

PAPER • OPEN ACCESS

Mobile Application Security Penetration Testing Based on OWASP

To cite this article: Aide Alanda *et al* 2020 *IOP Conf. Ser.: Mater. Sci. Eng.* **846** 012036

View the [article online](#) for updates and enhancements.

You may also like

- [Wireless smartphone-assisted personal healthcare monitoring system using a MoS₂-based flexible, wearable and ultra-low-cost functional sensor](#)

Akash Shinde, Parikshit Sahatiya, Anand Kadu et al.

- [Watch-Type Dual-Mode Wearable Health Device](#)

Husim Park, Liem H. T. Nguyen, Shanthala Lakshminarayana et al.

- [Graph Structure-based Clustering Algorithm for Android Third-party Libraries](#)
Shuyuan Liu and Gang Gan



HONOLULU, HI
October 6-11, 2024

Joint International Meeting of
The Electrochemical Society of Japan (ECSJ)
The Korean Electrochemical Society (KECS)
The Electrochemical Society (ECS)



Early Registration Deadline:
September 3, 2024

**MAKE YOUR PLANS
NOW!**



Mobile Application Security Penetration Testing Based on OWASP

Alde Alanda¹, Deni Satria², H.A Mooduto³, Bobby Kurniawan⁴,

¹²³Information Technology Department, Politeknik Negeri
Padang¹alde@pnp.ac.id, ²deni@pnp.ac.id, ³hamodutoo@gmail.com,
⁴bobbykurniawan@gmail.com

Abstract. Evolution of smartphones and smart devices affected one of the most used operating systems on smartphones and smart device is Android. Android growth with fast and affected the growth of applications used by that operating system. That application developed by many developers and can be downloadable on the play store. besides the benefits and features that operating systems are given and support from the application can affect security to users . The purpose of this research is to know vulnerability and technics that used to find a vulnerability in Android operating systems and Android applications. In addition, to give recommendations and prevention from the vulnerability. Technics and methods that used based on research from the OWASP Foundation consisting of 10 main vulnerability in Android application that is improper platform usage, insecure data storage, insecure communication, insecure authentication, insufficient cryptography, insecure authorization, client code quality, code tampering, reverse engineering, and extraneous functionality. The results from testing of five applications downloaded from Play Store. 4 application have vulnerability based on OWASP Mobile Top Ten documentation. The OWASP documentation can give an illustration of the vulnerability that most found in Android applications from the market.

1. Introduction

Android is an operating system that is widely used for mobile or smartphone. The development of Android can be said to be very fast so that a lot of updates from the previous operating system version. Android has applications that are used to do various things, such as word or data processing, image processing, sound processing, video processing, and various other application features.

The application was developed using Java and Kotlin programming languages. The Kotlin programming language is a new, more practical programming language. However, at this time many developers have developed a framework for the creation and development of other mobile applications, both Android and IOS. The framework was developed using various programming languages and various technologies. Examples of such frameworks are React Native, Flutter, and Kivy. The rapid development of the Android operating system resulted in this operating system being widely used for mobile or smartphone platforms. This development ultimately makes the application



that comes from the developer only concerned with the function without regard to the security of the application.

Tests conducted on mobile applications since 2012, found several general categories from the client side that cause weaknesses in mobile applications, these weaknesses include insecure data storage with a percentage of 63%, insecure transmission of data with a percentage of 57%, lack of binary protection with a percentage of 92%, client-side injection with a percentage of 40%, hard-coded password / keys with a percentage of 23%, and leakage of sensitive data with a percentage of 69%.[1]

2. Literature Review

A. OWASP

OWASP stands for Open Web Application Security Project is a non-profit organization located in the United States and was established on April 21, 2004. The OWASP Foundation itself was online on December 1, 2001, and has become an international organization and supports all OWASP activities in the world. OWASP is an open community that focuses on understanding, developing, obtaining, operating and maintaining applications that can be trusted. All OWASP tools, documents, and forums are free and open to anyone interested in improving and learning application security.[2]

One product of OWASP on mobile application security is documentation of the top 10 weaknesses that exist in mobile applications under the name OWASP Mobile Top Ten. OWASP Mobile Top Ten consists of the 10 most common weaknesses found during the development of mobile applications.

Table 1. OWASP Top 10 Mobile Application Vulnerability

No	Vulnerabilities
1	Improper Platform Usage
2	Insecure Data Storage
3	Insecure Communication
4	Insecure Authentication
5	Insufficient Cryptography
6	Insecure Authorization
7	Client Code Quality
8	Code Tampering
9	Reverse Engineering
10	Extraneous Functionality

B. Penetration Testing

Penetration testing is a very useful measurement tool for finding and finding weaknesses in network infrastructure, showing how vulnerable the network is when attacked . Penetration testing has proven effective in helping to deal with security issues on the network. Penetration testing techniques are not only aimed at applications, but can also be applied to networks, and operating systems, where the main purpose is to find and then try to exploit vulnerabilities that are known or detected in previous evaluations contained in certain technologies [3]. Mobile devices such as smartphones and tablets are widely used for personal and business purposes. A mobile device may carry sensitive data and becomes an easy target for cyber criminals.[4]

Penetration testing, also referred to as pentest or white hat hacking, is the process of a company hiring computer security professionals to try to break into their IT infrastructure with the intent to find where

the greatest vulnerabilities lie. Basically, a company hires security professionals to evaluate and hack into their network, servers, and services before the malicious users can do the same thing.[5] most penetration tests tend to focus on the exploitation of networks and the extraction of sensitive data, the loss of service and the inability of an organization to utilize its own wireless network can also have a significant impact on their productivity.[6]

3. Methodology

this research uses internal network penetration testing method. The steps that will be carried out in this research can be seen in the picture below

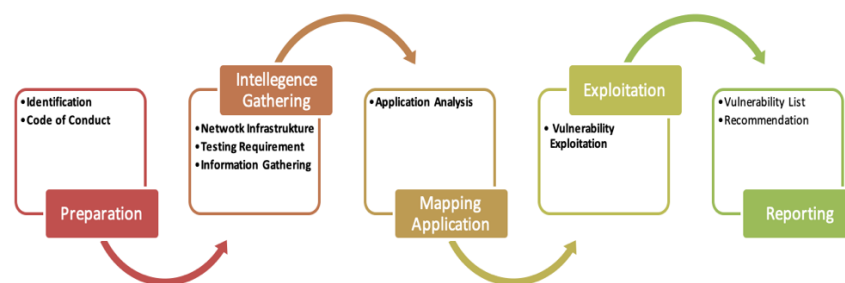


Figure 1. Methodology

1. Preparation, which is the stage that defines the scope of security testing, which includes the identification of security controls used, testing objectives, and sensitive data. In general, the preparation stage is all synchronization with the client in which there is an agreement between the examiner and the client to protect the examiner from legal actions[7]
2. Intelligence Gathering is a stage to analyze the scope and architecture of the application to get a general understanding of the application.
3. Mapping the Application, is the next step to complete the previous stage by scanning automatically and manually exploring applications. Mapping can provide a deeper understanding of the application to be tested such as entry points, data held, and other potential major weaknesses.
4. Exploitation is the stage where security testers penetrate an application by exploiting the weaknesses that have been identified during the previous process. At this stage also the determination of real weaknesses and true positives.
At this stage, using the attack reference found in OWASP top 10 to get the type of attack most often faced by mobile applications. At this stage several attack techniques will be carried out that are likely to be carried out on mobile applications.
The OWASP Security Knowledge Framework is intended to be a tool that is used as a guide for building and verifying secure software. It can also be used to train developers about application security. Education is the first step in the Secure Software Development Lifecycle.
5. Reporting, an important stage for the client, security testers provide weakness reports regarding the applications found and can explain the adverse effects of application weaknesses.

4. Result and Discussion

Application testing is based on documentation and methods contained in OWASP, which consists of 10 security issues in mobile applications that have been researched and are often encountered in the development of mobile applications.

Table 2. Vulnerability testing

Code	Vulnerabilities
M1	Improper Platform Usage
M2	Insecure Data Storage
M3	Insecure Communication
M4	Insecure Authentication
M5	Insufficient Cryptography
M6	Insecure Authorization
M7	Client Code Quality
M8	Code Tampering
M9	Reverse Engineering
M10	Extraneous Functionality

A. Improper Platform Usage

Testing conducted for this weakness is an abuse of one of the android features, Intent. The intent is used to move from one activity to another. Testing to be done is to do sniffing or tapping the application to get data from the intent that is processed when there is change inactivity. This can happen if the developer implements BroadcastReceiver which is used to communicate with other systems or applications, so that broadcast messages are sent through interprocess-communication and can be tapped, so developers shouldn't use BroadcastReceiver to send sensitive or confidential data.

```

W/dalvik ( 309): Long monitor contention event with owner Method-void com.
android.server.wm.WindowAnimator$.doFrame(long) from WindowAnimator.java:12
I waiters=0 for 127ms
I/ActivityManager( 569): Displayed com.android.insecurebankv2/.ChangePasswo
rd: +1s332ms
E/Surface ( 2266): getSlotFromBufferLocked: unknown buffer: 0xe8002700
D/NetlinkSocketObserver( 569): NeighborEvent(elapsedMs=1907232, 192.168.23.
1, [3085c204ff8], RTM_NEWNEIGH, NUD_STALE)
I/System.out( 2266): phono:15555218135
I/System.out( 2266): For the changepassword - phonenumber: 15555218135 passw
ord is: Updated Password from: NewPass@123 to: PasswordNew@321
D/baseband-sms( 272): newsms
D/baseband-sms( 272): sender:(N/A)
D/baseband-sms( 272): receiver:15555218135
D/baseband-sms( 272): index:1/1
D/baseband-sms( 272): txt:'Updated Password from: NewPass@123 to: PasswordN
ew@321'
D/MmsService( 937): getAutoPersisting
W/EGL_emulation( 2266): eglSurfaceAttrib not implemented
W/OpenGLRenderer( 2266): Failed to set EGL_SWAP_BEHAVIOR on surface 0xde42a1
60, error=EGL_SUCCESS

```

Figure 2. M1 Testing

The results of the test are in the form of data sent to the system log. The data sent can also be known by the application, so it can cause danger if the data contains sensitive information. The summary table of the test is listed in Table 3.

Table 3. Testing Result for M1

Code	M1
Vulnerability	Improper Platform Usage
Risk Level	High
Impact	Theft of sensitive information and control of application logic
Recommendation	Implement secure coding and pay attention to application development security advice from the official documentation.

B. M2-Insecure Data Storage

Testing of unsafe data storage is done in many ways, namely by testing the security of internal storage, external storage, content service providers, log files, XML data, binary data, cookies, SQL database (SQLite). The stored data can be used by attackers and malicious applications to obtain sensitive and confidential data that can result in misuse of data. One that will be discussed is regarding internal storage, which is shared preference. Developers often use MODE_WORLD_READABLE & MODE_WORLD_WRITEABLE for storing data in applications that can be accessed by other applications or attackers and also do not use encryption to store sensitive information.

```

root@vbox86p:/data/data/org.owasp.goatdroid.fourgoats # ls -al
drwxrwx--x u0_a64 u0_a64      2019-08-10 21:29 cache
drwxrwx--x u0_a64 u0_a64      2019-08-10 21:29 code_cache
drwxrwx--x u0_a64 u0_a64      2019-08-11 02:21 databases
drwxrwx--x u0_a64 u0_a64      2019-08-11 02:21 shared_prefs
root@vbox86p:/data/data/org.owasp.goatdroid.fourgoats # cd shared_prefs
root@vbox86p:/data/data/org.owasp.goatdroid.fourgoats/shared_prefs # ls -al
-rw-rw-r-- u0_a64 u0_a64      209 2019-08-11 02:21 credentials.xml
-rw-rw-r-- u0_a64 u0_a64      157 2019-08-11 02:13 destination_info.xml
-rw-rw-r-- u0_a64 u0_a64      148 2019-08-11 02:13 proxy_info.xml
root@vbox86p:/data/data/org.owasp.goatdroid.fourgoats/shared_prefs # cat credentials.xml
cat: credentials.xml: No such file or directory
root@vbox86p:/data/data/org.owasp.goatdroid.fourgoats/shared_prefs # cat credentials.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="username">goatdroid</string>
  <string name="password">goatdroid</string>
  <boolean name="remember" value="true" />
</map>
root@vbox86p:/data/data/org.owasp.goatdroid.fourgoats/shared_prefs #

```

Figure 3. M2 Testing

The results of the above test allow a user's login account to be stolen and used by other applications for certain purposes. As for the summary of the above test described in Table 4.

Table. 4 Testing Result for M2

Code	M2
Vulnerability	Insecure Data Storage
Risk Level	High
Impact	Theft of sensitive information, privacy violations, fraud, reputation damage, and violations of external policies.
Recommendation	1. Pay attention to the data stored. 2. Use strong encryption for data storage.

C. M3-Insecure Communication

Security testing is done by analyzing the use of communication in sending data between the backend and the client-side.

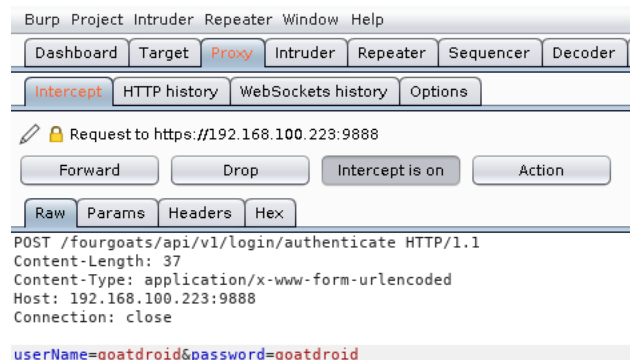


Figure 4. M3 Testing

The results of the above test are in the form of login account information that is sent in the form of the original text so that the information could be stolen and can harm the user. The summary of the tests above is described in table 5

Table 5. Testing Result M3

Code	M3
Vulnerability	Insecure Communication
Risk Level	High
Impact	Theft of sensitive information, violation of privacy, violation of user rights.
Recommendation	1. Use strong encryption on the data sent. 2. Use SSL / TLS in communication between server and client.

D. M4-Insecure Authentication

Insecure authentication is the weakness of the application regarding the user management of the application. Weaknesses that occur caused by users who are not entitled to access the application features without having to perform authentication. This can be done by bypassing the application and exploiting weaknesses in the application authentication process.

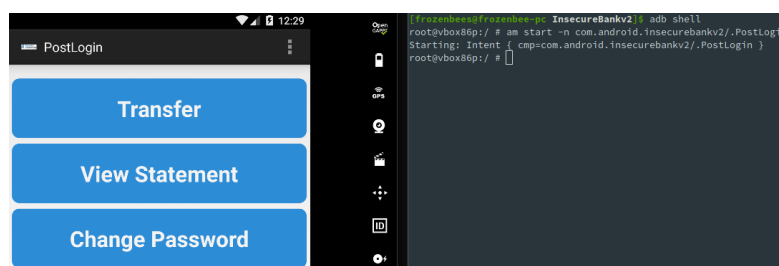


Figure 5. M4 Testing

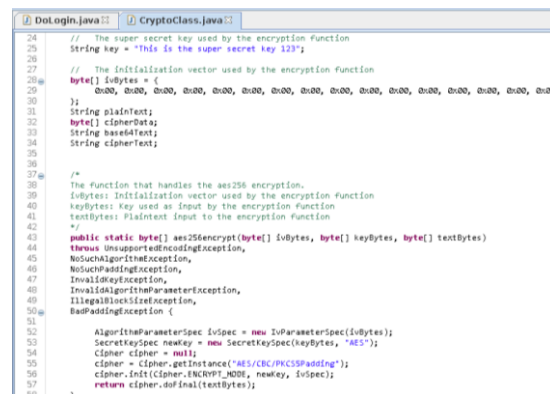
The results of testing obtained are bypass logins that can be performed so that users can enter the main page of the application without authentication. The summary of the tests is described in table 6

Table 6. Testing Result M4

Code	M4
Vulnerability	Insecure Authentication
Risk Level	High
Impact	Identity theft, violation of data access rights
Recommendation	<ol style="list-style-type: none"> 1. Make sure use of true on android: exported. 2. Make sure all authentication is done on the server side and not locally on the application.

E. M5-Insufficient Cryptography

The use of cryptography on data or information contained in the application is highly recommended so that the data can be safer and cannot be easily obtained by irresponsible users. Poor use of cryptography will not be able to protect the data or confidential information, because irresponsible users can easily obtain the data or information. Tests on M5 are performed to provide a description of poor cryptographic examples.



```

24 // The super secret key used by the encryption function
25 String key = "This is the super secret key 123";
26
27 // The initialization vector used by the encryption function
28 byte[] ivBytes = {
29     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
30 };
31 String plaintext;
32 byte[] cipherData;
33 String hexData;
34 String cipherText;
35
36
37 /*
38 The function that handles the aes256 encryption.
39 ivBytes: Initialization vector used by the encryption function
40 keyBytes: Key used as input by the encryption function
41 textBytes: Plaintext input to the encryption function
42 */
43 public static byte[] aes256encrypt(byte[] ivBytes, byte[] keyBytes, byte[] textBytes)
44 throws UnsupportedOperationException,
45     NoSuchAlgorithmException,
46     NoSuchPaddingException,
47     InvalidKeyException,
48     InvalidAlgorithmParameterException,
49     IllegalBlockSizeException,
50     BadPaddingException {
51
52     AlgorithmParameterSpec ivSpec = new IvParameterSpec(ivBytes);
53     SecretKeySpec newKey = new SecretKeySpec(keyBytes, "AES");
54     Cipher cipher = null;
55     cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
56     cipher.init(Cipher.ENCRYPT_MODE, newKey, ivSpec);
57     return cipher.doFinal(textBytes);
58 }

```

Figure 6. M5 Testing

The results obtained in testing the weaknesses of the use of cryptography are poor cryptographic implementation so it is easy to decrypt. The summary of the test is described in Table 7

Table 7 Testing Result M5

Code	M5
Vulnerability	Kriptografi yang tidak cukup
Risk Level	High
Impact	Identity theft, violation of data access rights
Recommendation	<ol style="list-style-type: none"> 1. Avoid storing sensitive data on internal storage. 2. Use cryptographic standards the past 10 years.

- | | |
|--|--|
| | 3. Use a strong cryptographic algorithm in accordance with NIST's recommendations. |
|--|--|

F. M6-Insecure Authorization

The issue of authorization has a close relationship with the problem of authentication and is sometimes combined. Managing the wrong authorization can result in unauthorized use of the feature and can cause problems for other users. Tests on M6 are carried out to give an idea of the application of bad authorization.

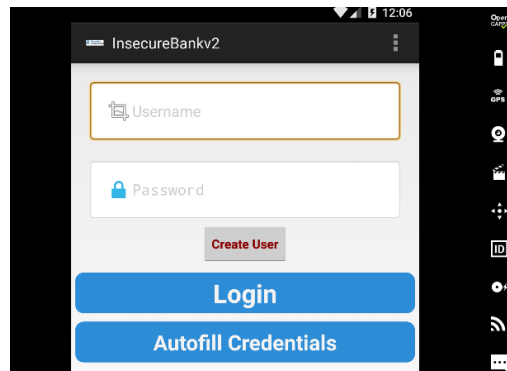


Figure 7. M6 Testing

A summary of the test results regarding authorization issues is explained in Table 8

Table 8. Testing Result for M6

Code	M6
Vulnerability	Insecure Authorization
Risk Level	High
Impact	Identity theft, violation of data access rights
Recommendation	<ol style="list-style-type: none"> 1. Verify that access rights can only be obtained from the backend system. 2. The backend system must be able to verify every request that comes. 3. Does not use storage or source code to set permissions.

G. M7-Poor Code Quality

Poor code quality can cause errors in the application so that the application does not run as it should even stop working. Applications that cannot filter input can cause irresponsible parties to run services without the need for authentication or authorization, change the application workflow and do what they want.

```

drozer Console (v2.3.4)
dz> run app.service.info --package org.owasp.goatdroid.fourgoats
Package: org.owasp.goatdroid.fourgoats
      org.owasp.goatdroid.fourgoats.services.LocationService
      Permission: null
dz>

```

Figure 8. M7 Testing

The results of testing regarding poor code quality are application services that can be run through applications or other sources aside from the application itself and do not need to authenticate even run applications. A summary of the test results regarding authorization issues is explained in Table 9

Table 9. Testing Result M7

Code	M7
Vulnerability	Poor Code Quality
Risk Level	Medium
Impact	Identity theft, violation of data access rights
Recommendation	<ol style="list-style-type: none"> 1. Menulis kode yang baik sesuai dokumentasi dan mudah dimengerti. 2. Gunakan penyaringan terhadap semua input pada aplikasi. 3. Memperhatikan penggunaan <i>permission</i> pada aplikasi.

H. M8-Code Tampering

Code tampering is a problem that occurs with an application due to changing and tampering with the application code by another party. This can cause changes to the logic and running of the application so that unauthorized parties can benefit from changing the application code.

**Figure 9.** M8 Testing

The result of testing regarding bad code tampering is that the process and logic in the application can be changed with the desire so that it can make the application no longer detect the root of the device. A summary of the test results regarding the problem of code tampering is explained in table 10

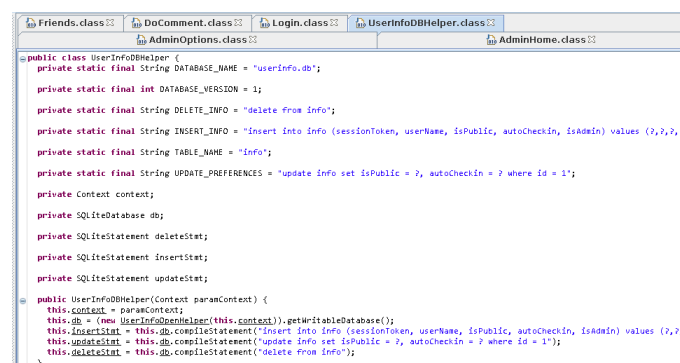
Table 10. Testing Result M8

Code	M8
Vulnerability	Code Tampering
Risk Level	Tinggi (Parah)
Impact	Loss of income by piracy, damage to reputation.

Recommendation	<ol style="list-style-type: none"> 1. The application must be able to check changes or additions to the code in the application. 2. Using anti-tampering.
----------------	---

I. M9-Reverse Engineering

Reverse engineering is a technique that is widely used to make changes to behavior and analysis of source code in the field of application development. Using reverse engineering techniques each party can get important information about the application starting from the application flow, application algorithms, cryptography used, information about backend server and other valuable information.



```

public class UserInfDBHelper {
    private static final String DATABASE_NAME = "userInfo.db";
    private static final int DATABASE_VERSION = 1;
    private static final String DELETE_INFO = "delete from info";
    private static final String INSERT_INFO = "insert into info (sessionToken, userName, isPublic, autoCheckin, isAdmin) values (?, ?, ?, ?, ?)";
    private static final String TABLE_NAME = "info";
    private static final String UPDATE_PREFERENCES = "update info set isPublic = ?, autoCheckin = ? where id = 1";

    private Context context;
    private SQLiteDatabase db;
    private SQLiteStatement deleteStat;
    private SQLiteStatement insertStat;
    private SQLiteStatement updateStat;

    public UserInfDBHelper(Context paramContext) {
        this.context = paramContext;
        this.db = (new UserInfDBHelper(this.context)).getWritableDatabase();
        this.insertStat = this.db.compileStatement("insert into info (sessionToken, userName, isPublic, autoCheckin, isAdmin) values (?, ?, ?, ?, ?)");
        this.updateStat = this.db.compileStatement("update info set isPublic = ?, autoCheckin = ? where id = 1");
        this.deleteStat = this.db.compileStatement("delete from info");
    }
  
```

Figure 10. M9 Testing

The results of tests regarding reverse engineering are important information from the application, namely the user database management process. A summary of the test results regarding authorization issues is explained in Table 11

Table 11. Testing Result M9

Code	M9
Vulnerability	Reverse Engineering
Risk Level	Medium
Impact	Identity theft, intellectual property theft, and destruction of the backend.
Recommendation	using obfuscation tool

J. M10-Extraneous Functionality

Extraneous Functionality that is usually found in applications is a backdoor. This backdoor planted by the developer on the application is developed, it could aim to facilitate the developers to improve the application when it was damaged, but can also be used to break into systems or applications for malicious purposes.

```

public void postData(String paramString) throws ClientProtocolException, IOException, JSONException, InvalidKeyException, NoSuchHostException {
    HttpResponse httpResponse = new DefaultHttpClient();
    HttpPost httpPost1 = new HttpPost(Dologin.this.protocol + Dologin.this.serverip + ":" + Dologin.this.serverport + "/login");
    HttpPost httpPost2 = new HttpPost(Dologin.this.protocol + Dologin.this.serverip + ":" + Dologin.this.serverport + "/devlogin");
    ArrayList<NameValuePair> arrayList = new ArrayList<>(2);
    arrayList.add(new BasicNameValuePair("username", Dologin.this.username));
    arrayList.add(new BasicNameValuePair("password", Dologin.this.password));
    if (Dologin.this.username.equals("devadmin")) {
        httpPost2.setEntity(new UrlEncodedFormEntity(arrayList));
        httpResponse = httpResponse.execute(httpPost2);
    } else {
        httpPost1.setEntity(new UrlEncodedFormEntity(arrayList));
        httpResponse = httpResponse.execute(httpPost1);
    }
}

```

Figure 11. M10 Testing

The test results regarding foreign functions are in the form of a backdoor in the application in the application login function so that the developer or those who know it can access the main page of the application by only using the username "devadmin" without requiring a password. A summary of the test results regarding authorization issues is explained in table 12

Table 12. Testing Result M10

Code	M10
Vulnerability	Extraneous Functionality
Risk Level	High
Impact	Unauthorized access to sensitive functions and theft of intellectual property.
Recommendation	<ol style="list-style-type: none"> 1. Check the application configuration to find hidden functions. 2. Did not enter a trial code in the production application. 3. Check the application log so that it does not display sensitive data from the backend.

From the 10 weaknesses discussed earlier in this study also tested several applications that can be downloaded at Playstore. The results of security testing on 5 applications in PlayStore can be seen in Table 13

Table 13. Penetration Testing Result

Applications	Vulnerability									
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
App A	-	√	-	-	-	-	-	-	-	-
App B	√	-	-	-	-	-	√	-	-	-
App C	-	-	-	-	-	-	-	-	-	-
App D	-	√	-	-	-	-	-	-	-	-
App E	-	-	-	√	-	-	-	-	-	-

5. Conclusion

In this research do some assessments for the weaknesses in mobile applications based on OWASP Mobile top 10. From the results of testing several android applications on Playstore applications have

vulnerabilities. 80% vulnerability the same as on OWASP Mobile to 10. In addition to testing, it also provides recommendations for various vulnerabilities of mobile applications so it can increase application security

References

- [1] D. Uttamchandani, *Handbook of Mems for Wireless and Mobile Applications*. 2013.
- [2] Owasp, *OWASP Top 10 - 2013*. 2013.
- [3] D. Satria, A. Alanda, A. Erianda, and D. Prayama, "Network Security Assessment Using Internal Network Penetration Testing Methodology," *JOIV Int. J. Informatics Vis.*, 2018.
- [4] Y. Wang and Y. Alshboul, "Mobile security testing approaches and challenges," in *2015 1st Conference on Mobile and Secure Services, MOBISECSESV 2015*, 2015.
- [5] L. Epling, B. Hinkel, and Y. Hu, "Penetration testing in a box," in *Proceedings of the 2015 Information Security Curriculum Development Conference on - InfoSec '15*, 2015, pp. 1–4.
- [6] J. R. R. Brian Sak, *Mastering Kali Linux Wireless Pentesting*. 2016.
- [7] R. Rahim, E. F. Armay, D. Susilo, R. F. Marta, and A. Alanda, "Cloud computing security issues and possibilities," *Int. J. Eng. Adv. Technol.*, 2019.