e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

AUTOMATIC AND MANUAL DISCOVERY OF VULNERABILITIES IN SELECTED ANDROID MOBILE APPLICATIONS (PENETRATION TESTING APPROACH)

Enoch Owoahene Acheampong*1, Er. Deepinder Kaur*2

*1Student, Department Of Computer Science And Engineering, Shaheed Udham Singh College Of Engineering And Technology, Mohali, Punjab, India.

*2HOD, Guide, Department Of Computer Science And Engineering, Shaheed Udham Singh College Of Engineering And Technology, Mohali, Punjab, India.

ABSTRACT

The majority of mobile phones today are running on the Android OS. Are those applications developed in a way that can be exploited by attackers? Usage of mobile applications have been increasing and their development is leading to increasing threats and vulnerabilities. Many automatic vulnerability scanners and testing tools are available but their results are combined with false positives. In order to result in a vulnerable free source code, one should perform not only automatic testing but also manual testing. In this research four (4) high level android mobile banking applications for four of the top ten banks in Ghana were tested for security vulnerabilities. Our research focused on using an android security framework called Mobexler for automated security testing whiles manual source code review was conducted to supplement the automated testing approach to identify as many vulnerabilities as possible. It was observed during the analysis that m-bank-app1 recorded few vulnerabilities whiles m-bank-app4 recorded the highest number of vulnerabilities the remaining m-bank-app2 and m-bank-app3 also recorded moderately few vulnerabilities. Common security practices were not followed in the aspect of the application permissions defined in the manifest file. There were number of dangerous permissions allowed which is in contrary with android application developer standard. It can be ascertained that m-bank-app4 has the highest number of dangerous permission whiles m-bank-app1 has the lowest number of dangerous permissions.

Keywords: Android OS, Penetration Testing, Android Security, Mobile Application, Vulnerabilities, Security Tools, Mobexler Framework.

I. INTRODUCTION

"Mobile phones are a necessity in our lives and the majority of us have become completely dependent on them in our daily lives. The majority of mobile phones today are running on the Android OS. The main reason for this is the ever-growing community of developers and massive number of applications released for the Android OS. This massive usage is not risk free and the main concern is security" [1]. "One cannot tell whether the applications that are based on the Android operating system are secure. How can a common user tell if the application they are using is not malicious? Are those applications developed in a way that can be exploited by attackers? This is an important question that must be addressed" [1]. Usage of mobile applications have been increasing and their development is leading to increasing threats and vulnerabilities. There are different coding languages and frameworks to build or develop mobile applications, but finding vulnerabilities in the developed code and mobile application is difficult. Many automatic vulnerability scanners and testing tools are available but their results are combined with false positives. In order to result in a vulnerable free secure code, one should perform not only automatic testing but also manual testing.

"Android is one of the most popular smartphone operating systems of the present day, accounting for more than half of the entire smartphone market. It has got a huge consumer base, as well as great support from the developer community resulting in over a million applications in the official Play Store "[1].

"In a relatively short period of time, Android has become the world's most popular mobile platform. Although originally designed for smartphones, it now powers tablets, TVs, and wearable devices, and will soon even be found in cars. Android is being developed at a breathtaking pace, with an average of two major releases per year. Each new release brings a better UI, performance improvements, and a host of new user-facing features which are typically blogged about and dissected in excruciating detail by Android enthusiasts" [2].

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

"With the speed of the technological age and the emergence of modern technologies for electronic marketing, there has been a strong spread of mobile applications, as a number of famous systems have emerged, which have become a large number of applications on the Internet, such as Android applications and iPhone applications and have become widely required in various fields such as education and health fields. Mobile applications (mobile apps) are software programs that are installed and implemented on mobile devices, such as smartphones, tablets, smart TVs, and smart watches. Nowadays the massive development in the smartphone's platforms and the use of mobile applications become increasing significantly according to Digital 2019 Reports" [3], "the total number of mobile users are 5.11 billion in the past year, and that is expected to increase substantially in the near future. In this technological era, smartphone usage and its associated applications are rapidly increasing" [4] "due to the convenience and efficiency in various applications and the growing improvement in the hardware and software on smart devices. It is predicted that there will be 4.3 billion smartphone users by 2023" [4].

Vulnerabilities are a serious problem in system security and information assurance. More data assurance and device protection can be enabled by a vulnerability-free application. Up until now, the need for vulnerability assessment and penetration testing has generally been neglected. By using continuous and appropriate vulnerability assessment, we can reduce the risk of being attacked and provide more stable mobile applications. Apart from the security issues in the Android platform itself, a lot more vulnerabilities exist in the Android application, which could lead to a breach of private data from smartphones.

A penetration testing approach is de-facto way to discover as many vulnerabilities as possible. It is the deliberate way to find weakness in systems and application. Using this approach to find weaknesses in mobile applications is ideal to provide a wider picture of what may or may not be needed. With the term penetration testing we refer to the operational process of analyzing or evaluating the security of a computer system or network. Conducted over several phases, many of them manually performed by security analysts from the point of view of a potential attacker by exactly simulating the cyber-attack of an attacker, it consists in the exploitation of the vulnerabilities detected by helping to determine if the system defenses are sufficient or if other vulnerabilities are present, listing in this case which ones defended the test defeated.

RESEARCH OBJECTIVES

- To ascertain the importance of secure coding in android mobile application development.
- To justify the need for hybrid approach in detecting vulnerabilities in mobile application.
- Which techniques can be used to analyze Android source code to detect vulnerabilities?

II. LITERATURE REVIEW

Over the years there have been a lot of survey and journal published on android mobile security. These surveys have focused on various components of android security which include; android security framework, android permissions, android malware, android hybrid security testing and also building secure system to ensure safe guard of android operating system. In this section, I discuss several papers and articles on various aspects on android devices and security.

Chetan et al. [5] performed a thorough research on exploring security mechanism in android devices. Their research focused on the discussion pertaining to various threats associated with android devices. They further stressed on the ways to strengthen android security mechanism. Again, the various threats that were analyzed had their respective solutions and mitigation techniques. Meanwhile techniques and procedures used to detect these threats were not discussed and no standard industry tools were stated.

Xinhong et al. [6] researched focused on the hardware component of the android operating system. Their research emphasizes the fact that there is no complete set of security controls to ascertain the reliability and integrity of the android system presently. To also ensure the fully fledged security of the android operating system, the security session of mobile terminals should be fully protected from the onset. This is current application layer cannot guarantee the security of the android system.

Durmuş and Hande [7] used an open-source security testing platform to test android operating system. In their research security mechanism was examined with the operating system framework, and possible exploitation of android security weaknesses were created. Their research aimed to inform developers and android users on

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

the possible attacks by examining the mobile attacks and security. Moreover, their research was limited to the android operating system security implications but did not emphasize on security challenges on the android mobile applications.

Ashwag et al. [8] surveyed on reverse engineering tools for android devices. The survey focused on the various reverse engineering tools in terms of methodology and their usage in developing secure applications. The survey further discussed how these tools are used to exploit weakness in android devices. It did not stress on how android mobile applications can be protected and ways to ensure continuous protection from attacks. Again, the survey did not mention the industry tools ideal for reverse engineering android applications.

Cheolmin and YoojaeWon [9] also discussed vulnerability evaluation method through correlation analysis of android applications. Their research proposed a process to automate de-compilation all users' applications on a device and a control process to detect inter-application vulnerabilities. They further discuss how developers carelessly introduced vulnerabilities into mobile application and most applications vulnerabilities arise through inter-application communication. The research identified security gap in the mobile application fraternity. Existing studies were not able to detect security loopholes outside an application. Their findings also visualized the inter-application risks in a matrix. Meanwhile the research also had a gap in which the proposed mechanism did not check for current vulnerabilities in mobile applications.

Ratul et al. [10] performed a survey on android security. Their research focused on the development and deployment hindrance and best practices. They also stressed on the point that vulnerabilities in android OS arise with respect to the increasing functionality, careless practices of developers and end users heedless. Their research also gave a wide view of android system level development, issues with privacy and guidelines for app developers on security measures to consider during app development. They also recommended that android smartphone manufacturers and Google android development team should collaborate properly in defined protocol to ensure timely security patches release.

Parag and Pandya [11] conducted a critical study on android permission related vulnerabilities. The paper concentrated on security weaknesses and privacy related issues. Their paper discussed android permissions and possible cyber-related crimes. They also emphasized on permission abuse of applications for hacking user's sensitive data. The paper focused on android security permissions as an integral threat to end-users.

Mishra et al. [12] proposed a privacy protection framework for android. In their paper a secure privacy protection framework was recommended to prevent an app from requesting for unnecessary permissions. Since most applications request for not so importance permission, bad actors also prey on this to steal end-users' sensitive data. Android OS uses system of permissions to protect users' data and also provide security.

Ramírez-López et al. [13] proposed a framework to secure the development and auditing of ssl pinning in mobile applications and devices. They argued on ssl pinning well-known in the professional world but unusual to see them in academia. Their research also explained on how an app can be toughen reviewing security measures, guarding some aspects to avoid attacks. Their framework was designed based on OWASP Mobile Application Security Verification Standard. (MASVS). The paper also proposed two new bypassing techniques.

Ikram and Tamim [14] presented a research paper based on penetration testing frameworks and development issues in secure mobile application development in a systematic review. Their paper highlighted a broad survey of various penetration testing tools, analyzed by using standard such as code review, code analysis, vulnerability analysis, vulnerability exploit, payload and whether these can be used in vulnerability modeling during the design phase. They also argued that inexperienced app developers do not possess secure coding knowledge of the latest android vulnerabilities and expand application attack surface that bad actors exploit.

Janaka et al. [15] provided a systematic review of machine learning android malware detection techniques. The research paper systematically analyzed carefully selected 106 articles and pointed out their strengths and weaknesses as well as required improvements. Their paper aimed to assist researchers to gain in-depth knowledge and further identify potential future findings and development. They argued that among the numerous countermeasures to detect malware, machine learning techniques have proven to be an effective way of detecting malware attacks since there are derived from a training example which eradicates the need for explicit definition of signatures when developing malware detectors.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

Rana et al. [16] paper also proposed a hybrid detection system that effectively utilize both static and dynamic analysis to detect ransomware with high accuracy. They argued that existing solutions mainly used static analysis. They further stressed that there were limited procedures used for dynamic analysis specifically for ransomware detection. The performance of these techniques mostly fails because of code obfuscation techniques and benign applications that used cryptography methods for their API usage.

Palacios et al [17] proposed mobile application security protocol using OWASP mobile security project. The protocol assists in the view of the security of the mobile application through the implementation of this protocol, security loopholes can be detected in order to correct them later in the application development cycle. The proposed protocol also serves as a guide to developers and security teams as a reference manual they require to build secure applications. The protocol also seeks to classify mobile application security risks for further and detailed reference. Meanwhile the defined protocol does not define the procedure or tools required for mobile application security testing.

Amin et al. [18] paper focused building a hybrid approach of static and dynamic analysis for detecting the vulnerabilities of Android applications. They were motivated with the notion that with the increasing production of android application the testing phase takes less time and security sometimes is completely neglected.

Lima et al. [19] conducted comprehensive research on protecting and monitoring mobile device assets by discussing mobile device security management. Their research paper emphasized on analyzing the preventive solutions of the mobile devices management. The aim was to analyze available security measures and control technologies for maximum security protection on android devices for private data and applications used by businesses.

Zachariah et al. [20] surveyed virus detection methods for android OS based on static analysis. These included (i.e., permission-based detection, signature-based detection, and Dalvik bytecode detection). They further discussed the advantages and disadvantages of their techniques. They did not stress on any future work for the total number of 27 studies that were included.

Demissie et al. [21] used static and dynamic taint analysis with the primary aim of detecting secured-based vulnerabilities. Considerably, taint analysis usually extracts data dependencies among request data from other applications and data used in primary operations. As a basic function in Android, sender requesting an operation from another application may not always cause a weakness. To reduce false alarms, a specific analysis method should differentiate between expected permission and actual permission re-authorization flaws.

Bhat and Dutta [22] classified intrusions on the applications of Android into major groups: hardware abstraction layer-based attacks, hardware-based attacks, Linux-Shell based attacks, and application-based attacks., their research focused on a variety of threats and the procedures that can be taken in terms of security within these groups., They have also analyzed and summarized the most interrupting problems in Android applications. A comparative study was also made between different methodologies used in detecting various malicious activities, and their weaknesses were highlighted too.

Ahmed and Sallow [23] discussed the present Android security threats and the present security proposed solutions and attempted to classify the proposed solutions and evaluate them., their proposed solution is divided into two groups such as static and dynamic satisfying three goals: assessment, analysis, and detection., this review paper aimed for expanding the coverage of malicious application growth and Android security threats.

Guyton [24] described the threats faced by mobile devices running Android in terms of malware types and many other well-known attacks, such as Trojan, Botnet, Backdoor, Worm, Spyware, Ransomware, and Aggressive Adware., The authors used four concepts like Support Vector Machines (SVMs), Self-Organizing Maps (SOMs), decision trees, and Bayesian networks., Those methods can be utilized for malware detection analysis that aims at combating Android malware.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

III. METHODOLOGY

The methodology of this project lies on the selected android mobile applications(apk) and the selected framework for the analysis. The selected framework comprises of three phases which attribute the test environment, the installation processes on the emulator (Genymotion), as well as download of the latest APK from Google Play Store. APK files contain vital information of each selected mobile application that can assist in conducting the analysis process. A selection criterion for the applications and tools used in this study is provided. Finally, setup processes to develop the test environment and analysis result comparison are provided. More details of these approaches are presented in the subsequent subsections.

1. Introductory Stage

This thesis focuses on using Mobexler Mobile Application Penetration Testing Framework for verifying exploitable security vulnerability in four (4) selected android mobile applications.

Figure 1: An image of Mobexler Framework

2. Overview

As shown in Figure 12, Mobexler framework integrates various components that work cohesively such as the static code analyzer, Android emulator system, Android Debug Bridge Interface, dynamic code analyzer.

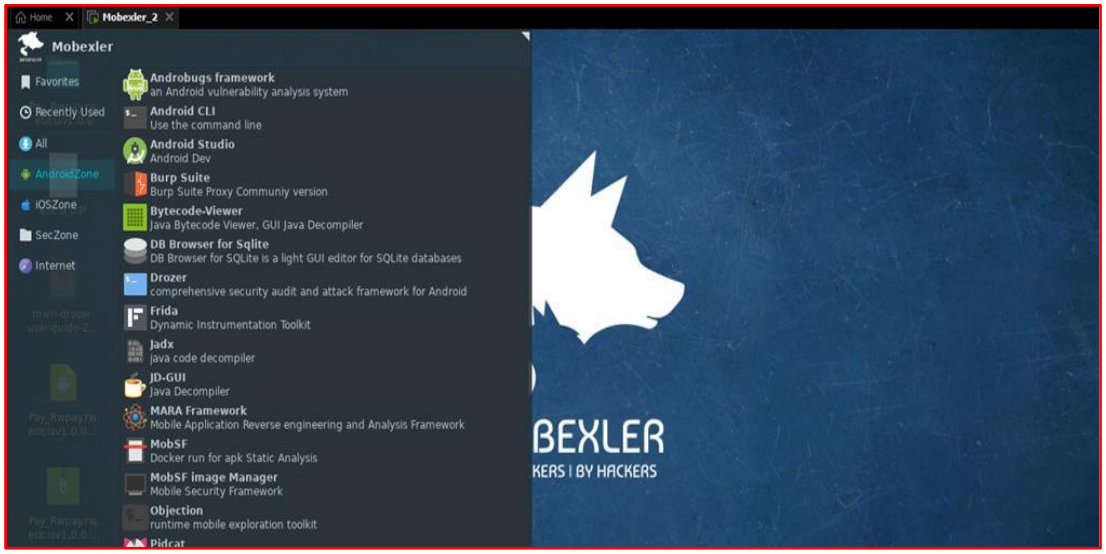

Figure 2: An image of Android zone security tools

3. APK files and Information Gathering

The apk files of the selected android applications will be downloaded from google playstore. The apk files will then be extracted from the application using apps backup restore application. The files will then be installed on an android emulator called Genymotion. As shown below in Figure 3.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

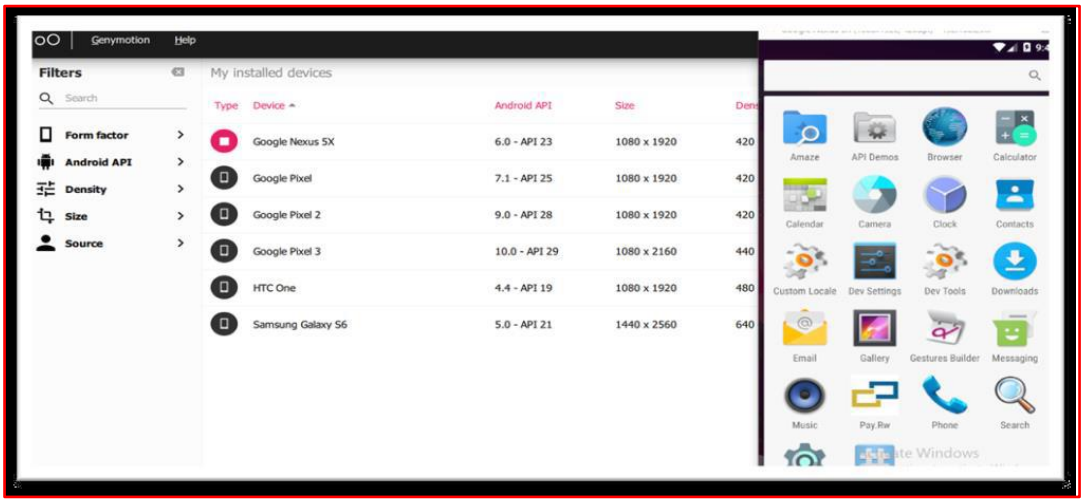

Figure 3: Various android devices installed on Genymotion

4. Selection Criteria

Applications and tools are selected based on specific selection process that will fit the scope of the study.

5. Selection criteria for android mobile applications.

These selected android mobile applications are banking apps for four famous banks in Ghana. These banks are among the top ten financial institutions in Ghana. These selections as a case study fit into the selection criteria for this project to be a success. These banking applications are chosen because of its popularity and customer base. For security purposes and not to release any sensitive information to the public domain, these applications are named as follows;

- m-bank-app1
- m-bank-app2
- m-bank-app3
- m-bank-app4

6. Selection criteria for security tools.

For effective analysis of the selected mobile applications open-source security tools are used for this project. These tools work seamlessly with the APK files. Below tools in table 2 are the chosen tools for activity.

Table 1: Various open-source security tools for android application testing

	Tool	Purpose(s)
1	MobSF	All-in-one automated tool for static, dynamic and malware analysis
2	APKTool	Reverse engineering tool
3	Drozer	Mobile application dynamic analysis security tool
4	Frida	Mobile application hooking tool
5	Burpsuite	Manual and automated penetration testing tool
6	Logcat	Application package log analysis tool
7	Inspeckage	Dynamic analysis tool for android applications

These tools are chosen because of its popularity and effectiveness to perform android penetration testing. Selection criteria also considered tools availability and community support. Frequent and timely update.

7. Setup and Analysis

- The APK package file contains the code of the application. (It is similar to zip file). It contains assets, res, lib, META-INF, AndroidManifest.xml, classes. Dex (Dalvik binary bytecode). To see all these contents the file

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

extension .apk will be converted to .zip extension. This process will enable us to unzip the file to view all the contents.

- To be able to read the AndroidManifest.xml, an APKTool will be installed to convert the Manifest.xml file to a readable format.
- After getting the java source code, it will then be analyzed manually for sensitive information hardcoded into code like password, API keys etc. To also speed up the process MobSF will also be used for the static analysis.

8. Manual review of AndroidManifest.xml

- Every app project must have an AndroidManifest.xml file at the root of the project source set. The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play. Among many other things, the manifest file is required to declare the following:
- The components of the app, which include all activities, services, broadcast receivers, and content providers. It can also declare capabilities such as which device configurations it can handle, and intent filters that describe how the component can be started.
- The permissions that the app needs in order to access protected parts of the system or other apps. It also declares any permissions that other apps must have if they want to access content from this app.
- The hardware and software feature the app requires, which affects which devices can install the app from Google Play.
- Manifest.xml as described above is the backbone of any android application. Since automated tools miss sensitive features, manual verification will be used to check state and permissions declared in the file. Some components that will be checked include;
- Check android: debuggable attribute in AndroidManifest.xml file.
- Typo in using custom permissions which don't match declared custom permissions.
- Test for Exported Activity.
- Check for the exported broadcast receiver in the AndroidManifest.xml file.

9. Dynamic Analysis

This approach will also be used to check for vulnerabilities when the application runs in real-time. Most security misconfigurations can be detected when the application is in the dynamic mode.

Some of the components that will be look up for include;

Table 2: Android mobile application dynamic analysis test components

Analyzing App logs using pidcat tool	Test for Intent Sniffing
Check app UI is protected against a screenshot of sensitive information	Test for Deep Linking Vulnerabilities
Check root detection mechanism is implemented	Test for WebView Vulnerabilities
Check shared preferences for persistent login	Test for local encryption issues
Checking for Keyboard cache using SQLite browser	Finding Storage issues from android manifest file
Test for vulnerable Broadcast receivers	Check location for the secrets
Check for Input Validation issues	Check for Access Control issues

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

IV. MODELING AND ANALYSIS

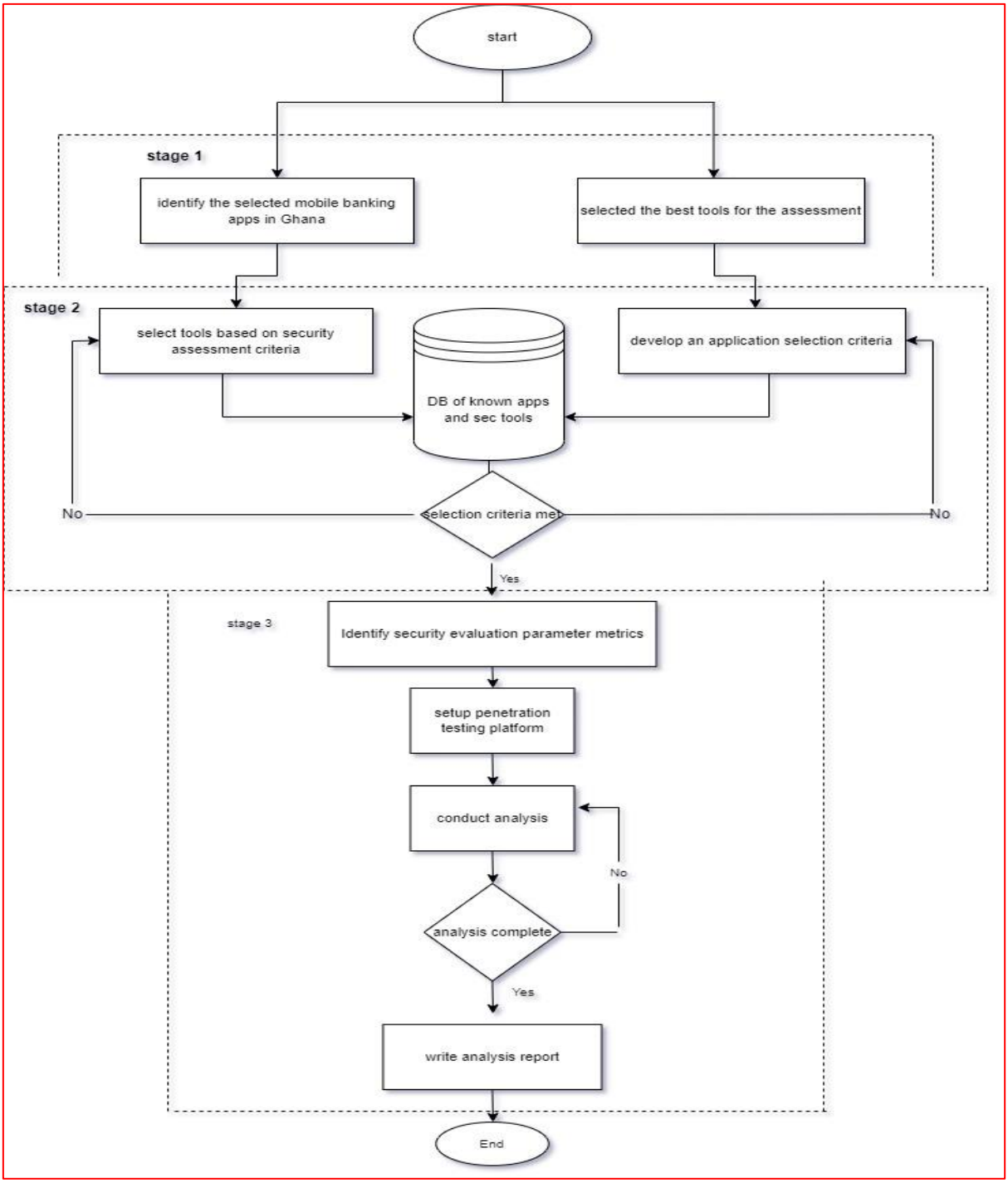

Figure 4: Operational Framework Diagram

V. RESULTS AND DISCUSSION

The results obtained during static and dynamic analysis are discussed in this section. The summarized results for security assessment tool MobSF for all mobile application discussed is presented in the below table 4. MobSF is a famous and open-source tool for static security assessment for android application. Its analysis all aspect of android manifest file and present the result against various standard and benchmark. There were considerable number of identified vulnerabilities that range from low to critical severity level. Identified vulnerabilities were assessed based of CVSS which common vulnerability scoring sheet a standard for rating observable severity of a vulnerability of a system under investigation. MobSF gives a security view of all the

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

identified vulnerabilities in the apk. It further rates various vulnerabilities against CVSS and OWASP Top Ten mobile application vulnerabilites. After the analysis the application is given a scoring sheet on a scale of 100 to ascertain the severity of the vulnerabilities.

1. m-bank-app1

This section discusses the various findings for m-bank-app1.The findings emphasizes on vulnerabilities based on application permissions, exported activities, broadcast receivers and content providers. Source code review best practices were also considered. Table 3,4,5 below summarizes vulnerabilities based on application analysis.

a. Application Permissions

Table 3: m-bank-app1 application permission

Issue	Severity	CVSS	Description
Android Permission camera is set to true	dangerous	7.5	Allows application to take pictures and videos with the camera. This allows the application to collect images that the camera is seeing at any time.

b. Manifest Analysis

Table 4: m-bank-app1 application manifest file analysis

Issue	Severity	CVSS	Description	Standard
Files may contain hardcoded sensitive information like usernames, passwords, keys etc.	High	7.4	There is hardcoded sensitive information available in the source code	CWE: CWE-312 Cleartext Storage of Sensitive Information OWASP Top 10: M9: Reverse Engineering OWASP MASVS: MSTG-STORAGE-14
App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also, sensitive information should be encrypted and written to the database.	Medium	5.9	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	CWE: CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') OWASP Top 10: M7: Client Code Quality
The App uses the encryption mode CBC with PKCS5/PKCS7 padding. This configuration is vulnerable to padding oracle attacks.	High	7.4	Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking	CWE: CWE-649 Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking OWASP Top 10: M5: Insufficient Cryptography OWASP MASVS: MSTG-CRYPTO-3

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

c. Code Analysis

Table 5: m-bank-app1 application source code analysis

Issue	Severity	CVSS	Description	Standard
App can read/write to External Storage. Any App can read data written to External Storage.	High	7.6	Insecure Data Storage	CVSS V2: 5.5 (medium) CWE: CWE-276 Incorrect Default Permissions OWASP Top 10: M2: Insecure Data Storage OWASP MASVS: MSTG-STORAGE-2
App creates temp file. Sensitive information should never be written into a temp file.	Medium	5.5	Insecure Data Storage	CVSS V2: 5.5 (medium) CWE: CWE-276 Incorrect Default Permissions OWASP Top 10: M2: Insecure Data Storage OWASP MASVS: MSTG-STORAGE-2
Insecure WebView Implementation. WebView ignores SSL Certificate errors and accept any SSL Certificate. This application is vulnerable to MITM attacks.	High	7.4	Improper Certificate Validation	CVSS V2: 7.4 (high) CWE: CWE-295 Improper Certificate Validation OWASP Top 10: M3: Insecure Communication OWASP MASVS: MSTG-NETWORK-3

2. m-bank-app2

This section discusses the various findings for m-bank-app2.The findings emphasizes on vulnerabilities based on application permissions, exported activities, broadcast receivers and content providers. Source code review best practices were also considered. Table 6,7,8,9 below summarizes vulnerabilities based on application analysis.

a. Application Permissions

Table 6: m-bank-app2 application permissions

Issue	Severity	CVSS	Description
android.permission.CALL_PHONE | directly call phone numbers.	dangerous	7.5	Allows the application to call phone numbers without your intervention. Malicious applications may cause unexpected calls on your phone bill. Note that this does not allow the application to call emergency numbers.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

android.permission.CAMERA | take pictures and videos.	dangerous	7.5	Allows application to take pictures and videos with the camera. This allows the application to collect images that the camera is seeing at any time.
android.permission.WRITE_EXTERNAL_STORAGE | read/modify/delete external storage contents.	dangerous	7.5	Allows an application to write to external storage.

b. Network Security

Table 7: m-bank-app2 application network security

Issue	Severity	CVSS	Description
localhost [redacted] 0.0.0.0	medium	6.0	Domain config is insecurely configured to permit clear text traffic to these domains in scope.

```
<?xml version="1.0" encoding="UTF-8"?>
- <network-security-config>
  - <domain-config cleartextTrafficPermitted="true">
      <domain includeSubdomains="true">localhost</domain>
      <domain includeSubdomains="true">[redacted]</domain>
      <domain includeSubdomains="true">0.0.0.0</domain>
    </domain-config>
  </network-security-config>
```

Figure 5: cleartext Traffic Permitted on m-bank-app2

c. Manifest Analysis

Table 8: m-bank-app2 application manifest file analysis

Issue	Severity	CVSS	Description
Application Data can be Backed up [android: allowBackup] flag is missing.	medium	5.5	The flag [android: allowBackup] should be set to false. By default, it is set to true and allows anyone to back up your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.
Activity(de.niklasmerz.cordova.biometric.BiometricActivity) is not Protected. [android: exported=true]	high	8.0	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.
Broadcast Receiver (nl. xservices.plugins.ShareChooserPendingIntent) is not Protected. An intent-filter exists.	high	8.0	A Broadcast Receiver is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

			The presence of intent-filter indicates that the Broadcast Receiver is explicitly exported.
Activity (androidx.biometric.DeviceCredentialHandlerActivity) is not Protected. [android:exported=true]	high	8.0	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.

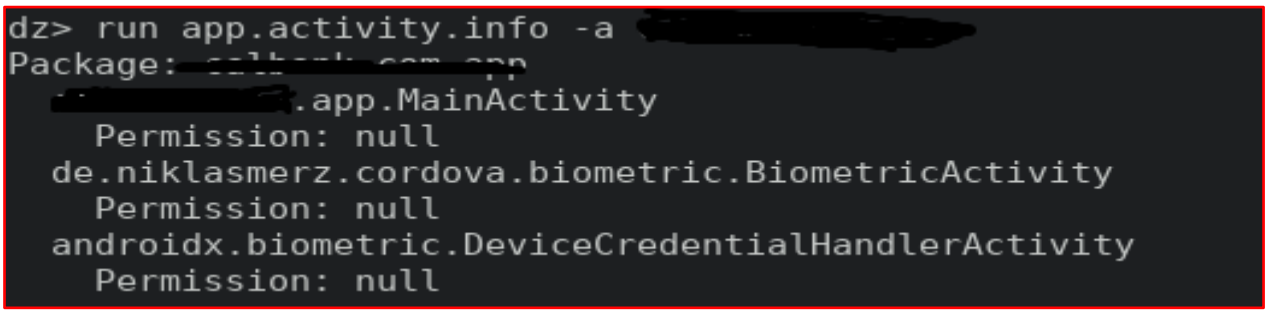

Figure 6: Exported activities without permission set

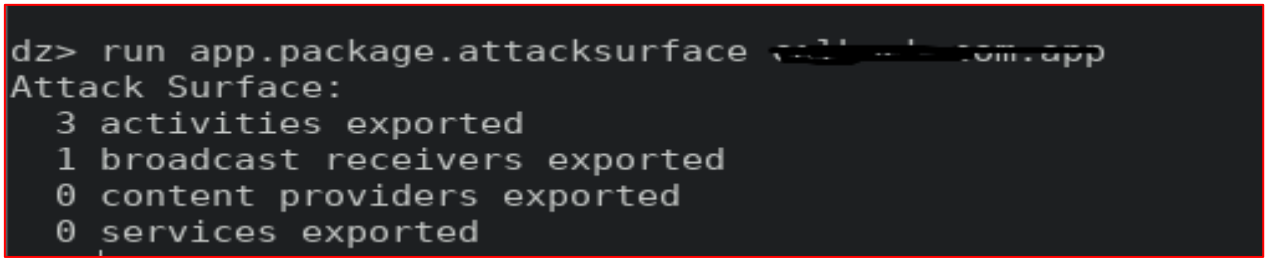

Figure 7: Attack surface analysis using Drozer

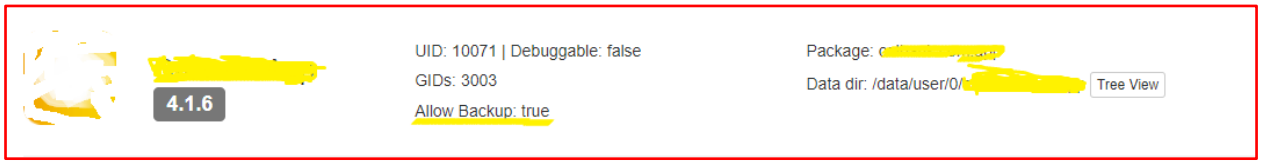

Figure 8: Application backup is set to true

d. Code Analysis

Table 9: m-bank-app2 application source code analysis

Issue	Severity	CVSS	Description	Standard
Files may contain hardcoded sensitive information like usernames, passwords, keys etc.	High	7.4	Reverse engineering of apk revealed sensitive information	CVSS V2: 7.4 (high) CWE: CWE-312 Cleartext Storage of Sensitive Information OWASP Top 10: M9: Reverse Engineering OWASP MASVS: MSTG-STORAGE-14
The App uses an insecure Random Number Generator.	high	7.5	Insufficient cryptography	CVSS V2: 7.5 (high) CWE: CWE-330 Use of Insufficiently Random

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

				Values OWASP Top 10: M5: Insufficient Cryptography OWASP MASVS: MSTG-CRYPTO-6
SHA-1 is a weak hash known to have hash collisions.	medium	5.9	Insufficient cryptography	CVSS V2: 5.9 (medium) CWE: CWE-327 Use of a Broken or Risky Cryptographic Algorithm OWASP Top 10: M5: Insufficient Cryptography OWASP MASVS: MSTG-CRYPTO-4
App can read/write to External Storage. Any App can read data written to External Storage.	high	7.5	Insecure data storage	CVSS V2: 5.5 (medium) CWE: CWE-276 Incorrect Default Permissions OWASP Top 10: M2: Insecure Data Storage OWASP MASVS: MSTG-STORAGE-2

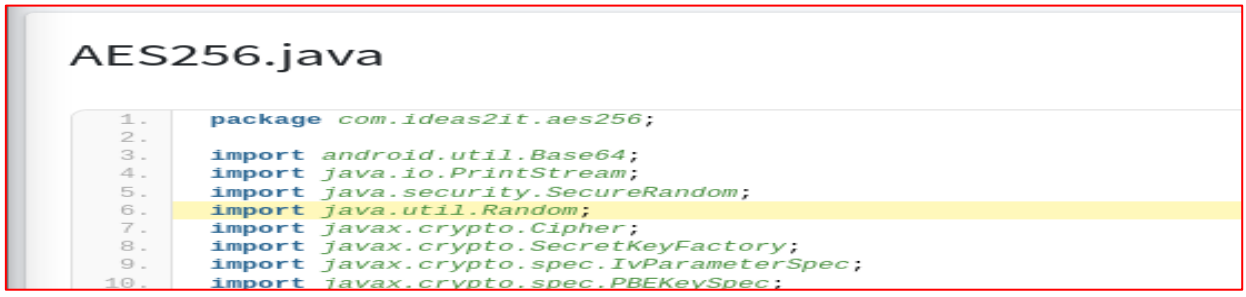

Figure 9: Insecure random generator imported

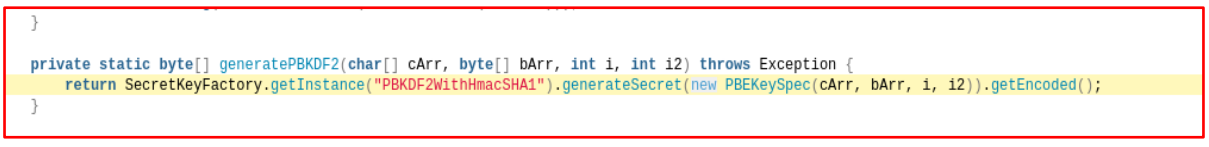

Figure 10: Sha1 algorithm used in the application

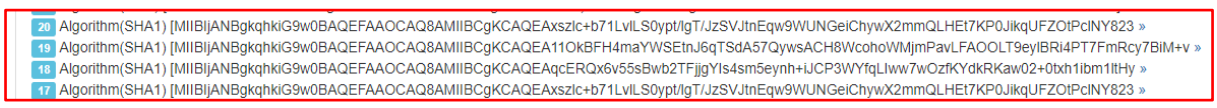

Figure 11: Sha1 algorithm displayed during dynamic analysis

3. m-bank-app3

This section discusses the various findings for m-bank-app3.The findings emphasizes on vulnerabilities based on application permissions, exported activities, broadcast receivers and content providers. Source code review best practices were also considered. Table 10 and 11 below summarize vulnerabilities based on application analysis.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

a. Application Permissions

Table 10: m-bank-app3 application permissions

Issue	Severity	CVSS	Description
android.permission.AUTHENTICATE_ACCOUNTS | act as an account authenticator	dangerous	7.5	Allows an application to use the account authenticator capabilities of the Account Manager, including creating accounts as well as obtaining and setting their passwords.
android.permission.CALL_PHONE |directly call phone numbers	dangerous	7.5	Allows the application to call phone numbers without your intervention. Malicious applications may cause unexpected calls on your phone bill. Note that this does not allow the application to call emergency numbers.
android.permission.GET_ACCOUNTS | list accounts	dangerous	7.5	Allows access to the list of accounts in the Accounts Service.
android.permission.MANAGE_ACCOUNTS | manage the accounts list	dangerous	7.5	Allows an application to perform operations like adding and removing accounts and deleting their password.
android.permission.USE_CREDENTIALS | use the authentication credentials of an account	dangerous	7.5	Allows an application to request authentication tokens.

b. Manifest Analysis

Table 11: m-bank-app3 application manifest file analysis

Issue	Severity	CVSS	Description
Application Data can be Backed up | [android:allowBackup=true]	medium	5.5	This flag allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.
(ma.bits.cbg_eps_mobilebankingapp.MBAuthenticatorService) is not Protected. An intent-filter exists.	high	7	A Service is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. The presence of intent-filter indicates that the Service is explicitly exported.

Figure 12: application allowBackup set to true

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

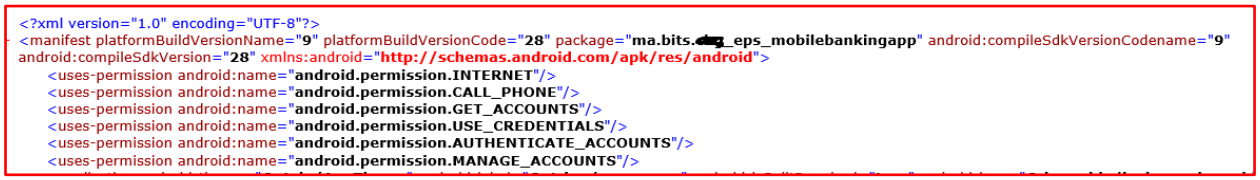

Figure 13: application permission allowed

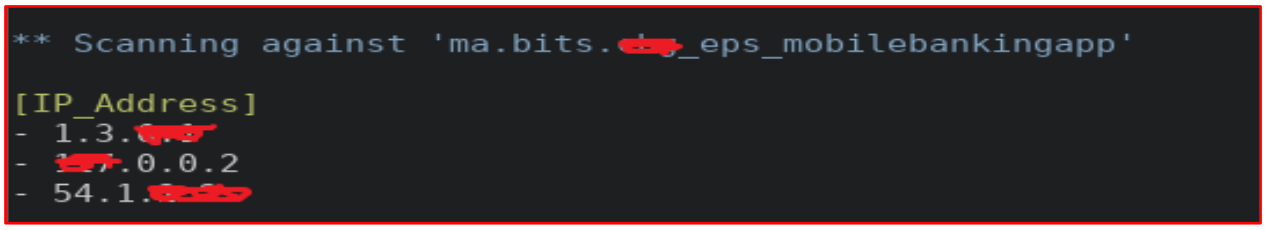

Figure 14: application apkleaks

4. m-bank-app4

This section discusses the various findings for m-bank-app4.The findings emphasizes on vulnerabilities based on application permissions, exported activities, broadcast receivers and content providers. Source code review best practices were also considered. Table 12,13 and 14 below summarize vulnerabilities based on the mobile application analysis.

a. Application Permissions

Table 12: m-bank-app4 application permissions

Issue	Severity	CVSS	Description
android.permission.CAMERA dangerous | take pictures and videos	dangerous	7.5	Allows application to take pictures and videos with the camera. This allows the application to collect images that the camera is seeing at any time.
android.permission.READ_CONTACTS | read contact data |	dangerous	7.5	Allows an application to read all of the contact (address) data stored on your phone. Malicious applications can use this to send your data to other people.
android.permission.ACCESS_FINE_LOCATION | fine (GPS) location	dangerous	7.5	Access fine location sources, such as the Global Positioning System on the phone, where available. Malicious applications can use this to determine where you are and may consume additional battery power.
android.permission.READ_PHONE_STATE | read phone state and identity	dangerous	7.5	Allows the application to access the phone

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

			features of the device. An application with this permission can determine the phone number and serial number of this phone, whether a call is active, the number that call is connected to and so on.
android.permission.READ_EXTERNAL_STORAGE | read external storage contents	dangerous	7.5	Allows an application to read from external storage.
android.permission.WRITE_EXTERNAL_STORAGE | read/modify/delete external storage contents	dangerous	7.5	Allows an application to write to external storage.
android.permission.AUTHENTICATE_ACCOUNTS | act as an account authenticator	dangerous	7.5	Allows an application to use the account authenticator capabilities of the Account Manager, including creating accounts as well as obtaining and setting their passwords.

b. Manifest Analysis

Table 13: m-bank-app4 application manifest file analysis

Issue	Severity	CVSS	Description
Clear text traffic is Enabled for App [android: usesCleartextTraffic=true]	medium	5.5	The app intends to use cleartext network traffic, such as cleartext HTTP, FTP stacks, DownloadManager, and MediaPlayer. The default value for apps that target API level 27 or lower is "true". Apps that target API level 28 or higher default to "false". The key reason for avoiding cleartext traffic is the lack of confidentiality, authenticity, and protections against tampering; a network attacker

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

			can eavesdrop on transmitted data and also modify it without being detected
Activity(com.mode.****.MoreServices.NotificationActivity) is not Protected.[android:exported=true]	high	7.8	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.
Activity (com.mode.****. Postlogin.CreditOneAcc) is not Protected. [android:exported=true]	high	7.8	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.
Activity (com.mode.*****.Postlogin.Creditform) is not Protected.[android:exported=true]	high	7.8	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.

c. Code analysis

Table 14: m-bank-app4 application source code analysis

Issue	Severity	CVSS	Description	Standard
Files may contain hardcoded sensitive information like usernames, passwords, keys etc..	High	7.4	Clear storage of sensitive information	CVSS V2: 7.4 (high) CWE: CWE-312 Cleartext Storage of Sensitive Information OWASP Top 10: M9: Reverse Engineering OWASP MASVS: MSTG-STORAGE-14
App can read/write to External Storage. Any App can read data written to External Storage.	Medium	5.5	Insecure Data Storage	CVSS V2: 5.5 (medium) CWE: CWE-276 Incorrect Default Permissions OWASP Top 10: M2: Insecure Data Storage OWASP MASVS:

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

				MSTG-STORAGE-2
The App uses an insecure Random Number Generator.	High	7.5	Insecure cryptography	CVSS V2: 7.5 (high) CWE: CWE-330 Use of Insufficiently Random Values OWASP Top 10: M5: Insufficient Cryptography OWASP MASVS: MSTG-CRYPTO-6
IP Address disclosure	medium	4.3	Information disclosure	CVSS V2: 4.3 (medium) CWE: CWE-200 Information Exposure OWASP MASVS: MSTG-CODE-2

```
Data Directory: /data/data/com.mode.[redacted]
APK Path: /data/app/com.mode.[redacted]-1/base.apk
UID: 10051
GID: [3003, 3002, 1028, 1015]
Shared Libraries: null
Shared User ID: null
Uses Permissions:
- android.permission.CAMERA
- android.permission.INTERNET
- android.permission.NFC
- android.permission.BLUETOOTH
- android.permission.ACCESS_NETWORK_STATE
- android.permission.READ_CONTACTS
- android.permission.ACCESS_FINE_LOCATION
- android.permission.READ_PHONE_STATE
- android.permission.READ_EXTERNAL_STORAGE
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.AUTHENTICATE_ACCOUNTS
- android.permission.USE_BIOMETRIC
- android.permission.USE_FINGERPRINT
- android.permission.VIBRATE
- android.permission.WAKE_LOCK
- com.google.android.c2dm.permission.RECEIVE
- com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE
Defines Permissions:
```

Figure 15: application permissions allowed

```
Package: com.mode.[redacted]
  com.mode.[redacted].preloginservice.VideoClass
    Permission: null
  com.mode.[redacted].MoreServices.NotificationActivity
    Permission: null
  com.mode.[redacted].Postlogin.CreditOneAcc
    Permission: null
  com.mode.[redacted].Postlogin.Creditform
    Permission: null
  com.mode.[redacted].BankingService.CardlessWithdrawlist
    Permission: null
  com.mode.[redacted].BankingService.ViewCardless
    Permission: null
  com.mode.[redacted].BankingService.ViewCardlessform
    Permission: null
  com.mode.[redacted].BankingService.CardReplacement
    Permission: null
  com.mode.[redacted].BankingService.AddDevice
    Permission: null
  com.mode.[redacted].MoreServices.ViewSOSamebankForm
    Permission: null
  com.mode.[redacted].BankingService.PFM
    Permission: null
  com.mode.[redacted].Postlogin.HideAccount
    Permission: null
```

Figure 16: application exported activities set to true

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

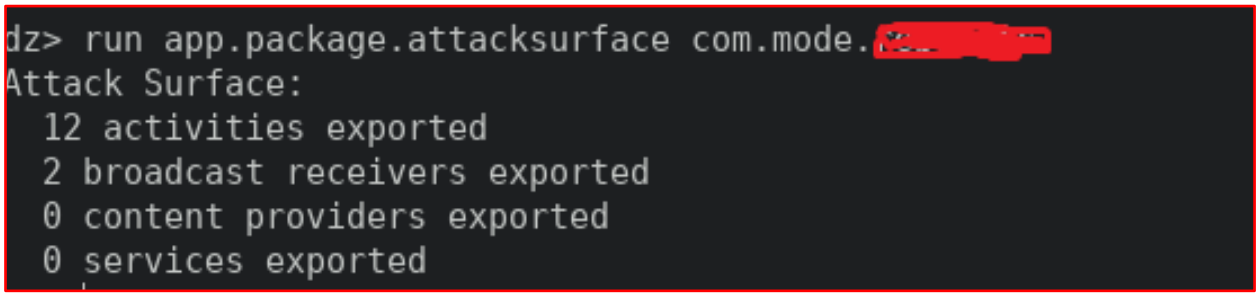

Figure 17: application attack surface analysis

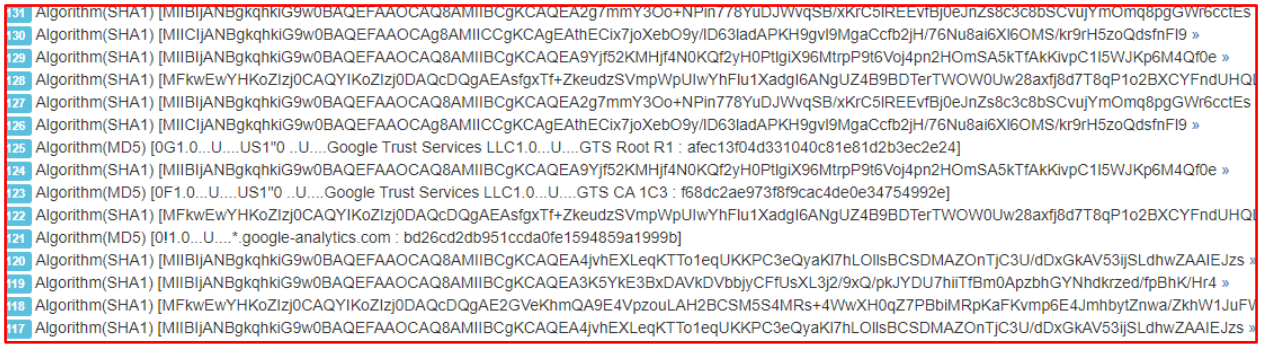

Figure 18: SHA1 & MD5 algorithm displayed during dynamic analysis

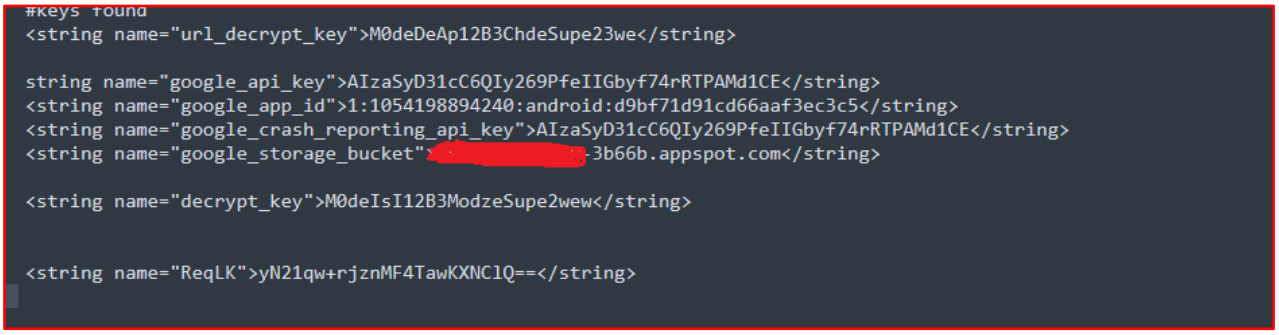

Figure 19: hardcoded secret key revealed during source code analysis

VI. DISCUSSION

It can be deduced from the results obtained during the mobile applications penetration testing that there were considerable number of high vulnerabilities across the applications. Vulnerabilities are categorized according to application permissions, code analysis and manifest analysis.

It can also be observed that m-bank-app1 recorded the least vulnerabilities of seven (7) with one (1) dangerous permission set for camera according to standard with CVSS of 7.5. It also recorded two (2) high vulnerabilities with CVSS of 7.4 during the manifest analysis and again one (1) medium severity level with CVSS of 5.9. The source code analysis also recorded two (2) high severity level with CVSS7.4 and 7.6 with one (1) medium severity level of 5.5. There were also quite best security practices of implementation of anti-root detection capabilities and also no exported activities, broadcast receivers and content providers were set to true.

m-bank-app2 also recorded considerably higher number vulnerabilities as compared to m-bank-app1.There were total number of twelve (12) vulnerabilities with three (3) application permissions set which according to standard are dangerous. They include android. permission. CALL_PHONE | directly call phone numbers, android.permission.CAMERA | take pictures and videos and android.permission.WRITE_EXTERNAL_STORAGE | read/modify/delete external storage contents. All permissions had a CVSS of 7.5. It was also observed that the network security was insecurely configured to allow cleartext traffic to domain in scope. It also recorded a CVSS of 6.0 with a medium severity level. During the manifest analysis there were three (3) high severity level identified with one (1) medium severity level. Of these there were two (2) activities set to true and one (1) broadcast receiver set to true. The highest CVSS recorded was 8.0 and least CVSS recorded was 5.5. The code

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

analysis also recorded three (3) high severity level with one (1) medium severity level with CVSS 5.9. It can also be concluded that m-bank-app2 had the third least number of vulnerabilities.

m-bank-app3 was the third application analyzed during the penetration testing. It recorded seven (7) total number of vulnerabilities with five (5) application permissions set whiles in the manifest analysis only two (2) vulnerabilities were identified with one (1) high and one (1) medium severity level recorded. It also recorded a CVSS of 7.5 and 5.5 for high and low respectively. There were no recorded exported activities, broadcast receiver and content provider vulnerabilities identified.

The last application analyzed was m-bank-app4 which recorded a tremendous number of high and medium vulnerabilities. It recorded the highest number of vulnerabilities a total of twenty-nine (29). There were seven (7) dangerous application permissions used for the mobile application. The highest CVSS recorded was 7.5. During the manifest analysis there were fourteen (14) identified vulnerabilities with only one (1) of medium severity level with a CVSS 0f 5.5 and remaining thirteen (13) of high severity level. M-bank-app4 had the highest number of twelve (12) exported activities with two (2) broadcast receiver during the attack surface analysis. The code analysis also contributed to the identification of any other vulnerabilites with a total number of nine (9). These vulnerabilities had five (5) issues with high severity level and four (4) medium severity level. Most security best practices were not identified during the penetration testing hence making it the worst secure application. Login portal was bypassed to access the backend of the application which ideally can be accessed through the required credentials.

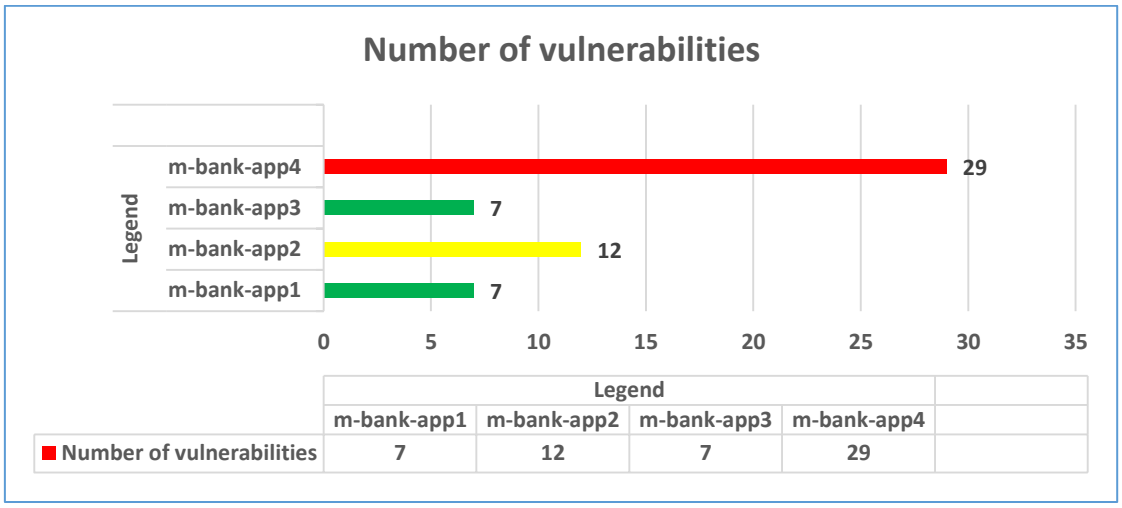

Figure 20: Applications with total number of vulnerabilities

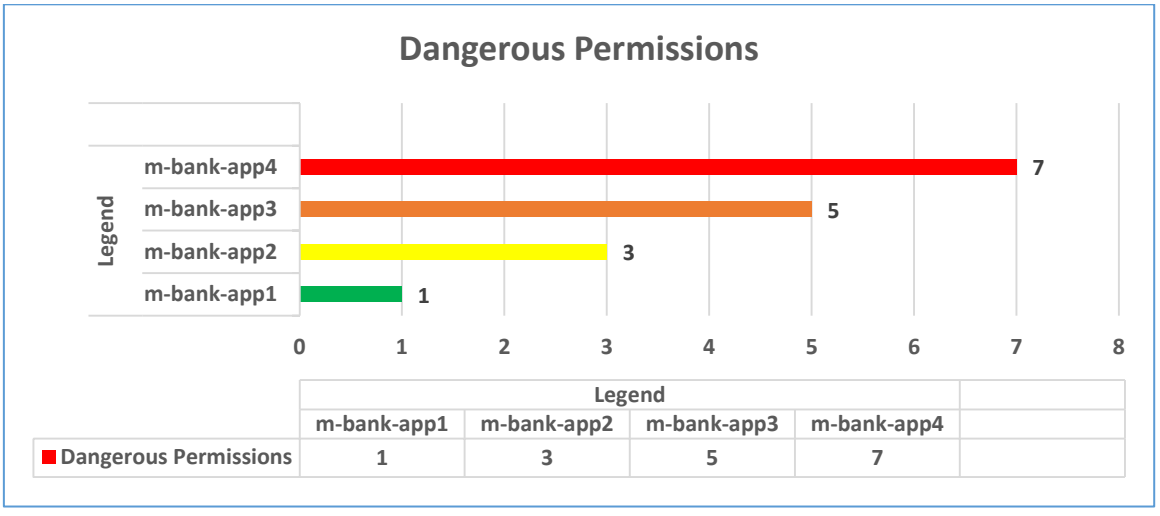

Figure 21: Applications with dangerous permissions

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:04/Issue:05/May-2022 Impact Factor- 6.752 www.irjmets.com

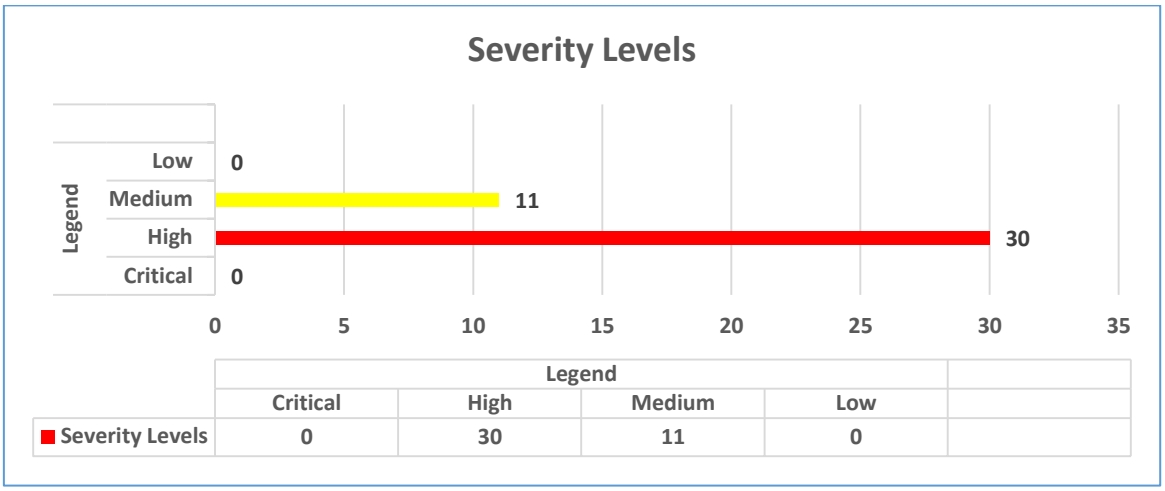

Figure 22: Applications severity levels

VII. CONCLUSION

Having performed comprehensive penetration testing on these high-level mobile applications for financial institutions, it can be concluded that security testing landscape has changed hence organizations should embrace hybrid approach as the de-facto way to ensure maximum and efficient applications before deployment.

Furthermore, organizations should note that automated security testing is a generic approach and critical vulnerabilities are likely to be missed hence leaving sensitive and critical applications vulnerable to attacks. Again, the kind of vulnerabilities identified in the applications depicted the security knowledge of the institution or third parties that developed respective mobile applications hence there is tremendous security knowledge gap in the mobile application industry.

VIII. FUTURE SCOPE

It is however recommended that penetration testing on mobile applications by institutions should be performed internally and supplement it with an advanced external penetration testing. Institutions should also encourage periodic security testing training and source code review for their developers and possibly build a simulated attack environment to test their application before deployment.

ACKNOWLEDGEMENTS

I am highly grateful to Dr. Manavjot Kaur, Director at Shaheed Udham Singh College of Engineering and Technology, (SUSCET), Tangori for providing this opportunity to carry out the present thesis/work.

The constant guidance and encouragement received from Er. Deepinder Kaur, Head, Department of CSE (SUSCET) has been of great help so carrying the present work and is acknowledged with reverential thanks.

I express gratitude to the faculty members of CSE Department, SUSCET, for their intellectual support throughout the course of this work

My acknowledgement also reaches out to my family for their financial support and prayers during my studies in India for their immeasurable love and understanding. I am highly favored and honor to have such a supportive person during my studies

IX. REFERENCES

[1] A. Gupta and E. Shapira, Learning pentesting for Android devices. Birmingham, UK: Packt Pub., 2014

[2] N. Elenkov, Android security internals. An In-Depth Guide to Android's Security Architecture (2015)

[3] "DIGITAL 2019: GLOBAL DIGITAL OVERVIEW". Global Digital Report, 31 January 2019, https://datareportal.com/reports/digital-2019-global-digital overview. Accessed 27 May 2019

[4] Number of Mobile Phone Users Worldwide from 2016 to 2023 (In Billions). Available online: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (accessed on 19 May 2021)

[5] Chetan C. Kotkar, Pravin Game Exploring Security Mechanisms to Android Device,2013

[6] Xinhong Hei, Wen Gao, Yichuan Wang, Lei Zhu, and Wenjiang Ji from Hardware to Operating System: A Static Measurement Method of Android System Based on TrustZone,2020
[7] Durmuş Özdemir, Hande Çavsizaim Investigation of Attack Types in Android Operating System,2021
[8] Ashwag Albakri, Huda Fatima, Maram Mohammed, Aisha Ahmed, Aisha Ali, Asala Ali and Nahla Mohammed Elzein Survey on Reverse-Engineering Tools for Android Mobile Devices,2022
[9] Cheolmin Yeom and YoojaeWon Vulnerability Evaluation Method through Correlation Analysis of Android Applications,2019
[10] Ratul Sikder, Md Shohel Khan, Md Shohrab Hossain, Wazir Zada Khan "A survey on android security: development and deployment hindrance and best practices", 2020
[11] Parag H. Rughani, H N Pandya Possibilities of Android Phone Based Cyber Crimes Due to Security Permissions – A Critical Study,2012
[12] Bharavi Mishra, Aastha Agarwal, Ayush Goel, Aman Ahmad Ansari, Pramod Gaur, Dilbag Singh, and Heung-No LEE Privacy Protection Framework for Android, 2022
[13] Francisco José Ramírez-López, Ángel Jesús Varela-Vaca, Jorge Ropero, Joaquín Luque, Alejandro Carrasco A Framework to Secure the Development and Auditing of SSL Pinning in Mobile Applications: The Case of Android, 2019
[14] Ikram U Haq and Tamim Ahmed Khan Penetration Frameworks and Development Issues in Secure Mobile Application Development: A Systematic Literature Review, 2021
[15] Janaka Senanayake, Harsha Kalutarage and Mhd Omar Al-Kadri Android Mobile Malware Detection Using Machine Learning: A Systematic Review,2021
[16] Rana Almohaini, Iman Almomani and Aala AlKhayer, Hybrid-Based Analysis Impact on Ransomware Detection for Android Systems, 2021
[17] José Palacios, Gabriel López, Franklin Sánchez Security Analysis Protocol for Android-Based Mobile Applications, 2019
[18] Amr Amin, Amgad Eldessouki, Menna Tullah Magdy, Nouran Abdeen, Hanan Hindy and Islam Hegazy AndroShield: Automated Android Applications Vulnerability Detection, a Hybrid Static and Dynamic Analysis Approach ,2019
[19] A. Lima, B. Sousa, T. Cruz, and P. Simoes, "Security for mobile device assets: a survey," in Proceedings of the ICMLG2017 5th International Conference on Management Leadership and Governance, p. 227, Academic Conferences and publishing limited, Braamfontein, Johannesburg, 2000, South Africa, Mar-2017.
[20] R. Zachariah, K. Akash, M. S. Yousef, and A. M. Chacko, "Android malware detection a survey," in Proceedings of the 2017 IEEE International Conference on Circuits and Systems. 238–244, ICCS, iruvananthapuram, Kerala, India, Dec-2017.
[21] B. F. Demissie, D. Ghio, M. Ceccato, and A. Avancini, "Identifying android inter app communication vulnerabilities using static and dynamic analysis," in Proceedings of the International Conference on Mobile Software Engineering and Systems, pp. 255–266, Austin, TX, May-2016.
[22] P. Bhat and K. Dutta, "A survey on various threats and current state of security in android platform," ACM Computing Surveys, vol. 52, no. 1, pp. 1–35, 2019.
[23] O. Ahmed and A. B. Sallow, "Android security: a review," Acad. Journal Nawroz University, vol. 6, no. 3, pp. 135–140, 2017.
[24] F. Guyton, "A survey of android security threats and machine learning techniques used for detection,", 2018.