Securing Android Applications from Known Vulnerabilities through Penetration Testing

Putra Venkata Raju, K V S N Rama Rao, Gudapati Syam Prasad

Abstract*: Android assumes a prime role in the present market. As indicated by recent survey about 85% of individuals stick to android which dangerously turned out to be well known for individual or business purposes. It is no uncertainty that the applications are very recognizable in the market for their outstanding the brilliant advantages of android applications make the clients prefer it. Android confers critical duty to application engineers for structuring the application with understanding the danger of security issues. The greatest challenge faced by android users today is the security of applications they are using. The usage of smartphones for communication, social media, banking and payments has been increasing day-to-day and many people depend on it for their daily transactions. Android OS became most effective operating system today. This project performs penetration testing of android applications using simplified application program to perform penetration testing on android application. In this project, an attempt was made to test and analyze the security architecture of the android application using an android emulator.

***Index Terms*: Android security, vulnerabilities.**

I. INTRODUCTION

Android is based on Linux working framework it is planned basically for contact screen cell phones, for example, advanced mobile phones and tablet PCs. The working framework have developed a lot in latest 15 years start from profoundly differentiating phones to late advanced cells or littler than anticipated PCs. A champion among the most by and large used adaptable OS these days is android. The android is customizing that was built up in Palo Alto of California in 2003.The android is an incredible working framework and it bolsters substantial number of uses in Smartphones. These applications are dynamically pleasant and advanced for the customers the hardware that supports android programming relies upon arm configuration. The android is an open source working structure suggests that it's free and any one can use it. The android has got a huge number of applications accessible that can help you dealing with your life one or other way and it is accessible minimal effort in market at that reasons android is prevalent.

Numerous associations are moving to fit to the new market and workplace that utilizes cell phones. Explores have estimated that tablet PCs will invade around 85% of the overall portable work constrain before the finish of 2020, because of client experience of these gadgets.

There is have to guarantee that the information transmitted and put away in the tablets is verified. To accomplish this, a hostile security rather than the customary method of cautious by utilization of anti-virus will be conveyed. Security spills and private information divulgence from web and versatile applications are very regular today. With the expanding number of mechanically rich versatile applications hitting the market, cell phones have turned into the new focus for programmers. Cell phones are ending up increasingly progressed and utilized in further developed circumstances. These days they are utilized in the diversion division as well as in increasingly basic areas. The expanding multifaceted nature makes testing versatile applications hard, since the mix of conceivable sources of info develops quickly. Ensuring that your application is trustworthy before pushing an update isn't a simple undertaking, yet a vital one. By testing the application's usefulness, deficiencies can be found and the application will turn out to be progressively trustworthy. Notwithstanding, there are a wide range of testing techniques to look over. Along these lines we will examine how well extraordinary testing strategies really work. This will be finished by performing entrance tests on various applications that are broadly utilized in today.

Revised Manuscript Received on April 25, 2019.

Putra Venkata Raju, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, A.P., India.

Dr. K V S N Rama Rao, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, A.P., India.

Dr. Gudapati Syam Prasad, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, A.P., India.

II. RELATED WORK

The OWASP Top Ten is a rundown of the 10 most unsafe current mobile application security defects, alongside powerful techniques for managing those blemishes. OWASP (Open Web Application Security Project) is an organization that gives useful and financially effective data about mobile and Internet applications. Project individuals incorporate an assortment of security specialists from around the globe who share their insight into vulnerabilities, dangers, attacks and countermeasures.
According to OWASP these are the top 10 most exploitable mobile vulnerabilities

M1: Improper Platform Usage

Improper platform usage is any misuse of android platform features or lack of important security controls. It contains issues related to keychain or intents. Technical impact level is severe while business impact is noticeable.

M2: Insecure Data Storage

Insecure information storage vulnerabilities can complete a decent measure of harm to applications and the device itself, as it does in occurrences where vulnerability in one application opens up access to nearby records, exposing any sensitive information present there. That incorporates access to passwords, validation tokens, such as any geological, individual, and money related data in the applications.

M3: Insecure Communication

Mobile applications most of the time don't secure system traffic. They may utilize SSL/TLS while verification however not everywhere. This irregularity prompts the danger of uncovering information and session IDs to attack.
The utilization of transport security does not mean the application has executed it effectively. To identify essential defects, monitor the mobile system traffic.
When developing an application, information is usually traded in a client-server design. At the point when the application transmits its information, it must cross the cell phone's carrier and the web. Attackers may misuse vulnerabilities to block sensitive information while it's in transit. The accompanying risk operators exist:

- A foe that shares your nearby system (Shared Wi-Fi);
- Mobile Carrier or system devices (switches, cell towers, proxy's); or
- Malware on your cell phone.

M4: Insecure Authentication

Applications have to identify a user securely and maintain his identity, mostly when the user sends and receives sensitive data for example financial information.
Poor or missing authentication enables an attacker to execute usefulness inside the application or backend server utilized by the mobile application. More fragile validation for applications is genuinely common because of the mobile info structure factor.

The structure factor exceptionally supports short passwords that are regularly absolutely dependent on 4-digit PINs. A validation necessity for applications can be very extraordinary to conventional web confirmation conspires because of accessibility prerequisites. In applications, clients are not expected to be online consistently in their session. Device conditions are substantially less solid or unsurprising than conventional web connections.

M5: Insufficient Cryptography

On the off chance that your cryptographic conventions aren't acceptable, attackers can unscramble any basic information put away in the application or on the device. Dangers of broken cryptography incorporate security infringement, code theft and reverse engineering, and information theft. The mobile app may implement or leverage an encryption/decryption algorithm that is weak in nature and can be directly decrypted by the adversary.

M6: Insecure Authorization

To test for a poor authorization, analyzers can perform binary attacks against the application and attempt to execute advantaged commands that should just be executable with a client of higher benefit while the application is in 'disconnected' mode.

Analyzers should attempt to execute any special useful functionality a low-benefit session token inside the relating POST/GET demands for the weak functionality to the back-end server. Poor or missing authentication enables a hacker to execute useful commands they ought not be qualified for utilizing a verified but rather lower-benefit client of the versatile application.

M7: Client Code Quality

Risks or vulnerabilities involving buffer overflows, format-string vulnerabilities and various other code implementations. Mostly concentrates on third-party libraries that can be exploited by malicious users in order to hack applications. Code examination tools can help a security expert discover the utilization of interpreters and follow the information flow through the application. Manual pentests can confirm these issues by exploiting the weakness or vulnerabilities.

M8: Code Tampering

Code tampering discovery is a preventive measure utilized in applications to guarantee that an outsider hasn't recompiled and distributed or published your application under their record or store without your assent.
To recognize and alleviate the altering of Android applications.

- Check if the application has been renamed.
- Check if the application has been distributed without your assent.

M9: Reverse Engineering

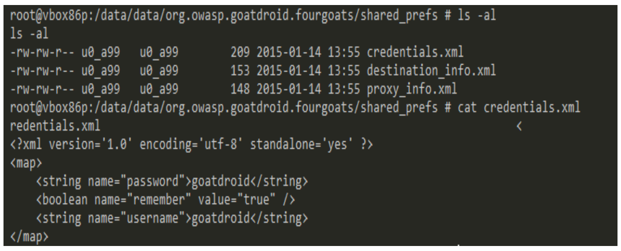
This classification incorporates investigation of the binary code to decide its source code, libraries, algorithms, and different resources. Programming, for example, IDA Pro, Hopper, otool, and other tools give the attackers knowledge into the inward activities of the application. This might be utilized to abuse other beginning vulnerabilities in the application, just as uncovering data about back end servers, cryptographic constants and ciphers, and protected IP rights.

M10: Extraneous Functionality

Most of the time developers incorporate hidden backdoor or other insider security controls that are not expected to be discharged into a production domain. For instance, an engineer may inadvertently incorporate a secret key as a comment in hybrid application. Another approach incorporates excluding of 2-factor verification testing.

III. PROPOSED FINDINGS

We propose usage of some free and open source tools for finding all mobile application vulnerabilities.

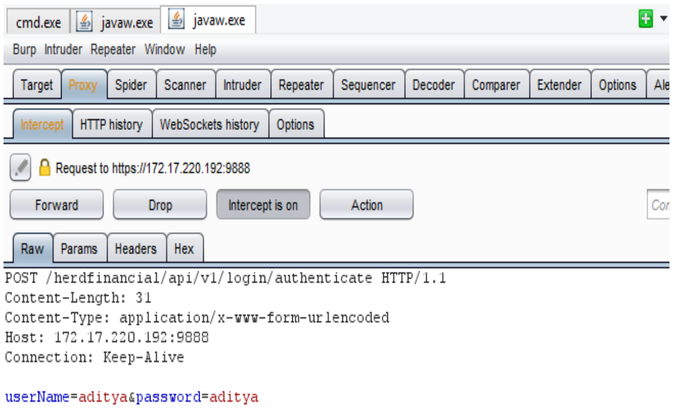


```
root@vbox86p:/data/data/org.owasp.goatdroid.fourgoats/shared_prefs # ls -al
ls -al
-rw-rw-r-- u0_a99   u0_a99       209 2015-01-14 13:55 credentials.xml
-rw-rw-r-- u0_a99   u0_a99       153 2015-01-14 13:55 destination_info.xml
-rw-rw-r-- u0_a99   u0_a99       148 2015-01-14 13:55 proxy_info.xml
root@vbox86p:/data/data/org.owasp.goatdroid.fourgoats/shared_prefs # cat credentials.xml
redentials.xml                                                                    <
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">goatdroid</string>
    <boolean name="remember" value="true" />
    <string name="username">goatdroid</string>
</map>
```

Fig 1 Finding insecure storage vulnerability

Insecure data storage vulnerability can be found in an application by inspect applications data through android debugging bridge shell which can access applications sensitive data with ease.
Preventive measures:

- Disallow sensitive information to be stored in log files.
- Encrypt sensitive files so that even if they are accessed, they're unreadable

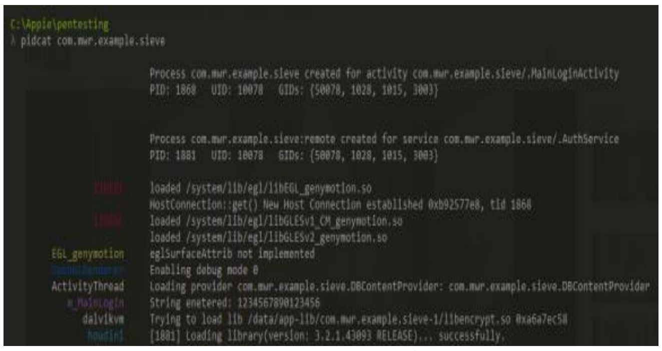


```
POST /herdfinancial/api/v1/login/authenticate HTTP/1.1
Content-Length: 31
Content-Type: application/x-www-form-urlencoded
Host: 172.17.220.192:9888
Connection: Keep-Alive

userName=aditya&password=aditya
```

Fig: 2 Insecure communication between client-server

Insecure communication between client and server can be easily identified by the burp suite tool which is used to sniff the traffic between two connections.
Preventive measures:

- Keep a constant eye on system traffic for occasions of conceivable attacks
- Ensure the application is appropriately configured to use SSL/TLS all through a client session or if nothing else on particularly delicate information, continually remembering the business estimation of the information gathered by an application
- Keep your SSL updated and it coordinates every one of the areas related with your portable application AND backend.

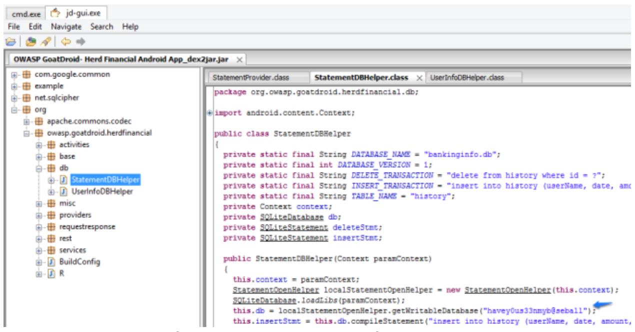


```
C:\Appie\pentesting
λ pidcat com.mwr.example.sieve

                  Process com.mwr.example.sieve created for activity com.mwr.example.sieve/.MainLoginActivity
                  PID: 1868   UID: 10078   GIDs: {50078, 1028, 1015, 3003}


                  Process com.mwr.example.sieve:remote created for service com.mwr.example.sieve/.AuthService
                  PID: 1881   UID: 10078   GIDs: {50078, 1028, 1015, 3003}

                  loaded /system/lib/egl/libEGL_genymotion.so
                  HostConnection::get() New Host Connection established 0xb92577e8, tid 1868
                  loaded /system/lib/egl/libGLESv1_CM_genymotion.so
                  loaded /system/lib/egl/libGLESv2_genymotion.so
   EGL_genymotion eglSurfaceAttrib not implemented
                  Enabling debug mode 0
   ActivityThread Loading provider com.mwr.example.sieve.DBContentProvider: com.mwr.example.sieve.DBContentProvider
      x_MainLogin String enetered: 1234567890123456
         dalvikvm Trying to load lib /data/app-lib/com.mwr.example.sieve-1/libencrypt.so 0xa6a7ec58
          houdini [1881] Loading library(version: 3.2.1.43093 RELEASE)... successfully.
```

Fig: 3 Sensitive data leakage from app logs

Applications data can be accidently leaked from many sources like copy/paste buffers and through crash logs. Sometimes data leakage can also be from data sent for third party analytics.
Preventive measures:

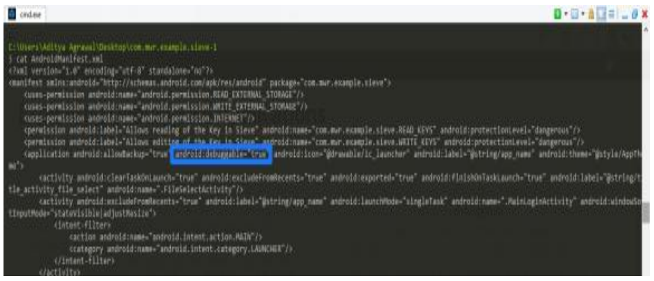
- Do not create logs when app crashes.



```
package org.owasp.goatdroid.herdfinancial.db;

import android.content.Context;

public class StatementDBHelper
{
  private static final String DATABASE_NAME = "bankinginfo.db";
  private static final int DATABASE_VERSION = 1;
  private static final String DELETE_TRANSACTION = "delete from history where id = ?";
  private static final String INSERT_TRANSACTION = "insert into history (userName, date, amo
  private static final String TABLE_NAME = "history";
  private Context context;
  private SQLiteDatabase db;
  private SQLiteStatement deleteStmt;
  private SQLiteStatement insertStmt;

  public StatementDBHelper(Context paramContext)
  {
    this.context = paramContext;
    StatementOpenHelper localStatementOpenHelper = new StatementOpenHelper(this.context);
    SQLiteDatabase.loadLibs(paramContext);
    this.db = localStatementOpenHelper.getWritableDatabase("havey0us33nmyb@seball");
    this.insertStmt = this.db.compileStatement("insert into history (userName, date, amount,
```

Fig 4: Information leakage from source code
An application may also contain sensitive information related to the database for example hardcoded keys or passwords.
Preventive measures:

- If your application doesn't require disconnected access, disable it
- Implement two-factor authentication whenever required the affectability of your application
- Take a favorable position of administrations like Microsoft's Access Control Service, which offers portable designers a simple method for approving and verifying clients outside of the application
- Always utilize either the devices local keychain



```
C:\Users\Aditya Agrawal\Desktop\com.mwr.example.sieve-1
$ cat AndroidManifest.xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.mwr.example.sieve">
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <permission android:label="Allows reading of the Key in Sieve" android:name="com.mwr.example.sieve.READ_KEYS" android:protectionLevel="dangerous"/>
    <permission android:label="Allows editing of the Key in Sieve" android:name="com.mwr.example.sieve.WRITE_KEYS" android:protectionLevel="dangerous"/>
    <application android:allowBackup="true" android:debuggable="true" android:icon="@drawable/ic_launcher" android:label="@string/app_name" android:theme="@style/AppThe
me">
        <activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:exported="true" android:finishOnTaskLaunch="true" android:label="@string/ti
tle_activity_file_select" android:name=".FileSelectActivity"/>
        <activity android:excludeFromRecents="true" android:label="@string/app_name" android:launchMode="singleTask" android:name=".MainLoginActivity" android:windowSo
ftInputMode="stateVisible|adjustResize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
```

Fig 5: Unused and dangerous app permissions
Applications have special rules called permissions, these permissions grant privileges to that applications for its tasks. These permissions can also be exploited in order to hack.
This include finding debuggable application from android permission file called AndroidManifest.xml.
Preventive measures:

- Set android:debuggable flag in AndroidManifest.xml file as false

The table below displays the threat and it definitions along with the effected property while analysing the applications

TABLE I. RESULT FINDINGS

Threat	Definition	Property
Insecure storage	Expose information without authorization	Confidentiality
Code Tampering	Modifying source code or data	Integrity
Insecure communication	Unencrypted communication between client and server	Integrity
Sensitive Data leakage	Exposure of sensitive data in logs	Confidentiality

*Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication*

IV. CONCLUSION

The mobile application infiltration testing procedure considers about versatile qualities and is vendor-neutral. It improves transparency and repeatability for mobile infiltration testing. It is an all-encompassing methodology with adequate adaptability and improves the security of applications. The strategy utilizes exhaustive knowledge, examination and exploitation, and guarantees clear introduction/announcing of the discoveries in a way that clearly imparts to both administration and the technical group.

REFERENCES

1. “Number of android apps 2019 – statistics,” [Online]. Accessible: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store.
2. “Smart device users around the world” – statistics,” [Online]. Accessible: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
3. Application Fundamentals Android Developers, “Application Fundamentals”,[Online].Available:www.stuff.mit.edu/afs/sipb/project/android/docs/guide/components/fundamentals.html. [Accessed: 05-Jan-2017].
4. “Android application security” 2019 https://source.android.com/security/overview/app-security.
5. J. Drake, Z. Lanier, C. Mulliner, O. Fora, A. Ridley and G. Wicherski “Android Security Design and Architecture,” in Android Hacker's handbook, 1st ed., John Wiley & Sons, Indianapolis: Wiley, pp. 27-28.
6. Kavitha. K, Salini. P, Ilamathy. V “Exploring the malicious android applications and reducing risk using static analysis,” in ICEEOT – 2016.
7. Yacouba KOURAOGO, Karim ZKIK, EI janati EI idrissi NOREDDINE, Ghizlane ORHANOU “Attacks on android banking applications,” in IEEE 2016.
8. Samaneh Hosseini Moghaddam, MaghsoodAbbaspour “Sensitivity Analysis of static features for android malware detection,” in ICEE 2014.
9. Shahid Iqbal, Amber Yasin, Talha Naqash “Android Security issues and solutions” in IEEE 2018.
10. BabuKhadiranaikar, PavolZavarsky, Yasir Malik “Improving android application security for intent based attacks” in IEEE 2017.
11. Xudong Wu, Xin Li “Hack android application and defence” in IEEE 2013
12. Yuan Zhang, Min Yang, Zhemin Yang, Guofei GU, Peng Ning, and Binyu Zang “Exploring Permission Induced Risk in Android–Applications for Malicious Detection” In IEEE transactions on information forensics and security, vol. 9, no. 11, November 2014.
13. Yuan Zhang, Min Yang, Zhemin Yang, and Binyu Zang.in “Permission Use Analysis for Vetting Undesirable Behaviors in Android Apps”In IEEE transactions on information forensics and security, vol. 9, no. 11, November 2014.
14. A. B. Ayed, “A literature Review on Android Permission System”, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4, Issue 4, April 2015.

AUTHORS PROFILE

Dr KVSN Rama Rao is working as a Professor in Koneru Lakshmaiah Education Foundation in Dept.of CSE. Had 20+ years of experience in academics and industry. International research experience at Australian university. His research interests include cyber security, machine learning and bioacoustics.

Dr. G. Syam Prasad is Currently working as Professor in CSE at KLEF(Deemed to be University), Vaddeshwaram, Guntur . Andhra Pradesh, India.He received the B.Tech and M.Tech degrees from the Department of computer Science and Engineering , Acharya Nagarjuna University, Guntur, India in 1999 and 2004 respectively, and Ph.D. degree from the Department of Computer Science and Systems Engineering , Andhra University at Visakhapatnam, India , in 2015. His research interests include network Security, cryptography, security and privacy, image processing, Data Mining, compilers and algorithms.