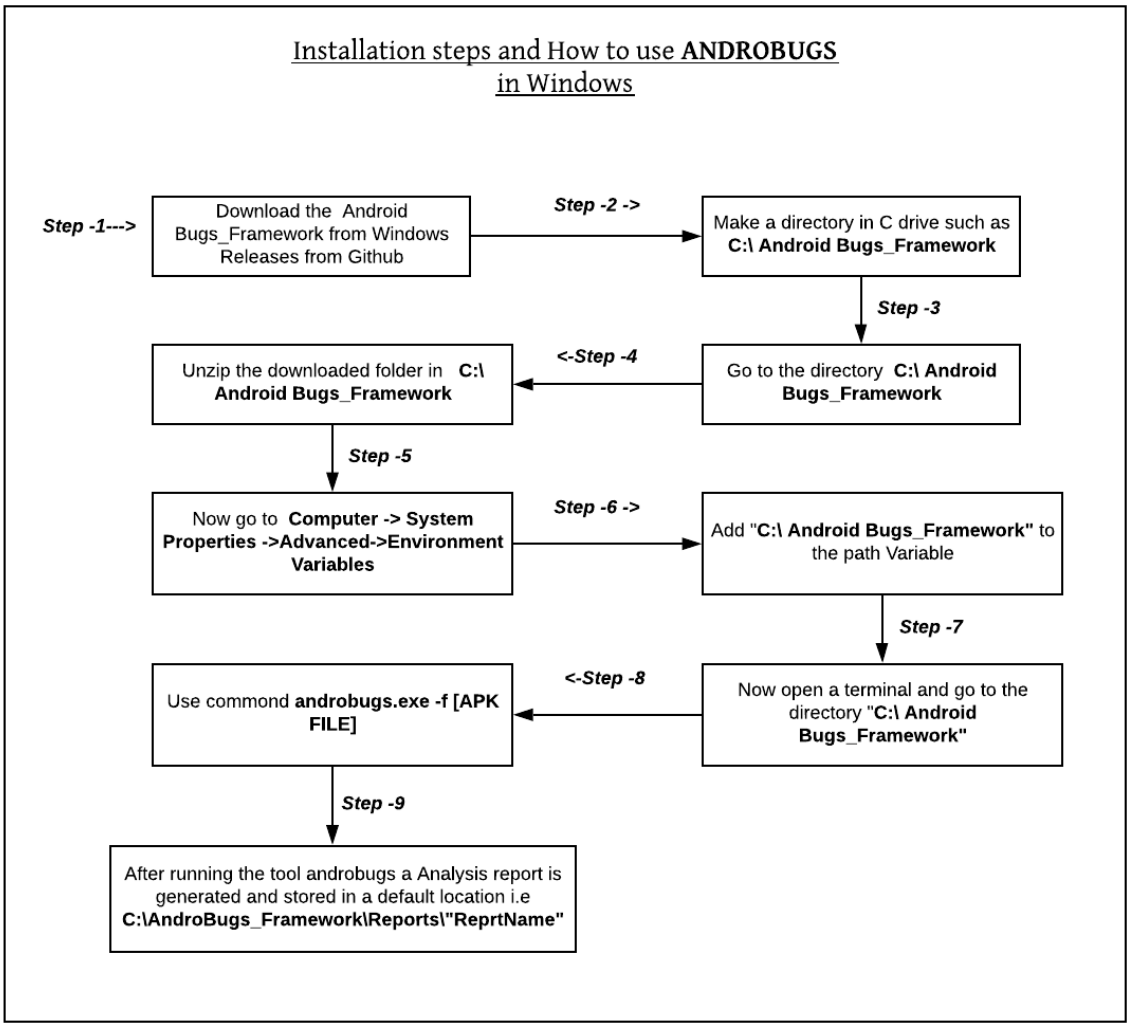

Figure 3: The figure shows the installation and the steps for using Androbugs Framework in Windows

***Step-1*:** Download the AndroBugs_Framework from GitHub.

The Androbugs framework can be downloaded from the GitHub repository i.e (https://github.com/AndroBugs/AndroBugs_Framework)

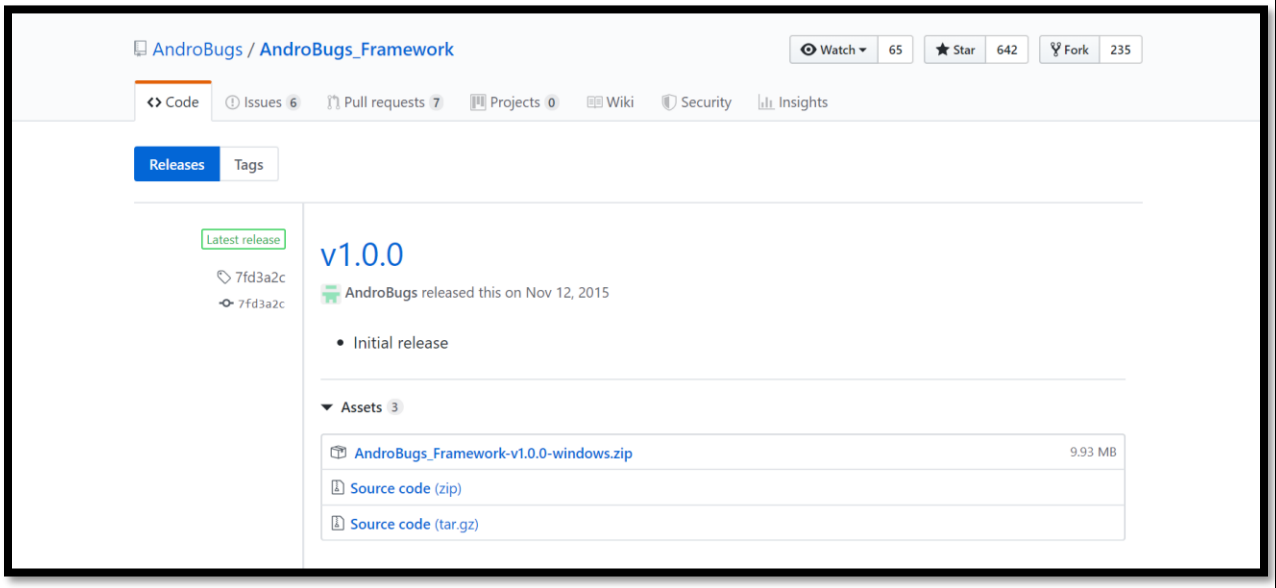

Figure 4: Screenshot of the github repository of Androbugs framework

***Step 2*:** After downloading, Create a new directory named AndroBugs_Frameowork. Using the following command:

mkdir AndroBugs_Framework

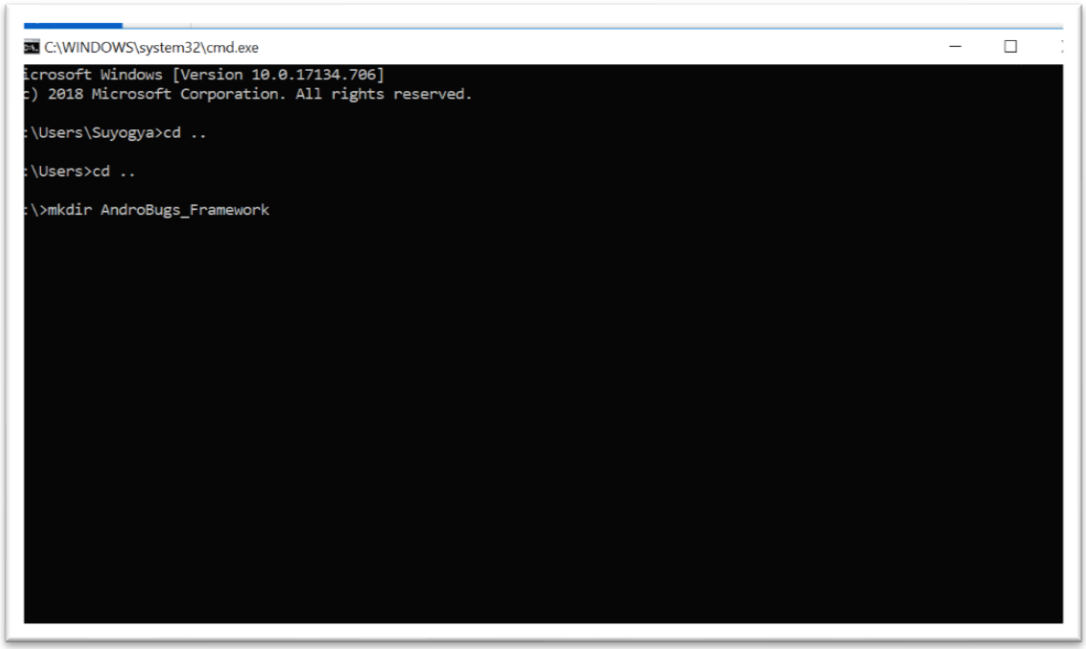

Figure 5: Androbugs framework directory is made to extract the downloaded files in it

***Step 3*:** After making the new directory, go to the directory and Unzip the AndroBugs_Framework in the same directory.

***Step 4*:** After unzipping the folder go to system and add the new path in Environment variables i.e C:\ AndroBugs_Framework. The following image shows how to set the path for Environment Variables.

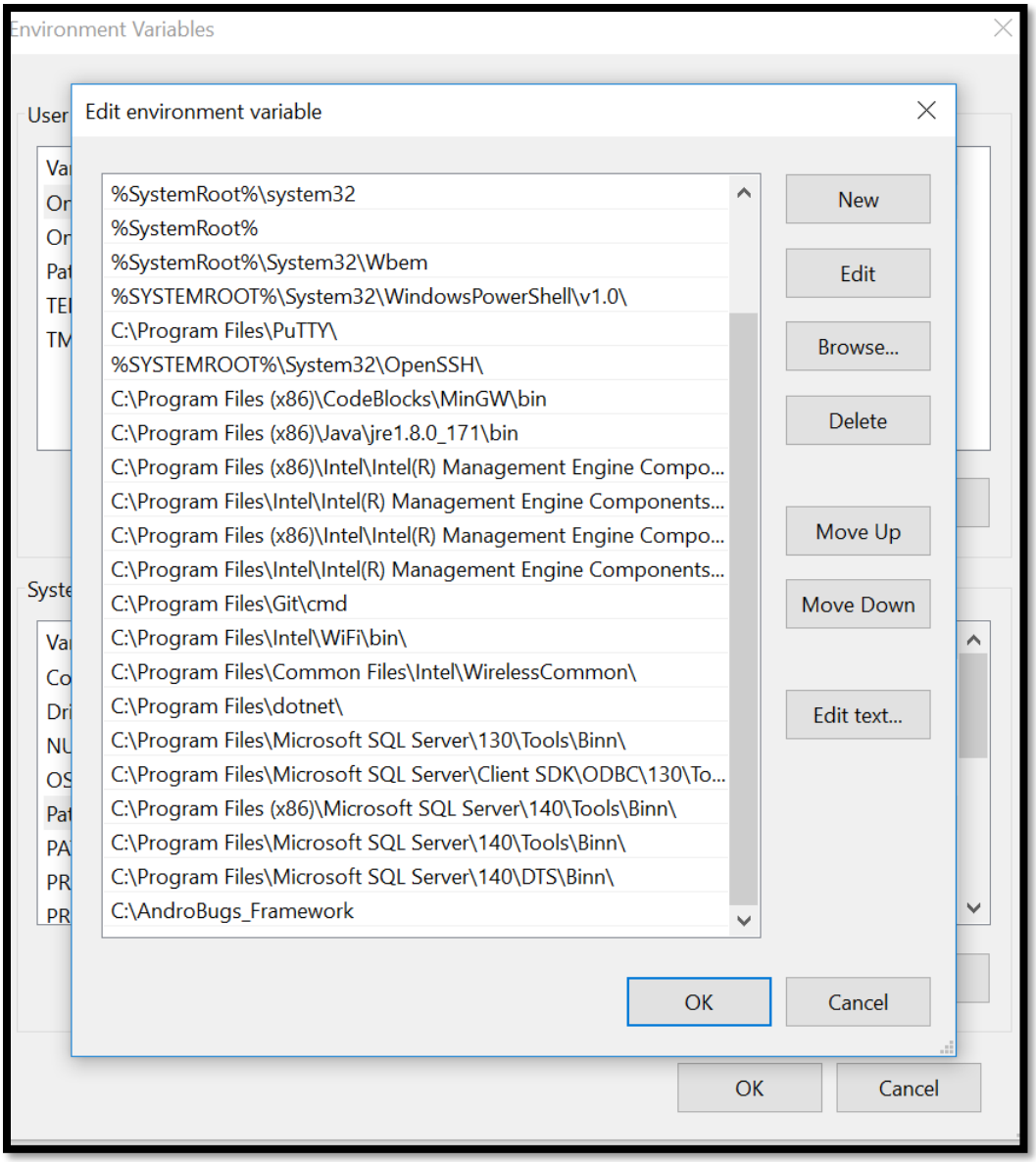

Figure 6: Environment Variables for Androbugs framework is added

***Step 5*:** After this, you can go to the terminal and enter the following command to run the androbugs framework:

Androbugs.exe -f [APK file]

***Step 6*:** Now scan the apk file of the Android application to find the vulnerabilities.

The above steps are only for Windows environment as the whole experiment was performed for windows only.

FINDING TARGET CODE FOR ANALYSIS-ANDROBUGS

The target code to analyze using the analysis tool was taken from the apkpure.com. The link for downloading the UC browser is https://apkpure.com/ucbrowser/com.UCMobile.intl/download?from=details.

Its size is approximately 45 MB. The main features of the UC browser are as follows:

1. It is fast and stable.
2. It compresses data and speeds up the navigation and saves the space.
3. It also blocks ads on the main sites which makes it more user-friendly.

UC BROWSER

UC browser is a mobile browser which is widely used in Android, iOS, windows phone, and many other platforms. The apk file of UC browser for Android was downloaded from apkpure.com. Later, the apk file for the UC browser was analyzed using Androbugs framework and the following vulnerabilities were found and there were ranked on the basis of the priority.

Figure 7: UC Browser attributes mentioned in the Androbugs framework.

Androbugs tool found the following vulnerabilities in the Android application.

PRIORITY	VULNERABILITY
CRITICAL	Runtime command checking
CRITICAL	Fragment Vulnerability checking
CRITICAL	Implicit Service checking
CRITICAL	SSL implementation checking
CRITICAL	SSL connection checking
CRITICAL	SSL certificate verification checking
CRITICAL	AndroidManifest Critical Use Permission Checking
CRITICAL	WebView RCE Vulnerability Checking

1. **RUNTIME COMMAND CHECKING**

   In Java java.lang.Runtime.exec(String command) method is used to execute a specified string command in a separate process. So in the android application, Runtime.exec() can be called to execute the operating system commands.

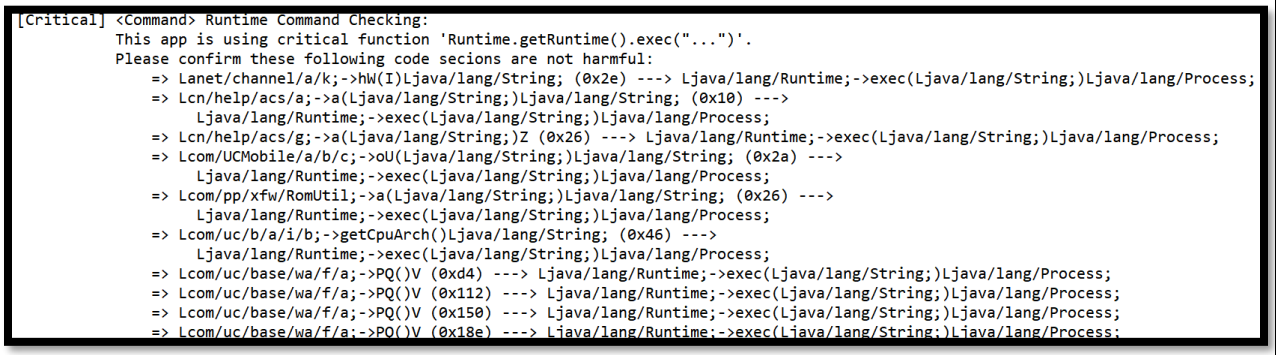

Figure 8: Run time command checking vulnerability in ANDROBUGS FRAMEWORK

Vulnerability:

It can be used to inject the malicious code within the JVM.

The biggest concern with the Runtime command checking is Command Injection. The attacker can execute arbitrary commands on the operating system using this application. This attack is possible in this android applications due to insufficient input validation. The attacker may extend the default functionality of the application which executes the system commands, without even injecting the code.

Solution:
The best practice is that the user should not Runtime.exec() in the program. In case, the programmer uses the method, it should be specified which characters and commands to be used.

2. **Fragment Vulnerability checking**

Vulnerability:

This vulnerability may affect many other applications such as settings, Google, Gmail, and many other important applications. The application which extended Preference activity class using an exported activity was automatically vulnerable.
PreferenceActivity is a base class which is provided by the Android framework in order to show the hierarchy of preferences to the users. Furthermore, these preferences are associated with a PreferenceFragment. An intent tells the preference activity which fragment should be displayed first.
Solution: In the latest update of Android KitKat, a patch has been provided to resolve this vulnerability. The new patch requires the applications to override the new method, Preference activity.isvalidfragment which has been added to the Android framework.

Figure 9: Fragment vulnerability checking vulnerability in Androbugs Framework

3. **Implicit Service checking**

The code implements implicit intent to start a service and it is not recommended as the behavior of other apps can be influenced by this intent.

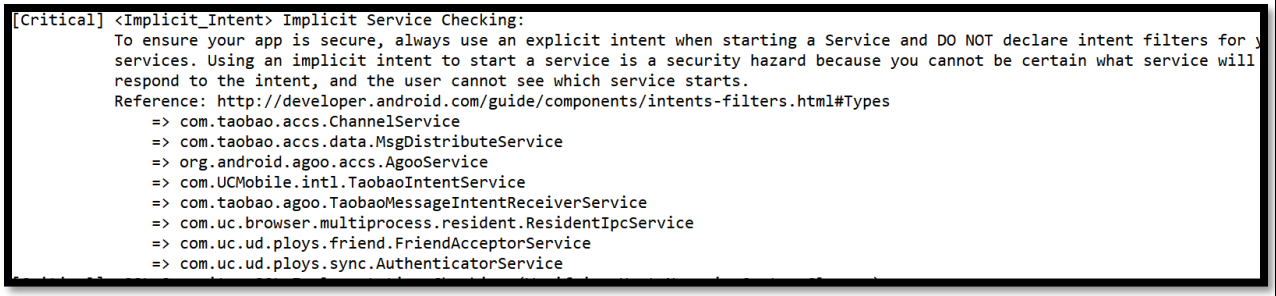

Figure 10: Implicit Service checking vulnerability in Androbugs framework.

Solution: Instead of this the code must use an explicit intent to start the service. Therefore, in order to ensure that there is less vulnerability in the Android application always use explicit intent when starting a service and do not declare the intent filters for the services too.

4. **SSL Implementation checking**

This is one of the critical vulnerabilities as it allows the attackers to do Man in the Middle attacks using valid certificated without the knowledge of the users.

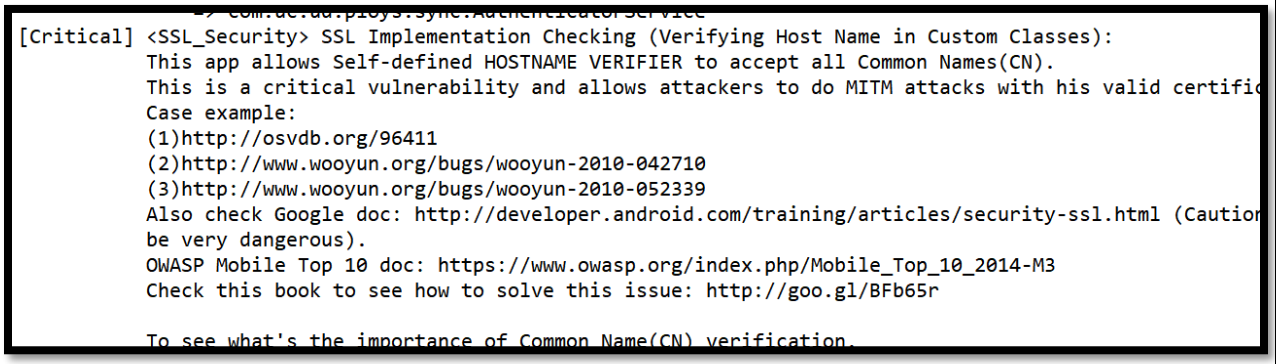

Figure 11: SSL Implementation checking vulnerability in Androbugs framework

Vulnerability :

This application allows self-defined Hostname verifier to accept all the common names(CN).

Solution:

Hostname verification should be done. A custom Certificate Authority may be added to the user certificate store.

5. SSL connection checking

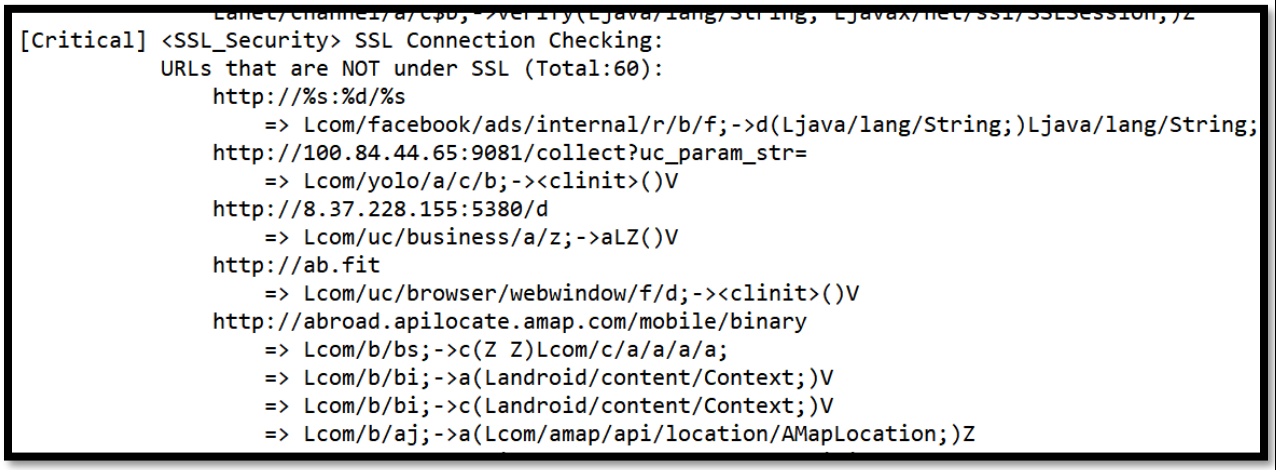

Figure 12: SSL connection checking vulnerability in Androbugs framework

Solution: The application should only allow secure HTTPS connections. The URLs which are not under SSL should be changed from HTTP to HTTPS.

6. **SSL certificate verification checking**

Vulnerability: This android application doesn't check the validation of SSL certificates making it vulnerable to Man In the middle attacks. It allows self-signed, expired or mismatched CN certificates for SSL connections.

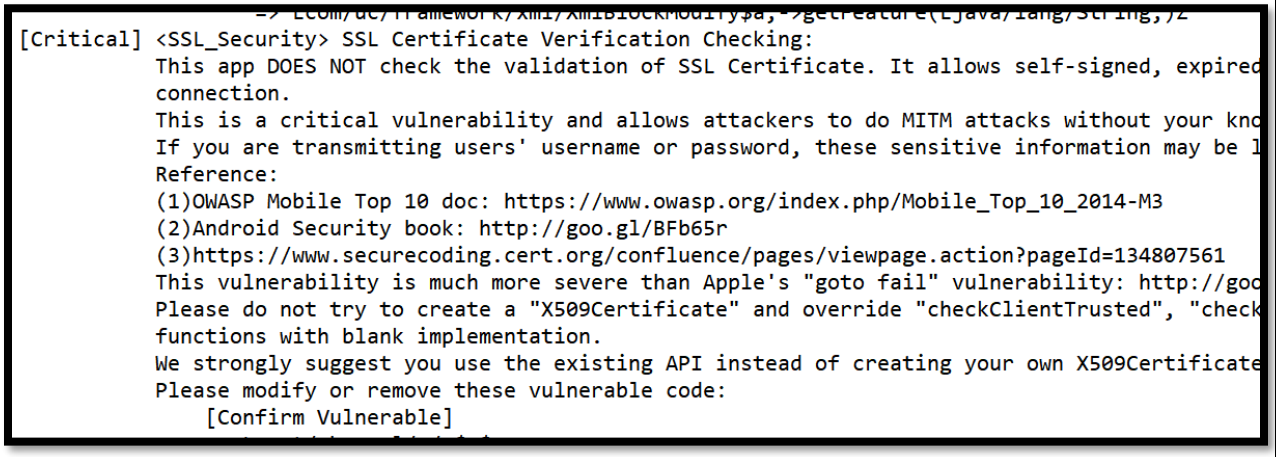

Figure 13: SSL certificate verification vulnerability in Androbugs framework

Solution: The existing APIs should be used instead of creating own X.509 certificate class. The vulnerable code should be modified if required.

7. **Android Manifest critical use permission checking**

Vulnerability: The app requests for high privilege permissions through the manifest file which may cause unintended side effects.

```
                    -> used by: Lcom/uc/browser/b/c/a$2;->run()V
[Critical]  AndroidManifest Critical Use Permission Checking:
           This app has very high privileges. Use it carefully.
               Critical use-permission found: "android.permission.MOUNT_UNMOUNT_FILESYSTEMS"
[Critical] <WebView><Remote Code Execution><#CVE-2013-4710#> WebView RCE Vulnerability Checking
           Found a critical WebView "addJavascriptInterface" vulnerability. This method can be
           application.
           This is a powerful feature, but also presents a security risk for applications targe
           because JavaScript could use reflection to access an injected object's public fields
           untrusted content could allow an attacker to manipulate the host application in unin
           permissions of the host application.
```

Figure 14: Android manifest critical use permission checking vulnerability in Androbugs framework

Solution: Instead of this, the permission needs to be reviewed and the only the required set of permissions must be requested.

8. **Webview RCE vulnerability checking**

Vulnerability: The application uses a web view with "addJavascriptInterface" method. The method can also allow JavaScript to control the application and as a result, malicious javascript can be injected to run with the app permissions.

```
Critical] <WebView><Remote Code Execution><#CVE-2013-4710#> WebView RCE Vulnerability Checking:
          Found a critical WebView "addJavascriptInterface" vulnerability. This method can be used to al
          application.
          This is a powerful feature, but also presents a security risk for applications targeted to API
          because JavaScript could use reflection to access an injected object's public fields. Use of t
          untrusted content could allow an attacker to manipulate the host application in unintended way
          permissions of the host application.
          Reference:
            1."http://developer.android.com/reference/android/webkit/WebView.html#addJavascriptInterface
          java.lang.String) "
            2.https://labs.mwrinfosecurity.com/blog/2013/09/24/webview-addjavascriptinterface-remote-cod
            3.http://50.56.33.56/blog/?p=314
            4.http://blog.trustlook.com/2013/09/04/alert-android-webview-addjavascriptinterface-code-exe
          Please modify the below code:
              => Lcom/uc/browser/webcore/d/b;->addJavascriptInterface(Ljava/lang/Object; Ljava/lang/Stri
                   Lcom/uc/webview/browser/BrowserWebView;->addJavascriptInterface(Ljava/lang/Object; L
```

Figure 15: Webview RCE vulnerability checking vulnerability in Androbugs framework

Solution: There must proper validation of the code that runs in the Web view.

Other than the critical vulnerabilities in the UC browser, there were few warnings such as **Dynamic Code loading, external storage accessing, AndroidManifest Exported components checking** which cannot be ignored. They may not cause any major security issue but should be resolved in order to improve the efficiency and performance of the application**.** The warnings may be resolved depending on its impact on the android application.

WARNING	Description/Analysis	ACTION
Dynamic code loading(DCL)	Android has provided Dynamic code loading since API level -1.DCL allows a developer to load additional code at runtime. By using DCL, an attacker can easily change the behavior of an application at run time and can easily deploy malicious code.	**No action required-** May require the security verification of DCL
External storage Accessing	The important files should not be written to the external storage.	**No action required**
AndroidManifest Exported components checking	The android application found "exported" components(except for Launcher) for receiving outside applications' actions (AndroidManifest.xml). These components may be initialized by other apps so the attribute should be	**Action required:** Modify or add the attribute to [exported="false"] if you don't want to receive outside applications actions**.**
Getting IMEI and Device ID	This app has code getting the "device id(IMEI)" but there are problems with this "TelephonyManager.getDeviceId()" approach	**Action required**
Getting ANDROID_ID	This app has code getting the 64-bit number "Settings.Secure.ANDROID_ID".	**Action required**
WebView Local File Access Attacks Checking	Found "setAllowFileAccess(true)" or not set(enabled by default) in WebView. The attackers could inject malicious script into WebView and exploit the opportunity to access local resources	**Action Required-** Disable local file system access**.**

WebView Potential XSS Attacks Checking	Found "setJavaScriptEnabled(true)" in WebView, which could exposed to potential XSS attacks	**Action Required-** Check webpage code and sanitize the output

FINDBUGS

Findbugs is an open source static code analyzer that is used to detect possible bugs in Java programs. The tool is written in Java language. The tool looks for the potential bugs in the Java code by looking for more than 200 bug patterns. The tool inspects the Java byte code which is in the form of the compiled class files and which is used to detect the occurrences of the bug patterns. It also helps in improving the efficiency at the run time by highlighting different bugs or threats in the bytecode.

HOW THE TOOL WORKS:

1. It basically analyses the Java byte code and specifically .classes to find the potential bugs and the design flaws in the source code.
2. It requires the compiled code to analyze and it works on the byte code level.
3. It basically finds the instances which are related to bug patterns that are, the code instances which are most likely to be errors:
4. The tool operates on the Java byte code rather than operating on the source code.
5. It doesn't focus on the indentation and the naming nomenclature in the source code.
6. The major categories in the reported bugs are as follows:

No.	REPORTED BUG CATEGORIES
1.	Bad Practice
2.	Malicious Code Vulnerability
3.	Correctness
4.	Performance
5.	Security
6.	Dodgy code
7.	Experimental
8.	Multithreaded correctness
9.	Internationalization

PRE-REQUISITES:

The plugins that are required to be installed in the Android studio are :

1. FindBugs-IDE
   This is a find bugs plugin which is used for analyzing the java source code and finding the bugs and reporting it.
2. QAPlug
   QAPlug is an IntelliJ IDEA plugin which is used for managing the code quality that integrates tools such as PMD, Checkstyle and Findbugs.
3. QAPlug- Find Bugs
   QAPlug- Find Bug is a FindBugs module for QAPlug that is an IntelliJ IDEA plugin which is used for managing the code quality that integrates the tools such as PMD, Checkstyle and finds bugs.

Installation steps for Findbugs

1. Download the Android studio. The Android studio used for the source code analysis was as follows:

   Android Studio 3.4.1

   Build #AI-183.6156.11.34.5522156, built on May 2, 2019

   JRE: 1.8.0_152-release-1343-b01 amd64

   JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

   Windows 10 10.0

2. Go to File, then Settings and then Plugins and then download the following plugins:
   1. Findbugs-IDE
   2. QAPlug
   3. QAPlug-Findbugs.
3. Restart the Android Studio and the tool is ready for analyzing the source code.

DRAWBACKS OF THE TOOL

1. It generally requires a lot of memory and a fast CPU.
2. It also detects only a few types of infinite loops.
3. It is also unable to detect the dead variables and dead functions.

POSSIBLE IMPROVEMENTS/SUGGESTIONS

1. The tool should be able to detect the dead variables and dead functions.
2. The Android studio plugin for findbugs can also provide automatic fixes for the common bugs as it would increase the convenience of using it on the Android Studio.

CODE ANALYSIS USING FINDBUGS

In order to do code analysis using find bugs, the source code was downloaded from the GitHub. The complex code was selected from the GitHub. The code that was selected was an Android application named **Android image cropper**. The code was selected on the basis of the following parameters:

Link for the code is: https://github.com/ArthurHub/Android-Image-Cropper

Parameters	Values
Language used	Java(100%)
Last Updated	24/05/2019
STAR	4,521
FORK	927
Number of releases	32
Contributors	44
Watch	164

The above parameters depict the complexity of the source code.

Android-Image-Cropper -SOURCE CODE ANALYSIS

STEPS for the Source code analysis:

4. The source code was downloaded from the GitHub repository.

Then, the code was unzipped and then later it was opened in Android studio.

5. After that, download the following plugins:
    4. Findbugs-IDE
    5. QAPlug
    6. QAPlug-Findbugs.
6. After installing all the required plugins, go to the menu bar and click on the Analyze button--->Analyze code.
7. Once the code is analyzed you will get the result produced by the Findbugs.s

8. After that, you can fix the bugs.
9. The results found after analyzing the code are as follows:

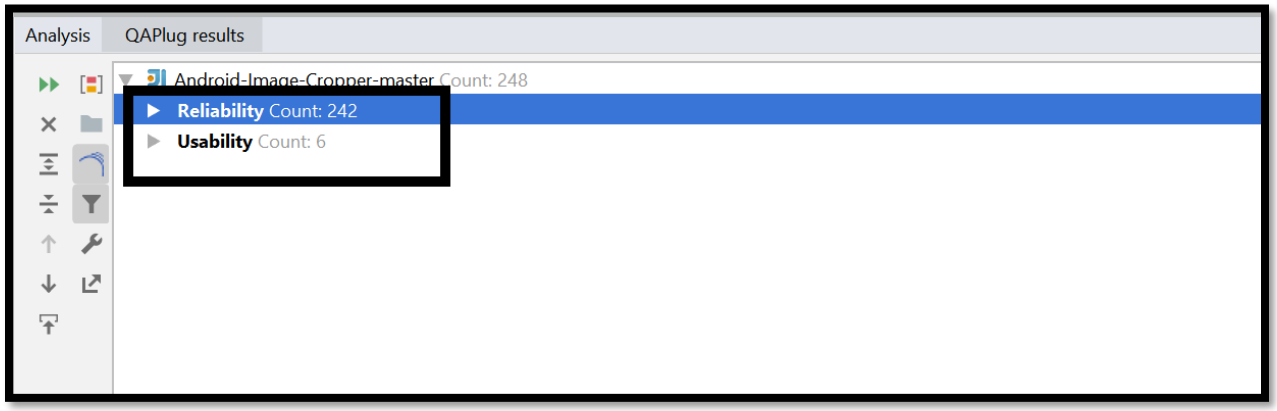

Figure 16: bugs that were produced for Android-Image-Cropper-master

The number of bugs that were found was as follows:

Figure 17: bugs count reported by findbugs

The bugs were classified on the reliability and the usability of the code. Following table shows the categories of the bugs reported:

Bug type	Bug categories	Description	Count
Reliability	Correctness	Method performs math using floating point precision	238
Reliability	Correctness	Null check of the value precisely dereferenced	1
Reliability	Correctness	Uninitialized read of the field in the constructor	2
Reliability	Malicious Code Vulnerability	May expose internal representation by returning a reference to the mutable object	1
Usability	Dodgy	Int division result cast to double or float	1
Usability	Dodgy	Possible null point dereference due to the return value of the called method.	2
Usability	Dodgy	Test for floating point equality.	1
Usability	Dodgy	Write to a static field from an instance method	2

POSSIBLE SOLUTION TO BUGS FOUND-FINDBUGS

The bugs found can be resolved and the method to resolve the bug is mentioned in the following.

1. **Bug type:** Reliability
   Bug Category: Correctness
   Description: Method performs math using floating point precision.

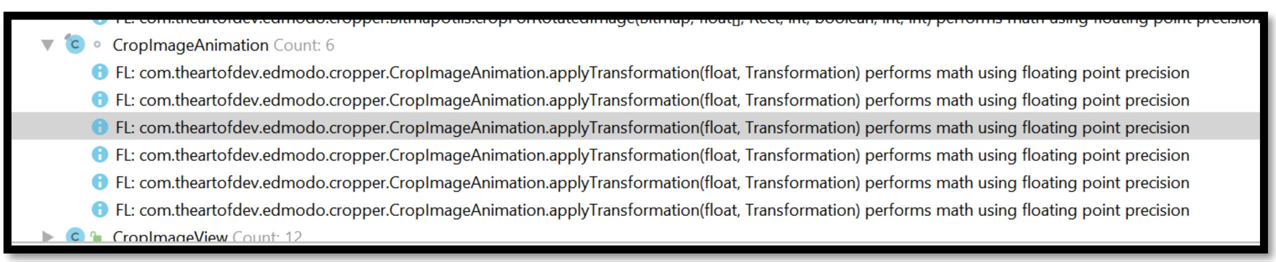

Figure 18: SCREENSHOT OF THE ERROR MESSGE i.e performaa math using Floating point precison in FINDBUGS

Analysis: The method performs math operations using floating point precision. Floating point precision is very imprecise. For example, 16777216.0f + 1.0f = 16777216.0f. Consider using double math instead.

Solution: It is a bad practice to use float in this case. This bug can be resolved by using DOUBLE instead of Float datatype. Double is more precise than a float data type.

2. **Bug type:** Reliability
   Bug Category: Correctness
   Description: Nullcheck of the value precisely dereferenced.

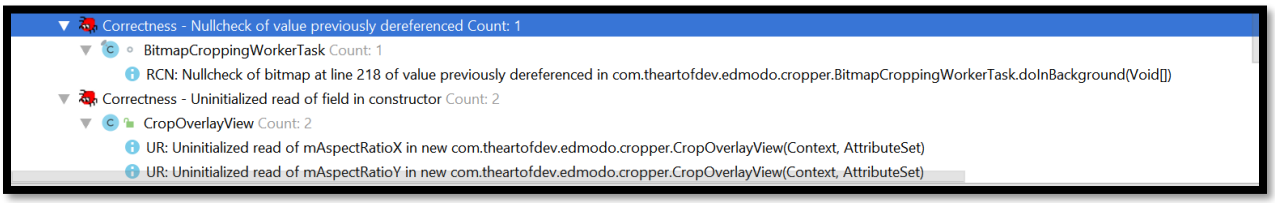

Figure 19: SCREENSHOT OF NULL CHECK for Correctness in FINDBUGS

Analysis: A value is checked here to see whether it is null, but this value can't be null because it was previously dereferenced and if it were null a null pointer exception would have occurred at the earlier dereference. Essentially, this code and the previous dereference disagree as to whether this value is allowed to be null. Either the check is redundant or the previous dereference is erroneous.

Solution: The following message for a null check is shown because the variable with null value is being checked whereas it has already been accessed by the variable before. Therefore, the null check is not needed as the value was null and hence a NULLPOINTException would have been thrown.

3. **Bug type:** Reliability
   Bug Category: Correctness
   Description: Uninitialized read of the field in the constructor.

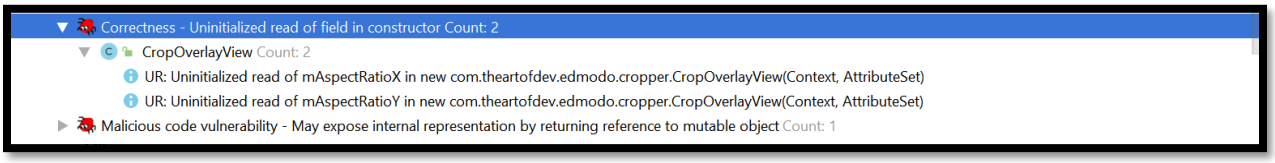

Figure 20: SCREENSHOT OF Uninitialised constructor error for Correctness in FINDBUGS

Analysis: In this, the constructor reads a field which has not yet been assigned a value. This is often caused when the programmer mistakenly uses the field instead of one of the constructor's parameters.

Solution: The data type for the variable mAspectRatioX and mAspectRatioX should be initialized a value i.e 0 at the beginning as follows:

Private int mFixApectRatioX=0;

Private int mFixApectRatioy=0;

```
private boolean mFixAspectRatio;

/** save the current aspect ratio of the image */
private int mAspectRatioX;

/** save the current aspect ratio of the image */
private int mAspectRatioY;

/**
 * The aspect ratio that the crop area should maintain; this variable is only used when
 * mMaintainAspectRatio is true.
 */
private float mTargetAspectRatio = ((float) mAspectRatioX) / mAspectRatioY;

/** Instance variables for customizable attributes */
```

Figure 21: Screenshot for the Bug in the source code

4. **Bug type:** Reliability
   Bug Category: Malicious code vulnerability
   Description: May expose internal representation by returning reference to the mutable object.

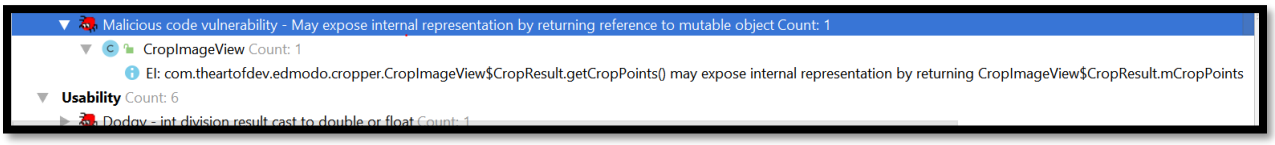

Figure 22: Screenshot for the Malicious code vulnerability Bug in the Findbugs console

Analysis: Returning a reference to a mutable object value stored in one of the object's fields exposes the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important

properties, you will need to do something different. Returning a new copy of the object is a better approach in many situations.

Solution:

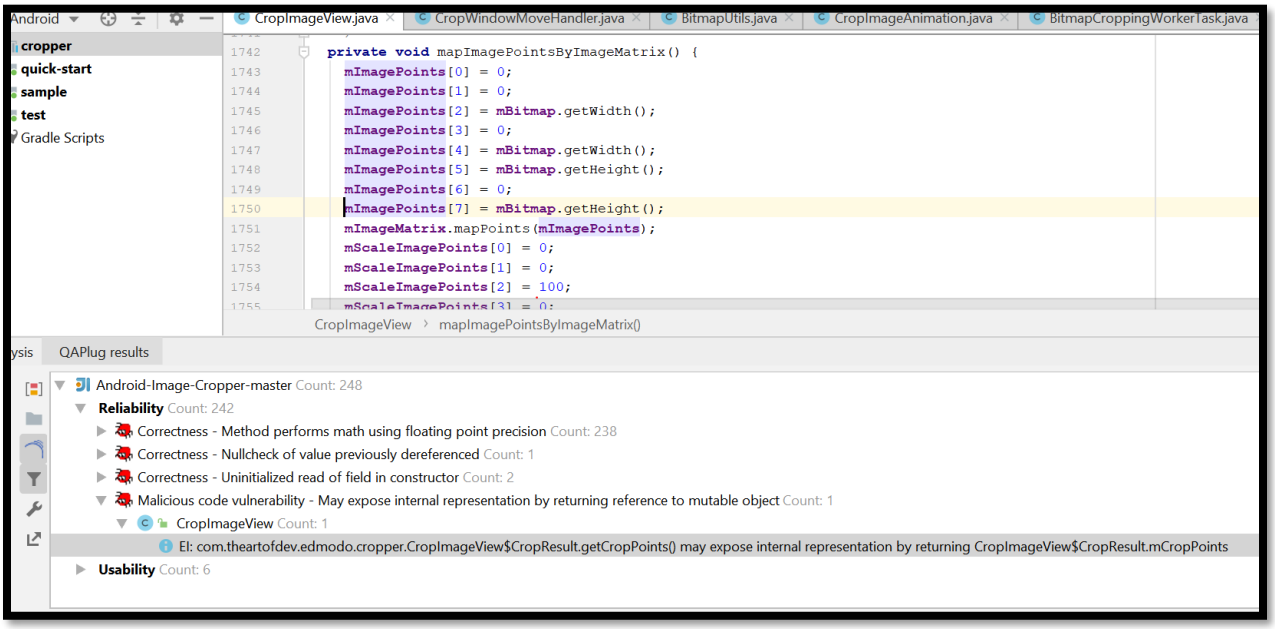

Figure 23 : Screenshot to show the error in the source code

Solution: mImagePoints may use one method to expose another method that is not ideally supposed to be exposed. Hence, the array mImagePoints should not be used by another method as it is private.

5. **Bug type:** Usability
   Bug Category: Dodgy
   Description: Int division result cast to double or float
   Analysis: This code casts the result of an integer division operation to double or float. Doing division on integers truncates the result to the integer value closest to zero. The fact that the result was cast to double suggests that this precision should have been retained. What was probably meant was to cast one or both of the operands to double before performing the division.

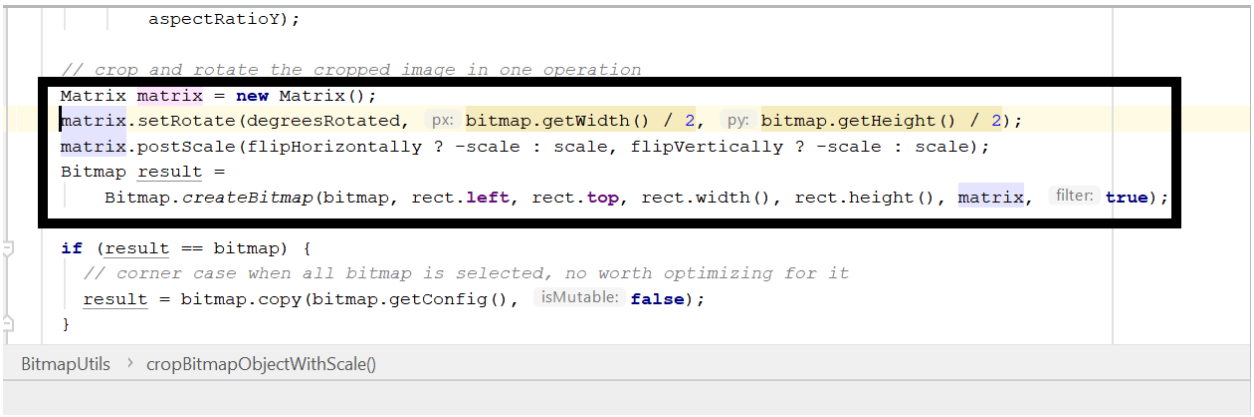

Figure 24: Screenshot for the Bug in the source code

Solution: In this case, the int data type may be used instead of the double or float in order to avoid getting an error.

6. **Bug type:** Usability
   Bug Category: Dodgy
   Description: Possible null point dereference due to the return value of the called method.
   Analysis: The return value from a method is dereferenced without a null check, and the return value of that method is one that should generally be checked for null. This may lead to a NullPointerException when the code is executed.

Figure 25: Screenshot for the Bug in the source code

Solution: A null check has been performed on the given value(i.e mBitmapLoadingWorkerTask).

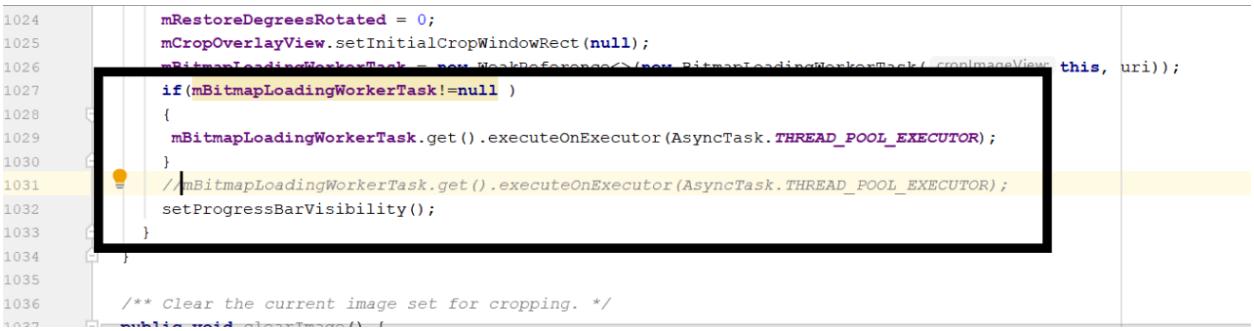

Figure 26: Screenshot for the BUG-FREE code

7. **Bug type:** Usability
   Bug Category: Dodgy
   Description: Test for floating point equality.
   Analysis: This operation compares two floating point values for equality. Because floating point calculations may involve rounding, calculated float and double values may not be accurate. For values that must be precise, such as monetary values, consider using a fixed-precision type such as BigDecimal. For values that need not be precise, consider comparing for equality within some range, for example: if (Math.abs(x - y) < .0000001).

Figure 27: Screenshot for the Bug in the source code

Solution: The data type should be BigDecimal instead of Double in order to get accurate results.

8. **Bug type:** Usability
   Bug Category: Dodgy
   Description: Write to a static field from an instance method
   Analysis: This instance method writes to a static field. This is tricky to get correct if multiple instances are being manipulated, and generally bad practice.

Figure 28: Screenshot for the Bug in the source code

Solution: The class CropImage view can be made static in order to avoid such error.

Figure 29: Screenshot for the Bug-free code in the source code

DIFFERENCE BETWEEN ANDROBUGS AND FINDBUGS

The difference between Androbugs and findbugs is as follows:

Androbugs	Findbugs
Androbugs is an open source static analysis tool which is used to find valid security vulnerabilities in an Android application	Findbugs is also an open source static analysis tool to inspect Java byte code for occurrences of bug patterns.
It is written in Python.	It is written in Java.
It is used for the analysis of an apk file.	It is used for finding bugs in Java programs.
It doesn't scan logical security issues.	It is based on the concept of bugs patterns.
It doesn't support plugin architecture, but you can easily extend new features or vulnerability vectors.	It supports plugin architecture allowing anyone to add new bug detectors.
It doesn't require source code for analysis.	It requires source code for analysis.
It scans for known common coding vulnerabilities to efficiently find bugs.	It scans for the occurrence of the bug patterns.
It categorizes the vulnerability on the basis of severity level i.e Critical, Warning, Notice, and info.	It categorizes the vulnerability on the basis of reported bug categories.
It doesn't require a lot of CPU memory.	It requires a lot of CPU memory.

CONCLUSION

Java source code and an apk file were analyzed using Findbugs and Androbugs framework. The results given by the static analysis tools were conclusive but may not be always correct. There were many warnings which may be avoided but in case of a large application, such warnings may affect the performance and the efficiency of the code and the application. In the case of Android applications, the warnings such as Dynamic code loading may be ignored according to the result given by androbugs as it is not critical according to the androbugs. But such vulnerabilities can be used by an attacker to deploy malware. Hence, the developer can not just ignore the warnings rather the necessary steps should be taken to overcome such warnings in order to secure the android application from vulnerabilities such as code injection. Further, the results generated by the tools were analysed in depth and the security fixes were explained in depth.

REFERENCES

DYDROID : Measuring Dynamic Code Loading and Its Security Implications in Android Applications Zhengyang Qu, Shahid Alam† , Yan Chen, Xiaoyong Zhou, Wangjun Hong, Ryan Riley† Northwestern University, Qatar University , Samsung Research America.

https://www.youtube.com/watch?v=Fc5R88fhR2c

https://www.youtube.com/watch?v=EMjFVoCi2fA

https://www.all-things-android.com/content/deeper-look-android-application-permissions

https://source.android.com/security/app-sandbox

https://github.com/AndroBugs/AndroBugs_Framework

https://tech.co/news/android-security-2014-07

https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

findbugs.sourceforge.net/

https://securityintelligence.com/new-vulnerability-android-framework-fragment-injection/

https://blog.netspi.com/four-ways-bypass-android-ssl-verification-certificate-pinning/

https://pdfs.semanticscholar.org/8f39/2d8c7211558219c3e72b6e05a6bd84796f40.pdf

http://findbugs.sourceforge.net/factSheet.html