OWASP MSTG-Hacking-Playground

Location:
https://github.com/OWASP/MSTG-Hacking-Playground/tree/master/Android/MSTG-Android-Java-App

Get apk from here:
https://github.com/OWASP/MSTG-Hacking-Playground/releases/tag/1.0

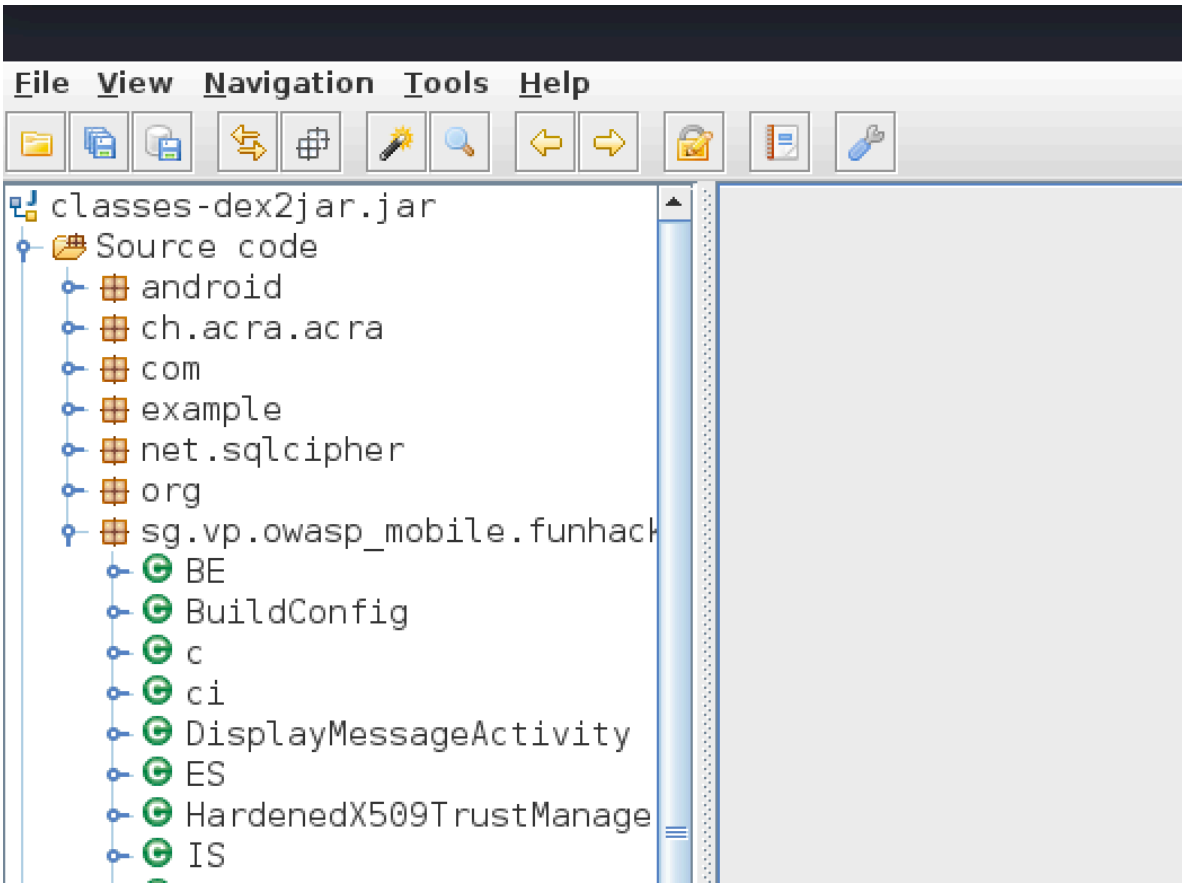
Guide (not a fill writeup):
https://github.com/OWASP/MSTG-Hacking-Playground/wiki/Android-App

Decompiling:

Way 1 : (view source code)

unzip assesment.apk --> uncompress the data
d2j-dex2jar classes.dex --> compile classes.dex to a readable format for jadx
jadx-gui < FULL path to classes-dex2jar.jar> --> fires jadx to view the source code

now we can view the source code



Way 2: (can view smali, AndroidManifest.xml , modify and recompile)

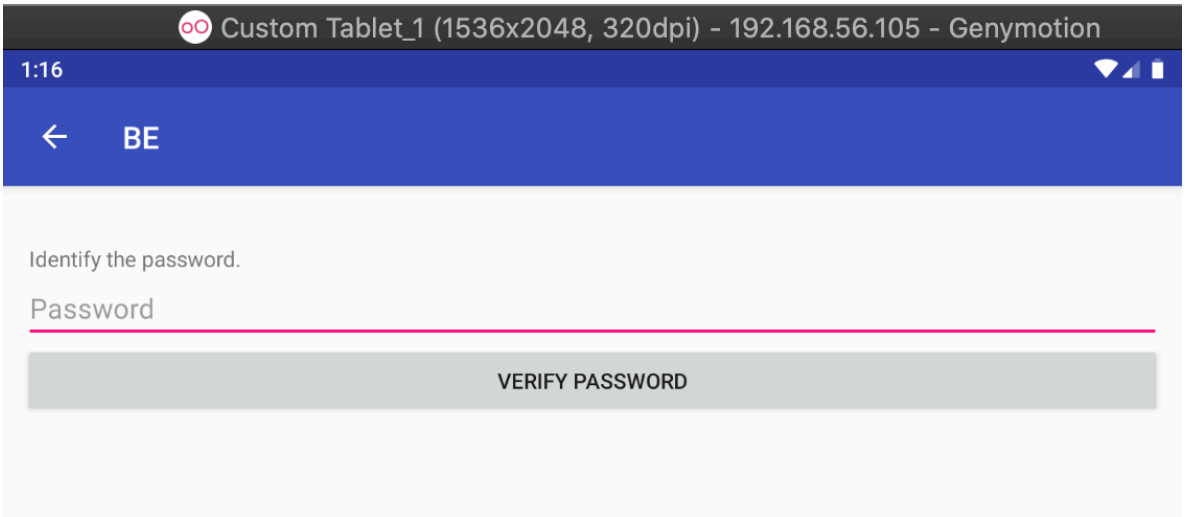
apktool d assesment.apk

Rebuild and Sign the apk with

apk uninstall
apktool b assesment -o assesment.s.apk
keytool -genkey -v -keystore test.keystore -alias Test -keyalg RSA -keysize 1024 -sigalg SHA1withRSA -validity 10000
jarsigner -keystore test.keystore assesment.s.apk -sigalg SHA1withRSA -digestalg SHA1 Test
jarsigner --verify --verbose assesment.s.apk
apktool install assessment.s.apk

APK info :
sg.vp.owasp_mobile.funhack

BE



we check the code to see how he processes the input and we find the following

sg.vp.owasp_mobile.funhack.BE

```java
        } else {
            Toast.makeText(this, "Try again!", 1).show();
        }
    }

    /* access modifiers changed from: private */
    public static boolean verify(String str) {
        byte[] decode = Base64.decode("vJqfip28ioydips=", 0);
        byte[] encrypt = encrypt(str);
        if (encrypt.length != decode.length) {
            return false;
        }
        for (int i = 0; i < encrypt.length; i++) {
            if (encrypt[i] != decode[i]) {
                return false;
            }
        }
        return true;
    }

    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_omtg__datast_001__bad_encryption);
        setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        this.passwordEditText = (EditText) findViewById(R.id.BadEnryptionPassword);
        this.btnVerify = (Button) findViewById(R.id.BadEnryptionButton);
        this.btnVerify.setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                BE.verify(BE.this.passwordEditText.getText().toString());
                BE.this.result(Boolean.valueOf(BE.verify(BE.this.passwordEditText.getText().toString())));
            }
        });
    }
```

This method is called once the button is pressed

this method verifies the password

shows the result

sg.vp.owasp_mobile.funhack.BE

```java
    /* access modifiers changed from: private */
    public void result(Boolean bool) {
        if (bool.booleanValue()) {
            Toast.makeText(this, "Congratulations, this is the correct password", 1).show();
        } else {
            Toast.makeText(this, "Try again!", 1).show();
        }
    }

    /* access modifiers changed from: private */
    public static boolean verify(String str) {
        byte[] decode = Base64.decode("vJqfip28ioydips=", 0);
        byte[] encrypt = encrypt(str);
        if (encrypt.length != decode.length) {
            return false;
        }
        for (int i = 0; i < encrypt.length; i++) {
            if (encrypt[i] != decode[i]) {
                return false;
            }
        }
        return true;
    }
```

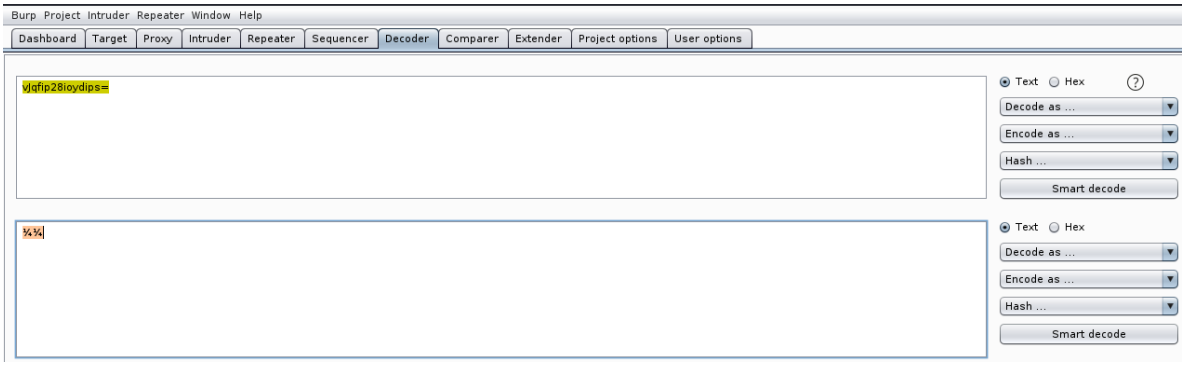
method that shows the result

we want to see this

this method encrypts the input and decodes this string and compares we need to make them equal

we now need to make a reverse operation and we get the supposed password

1- get the base 64 decode of the string :



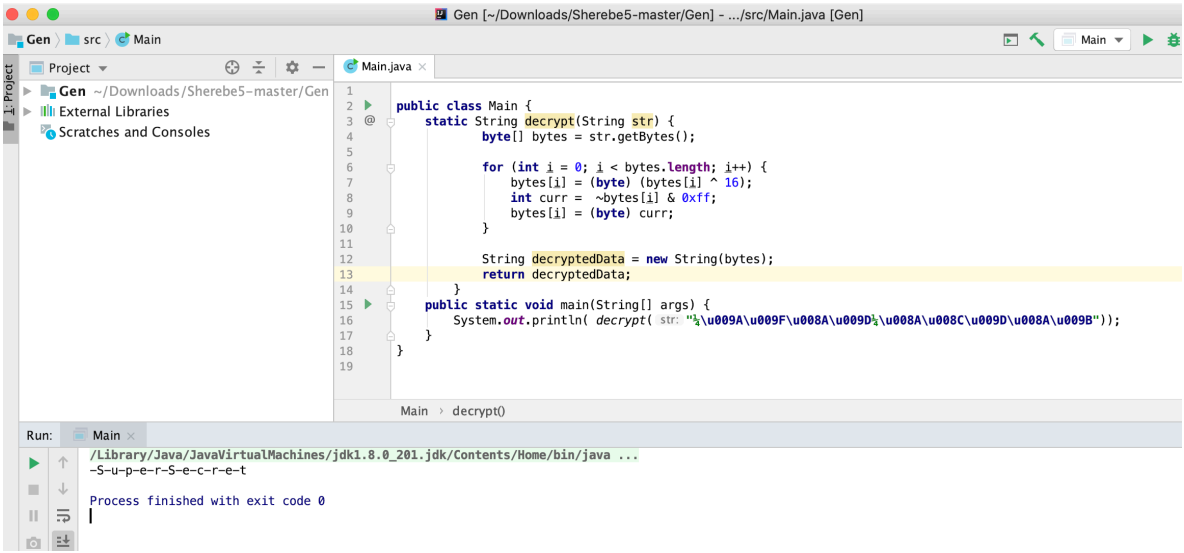
2- reverse the other two operations :

XOR :

reverse of xor is xor

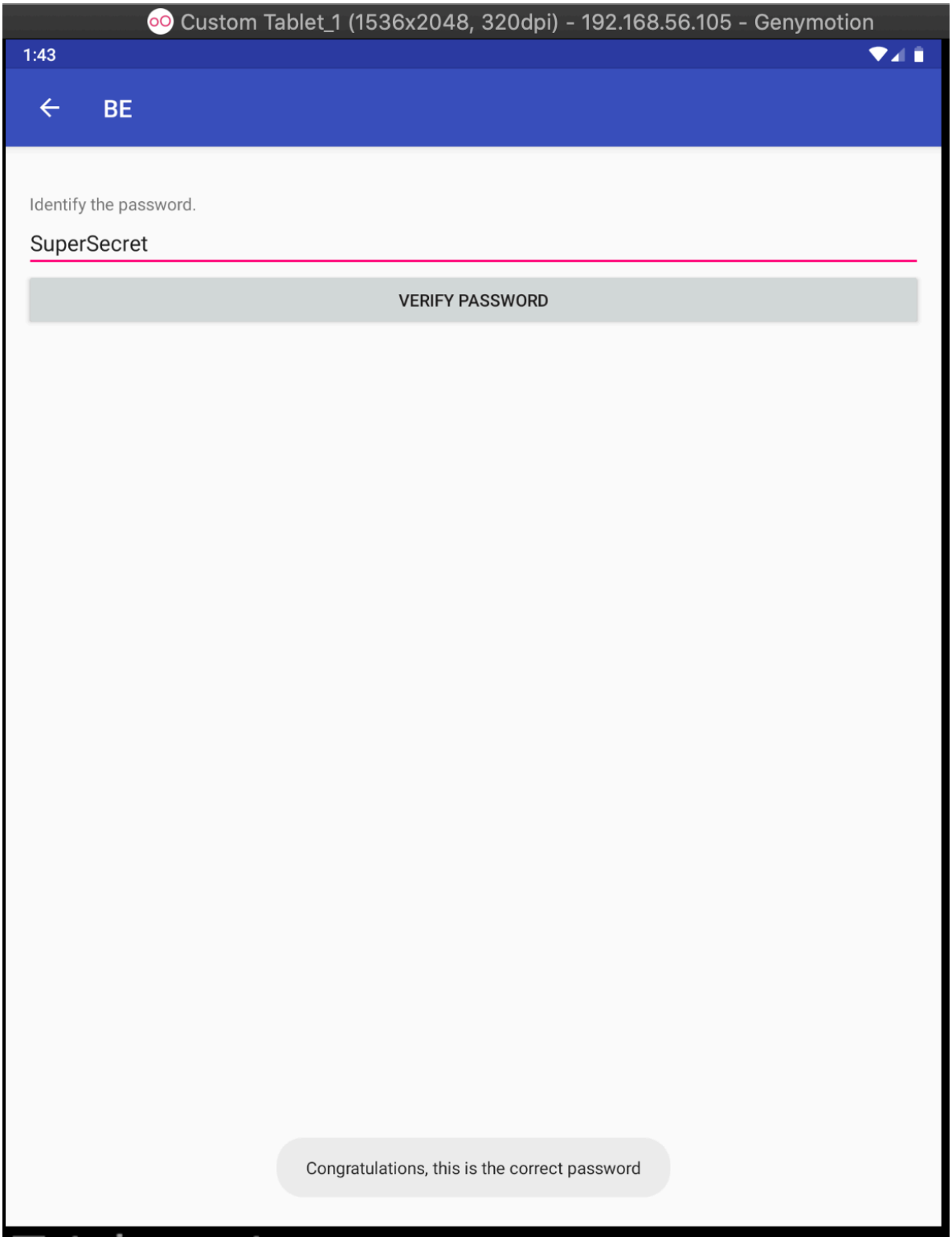
flipping the bits:

!x & 0xFF (if zero becomes one and if one becomes zero)



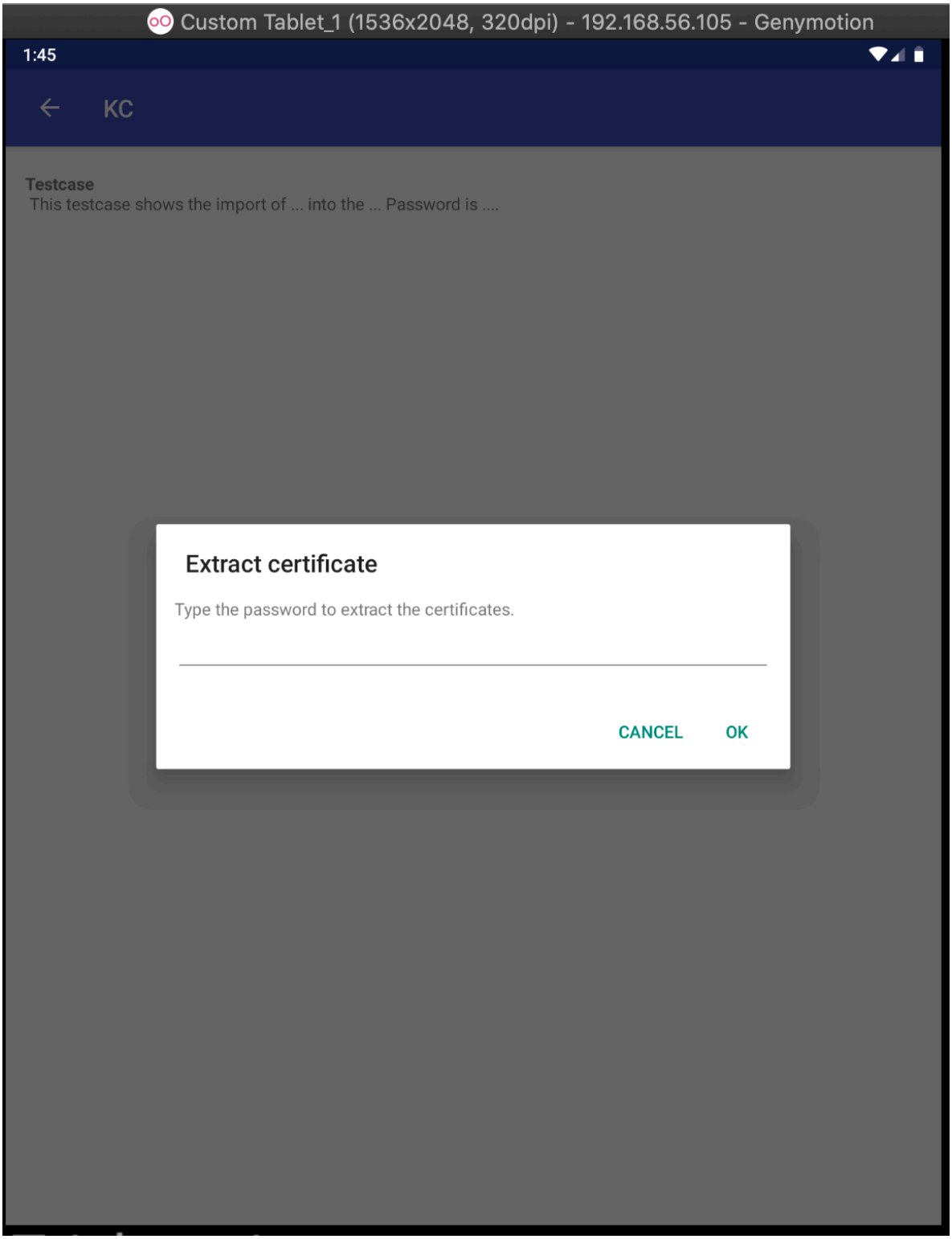
Since the decoded version is not a regular character we just use SuperSecret as it is suspected that the dashes are just unintened noise from the base64 decoding.

And we are now able to login



Recommendation :
You can use a strong hash to store the passwords locally instead of this weak encryption like sha512 where you encrypt the input and compare the hash.

KC



We check the code

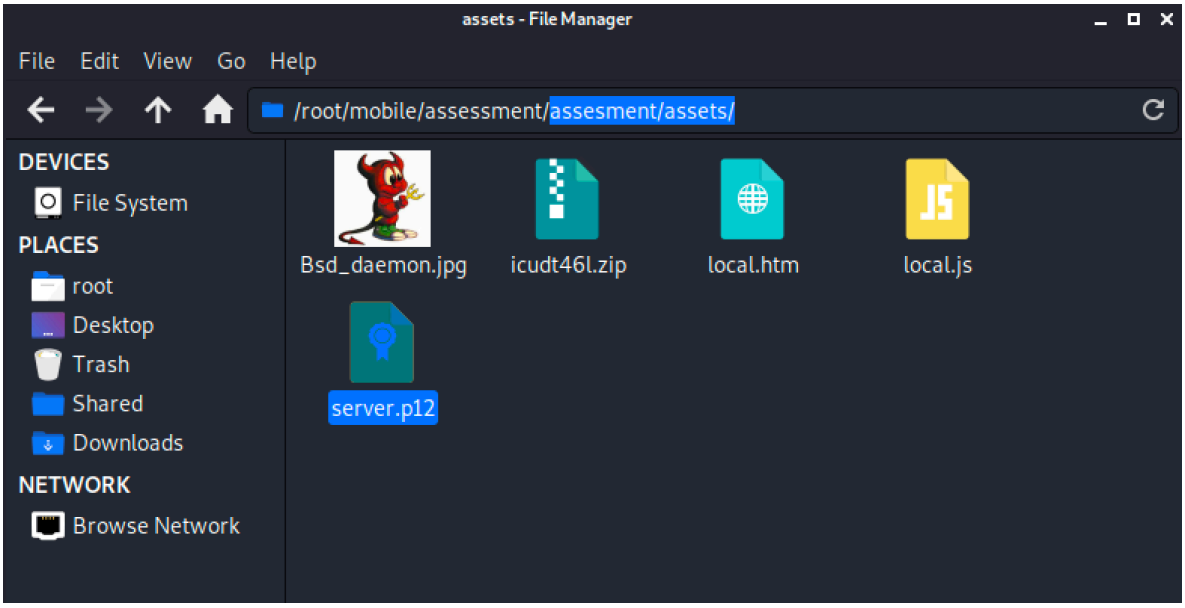
```
sg.vp.owasp_mobile.funhack.KC
8  import android.util.Log;
9  import java.io.BufferedInputStream;
10 import java.io.FileNotFoundException;
11 import java.io.IOException;
12
13 public class KC extends AppCompatActivity {
14     public static final String PKCS12_FILENAME = "server.p12";
15
16     private void installPkcs12() {
17         try {
18             for (String v : getAssets().list("")) {
19                 Log.v("names", v);
20             }
21             BufferedInputStream bufferedInputStream = new BufferedInputStream(getAssets().open(PKCS12_FILENAME))
22             byte[] bArr = new byte[bufferedInputStream.available()];
23             bufferedInputStream.read(bArr);
24             Intent createInstallIntent = KeyChain.createInstallIntent();
25             createInstallIntent.putExtra("PKCS12", bArr);
26             startActivity(createInstallIntent);
27         } catch (FileNotFoundException e) {
28             e.printStackTrace();
29         } catch (IOException e2) {
30             e2.printStackTrace();
31         }
32     }
33
34     /* access modifiers changed from: protected */
35     public void onCreate(Bundle bundle) {
36         super.onCreate(bundle);
37         setContentView((int) R.layout.activity_omtg__datast_001__key_chain);
38         setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
39         getSupportActionBar().setDisplayHomeAsUpEnabled(true);
40         installPkcs12();
41     }
42 }
```

maybe the certificate name

This method is called to once the activity KC is opened, It probably loads the ceritficate.

we find this certificate inside the decompressed folder of the apktool directory inside assest /server.12

you can find the password with a brute force which is 1234



This is probably not a vulnerability just shows the best practice to store the passwords.

KEYST

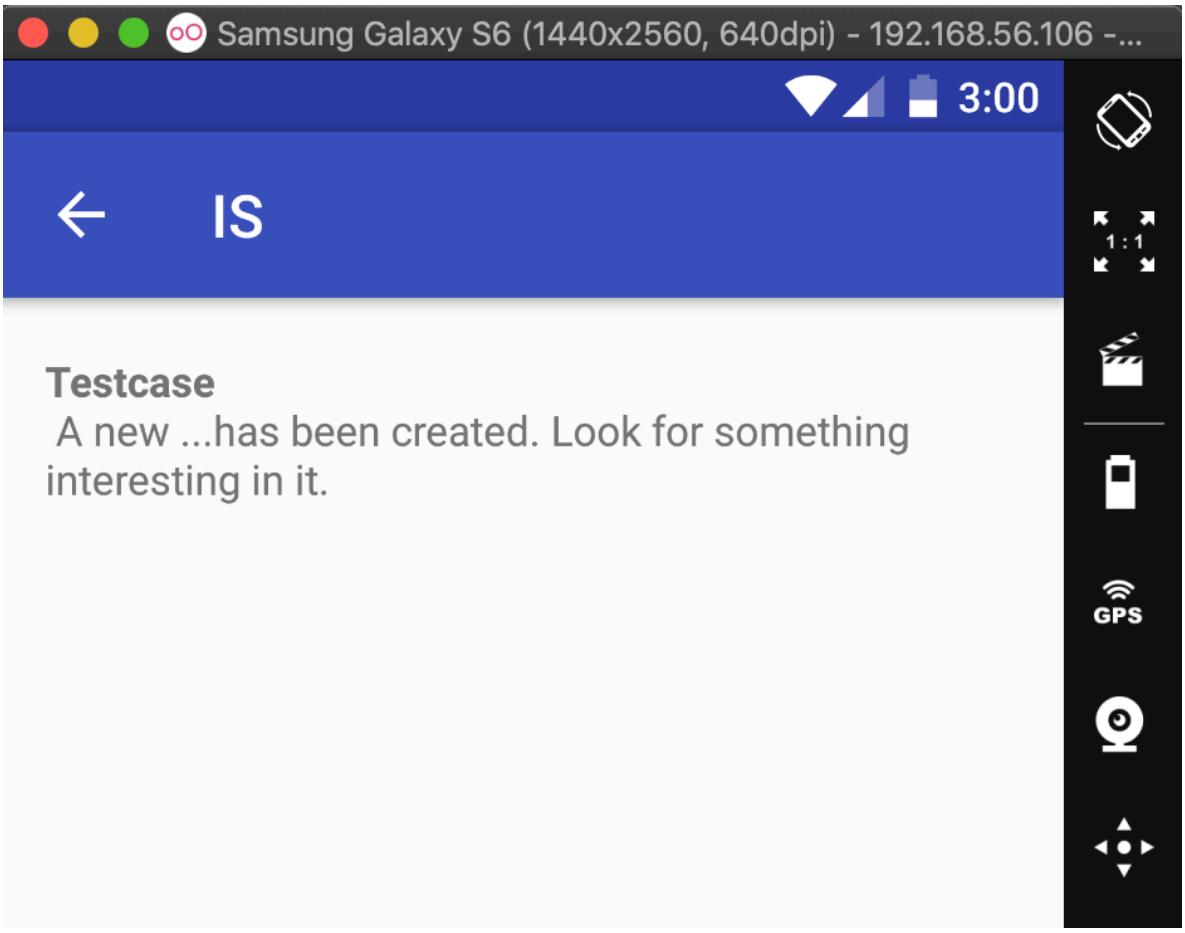
once we open the activity this method is called which creates a key pair.

```
sg.vp.owasp_mobile.funhack.Keyst
39      EditText startText;
40
41      public void createNewKeys() {
42          try {
43              if (!this.keyStore.containsAlias("Dummy")) {
44                  Calendar instance = Calendar.getInstance();
45                  Calendar instance2 = Calendar.getInstance();
46                  instance2.add(1, 1);
47                  KeyPairGeneratorSpec keyPairGeneratorSpec = null;
48                  if (VERSION.SDK_INT >= 18) {
49                      keyPairGeneratorSpec = new Builder(this).setAlias("Dummy").setSubject(new X500Principal("C
50                  }
51                  KeyPairGenerator instance3 = KeyPairGenerator.getInstance("RSA", "AndroidKeyStore");
52                  instance3.initialize(keyPairGeneratorSpec);
53                  instance3.generateKeyPair();
54                  Toast.makeText(getApplicationContext(), "Key Pair \"Dummy\" created.", 1).show();
55                  return;
56              }
57              Toast.makeText(getApplicationContext(), "Key Pair \"Dummy\" already created.", 1).show();
58          } catch (Exception e) {
59              StringBuilder sb = new StringBuilder();
60              sb.append("Exception ");
61              sb.append(e.getMessage());
62              sb.append(" occured");
63              Toast.makeText(this, sb.toString(), 1).show();
64              Log.e(this.TAG, Log.getStackTraceString(e));
65          }
66      }
67
```

this example just shows the best practice for storing the keys

IS

The developer seems to have created something once we opened this activity (view), we check the code to see what was it.



We check the code. and we find the following :

sg.vp.owasp_mobile.funhack.IS

```java
package sg.vp.owasp_mobile.funhack;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

public class IS extends AppCompatActivity {
    private void writeFile() throws IOException {
        FileOutputStream fileOutputStream = null;
        try {
            fileOutputStream = openFileOutput("test_file", 0);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        fileOutputStream.write("Credit Card Number is 1234 4321 5678 8765".getBytes());
        fileOutputStream.close();
    }

    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_omtg__datast_001__internal_storage);
        setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        try {
            writeFile();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Searching manually for the test_file we find it inside app/files

root@kali: ~/mobile/assessment/unzip | adb shel

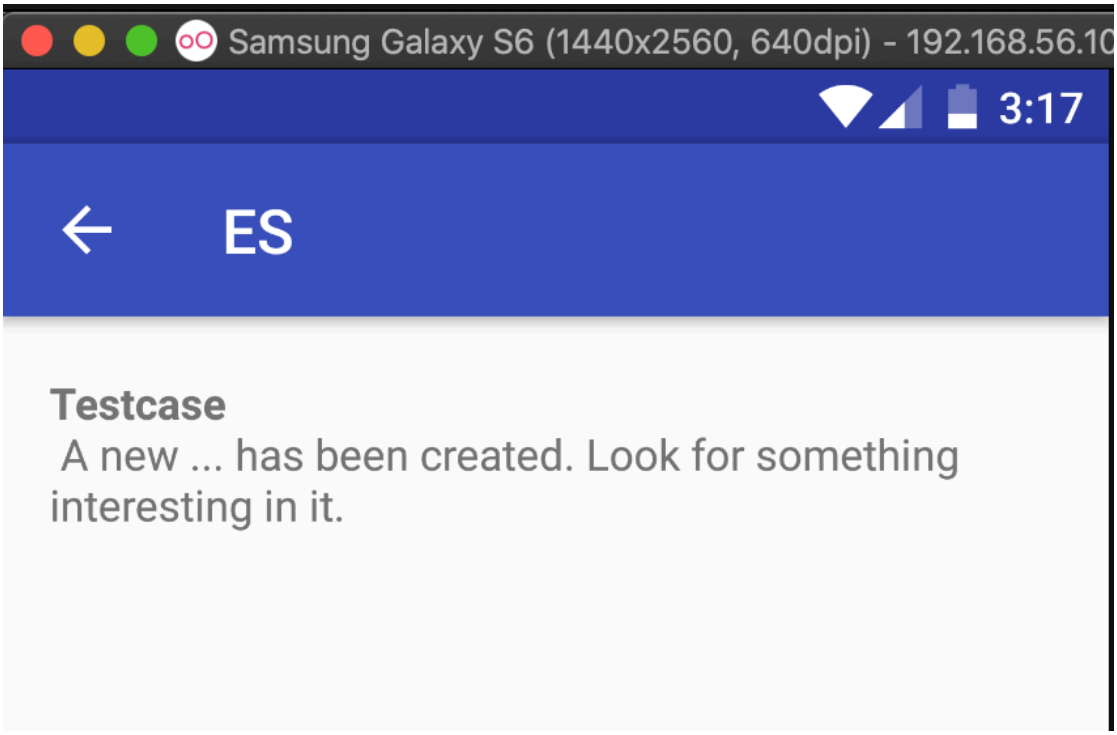
adb shell 105x28

```
root@vbox86p:/data/data/sg.vp.owasp_mobile.funhack/files # ls
test_file
root@vbox86p:/data/data/sg.vp.owasp_mobile.funhack/files # cat test_file
Credit Card Number is 1234 4321 5678 8765root@vbox86p:/data/data/sg.vp.owasp_mobi
```

This shows sensitive data exposure

ES

Same as last challenge :



We check the code:

sg.vp.owasp_mobile.funhack.ES

```java
import android.support.v7.widget.Toolbar;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

public class ES extends AppCompatActivity {
    public boolean isExternalStorageReadable() {
        String externalStorageState = Environment.getExternalStorageState();
        return "mounted".equals(externalStorageState) || "mounted_ro".equals(externalStorageState);
    }

    public boolean isExternalStorageWritable() {
        return "mounted".equals(Environment.getExternalStorageState());
    }

    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_omtg__datast_001__external_storage);
        setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        if (isExternalStorageWritable()) {
            try {
                FileOutputStream fileOutputStream = new FileOutputStream(new File(Environment.getExternalStorage
                fileOutputStream.write("L33tS3cr3t".getBytes());
                fileOutputStream.close();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e2) {
                e2.printStackTrace();
            }
        }
    }
}
```

Now the developer is writing inside the external .

```
root@vbox86p:/ # echo $EXTERNAL_STORAGE
/sdcard
root@vbox86p:/ # cd /sdcard
root@vbox86p:/sdcard # ls
Alarms
DCIM
Download
Movies
Music
Notifications
Pictures
Podcasts
Ringtones
root@vbox86p:/sdcard #
```

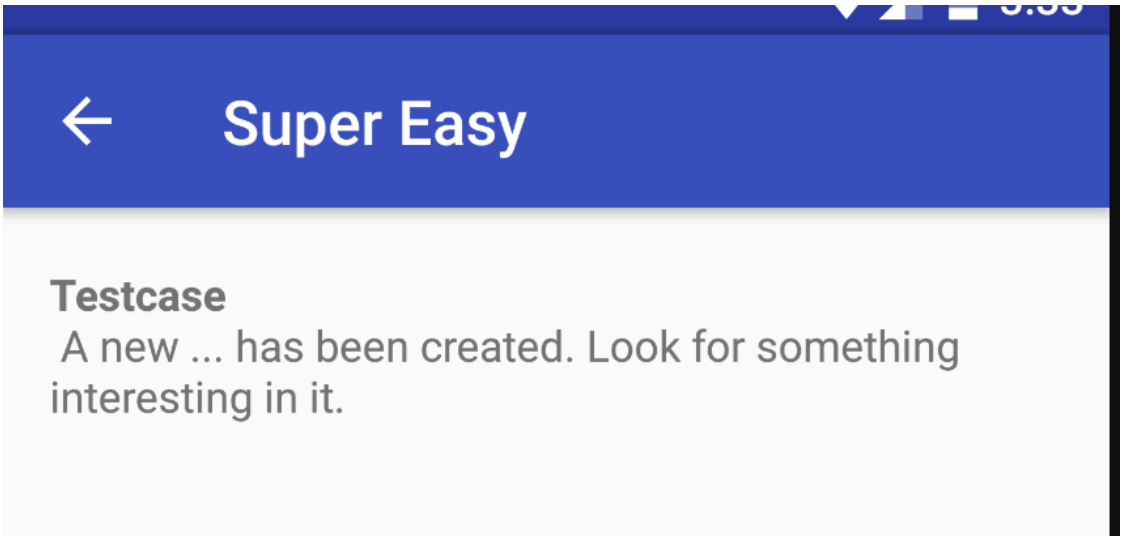
However due to unknown reasons (probably the android version) we couldnot find the file but if it was created it should be here)
We should find a file called password.txt

```
new FileOutputStream(new File(Environment.getExternalStorageDirectory(), "password.txt"))
```

Note :
echo $EXTERNAL_STORAGE --> this command shows the location of the external storage location which diifers from mobiles and versions to another.

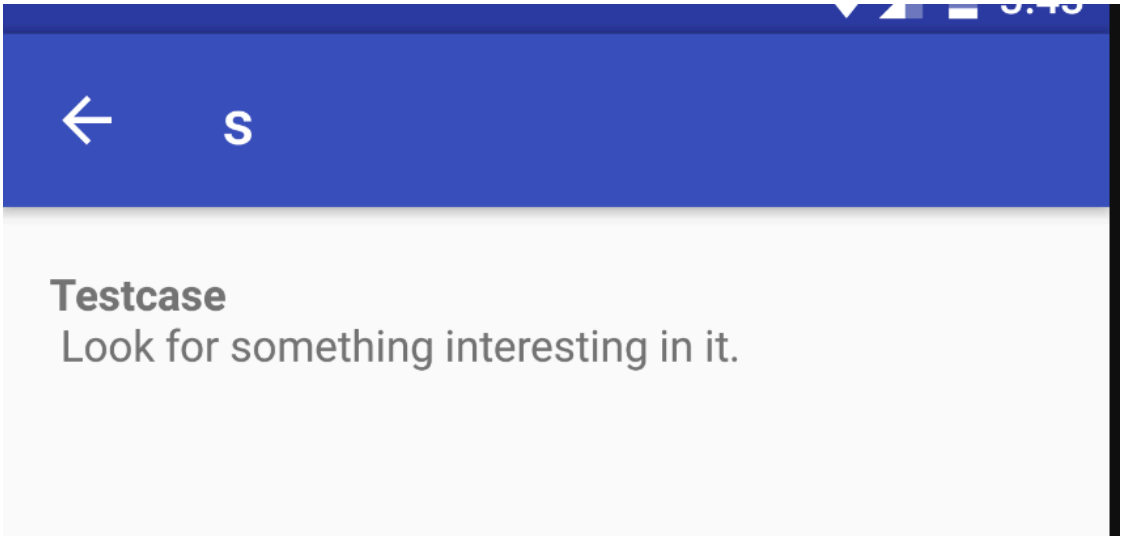
Recommendation:
Don't store any information that affects the application on the external storage as it is a shared location that all other applications can access and tamper with.

EASY



I could not find the code for this ...

S



Same as previous test case

SE

This one caused the application to crash whenever I run it , It might be doing something that is incompatible with my device
galaxy s6.

However from the code there is probably an SQL injection or Data is stored in clear text

```java
import net.sqlcipher.database.SQLiteDatabase.CursorFactory;

public class se extends AppCompatActivity {
    static {
        System.loadLibrary("native");
    }

    private void SQLiteEnc() {
        SQLiteDatabase.loadLibs(this);
        File databasePath = getDatabasePath("encrypted");
        databasePath.mkdirs();
        databasePath.delete();
        SQLiteDatabase openOrCreateDatabase = SQLiteDatabase.openOrCreateDatabase(databasePath, stringFromJNI(),
        openOrCreateDatabase.execSQL("CREATE TABLE IF NOT EXISTS Accounts(Username VARCHAR,Password VARCHAR);");
        openOrCreateDatabase.execSQL("INSERT INTO Accounts VALUES('admin','AdminPassEnc');");
        openOrCreateDatabase.close();
    }
```

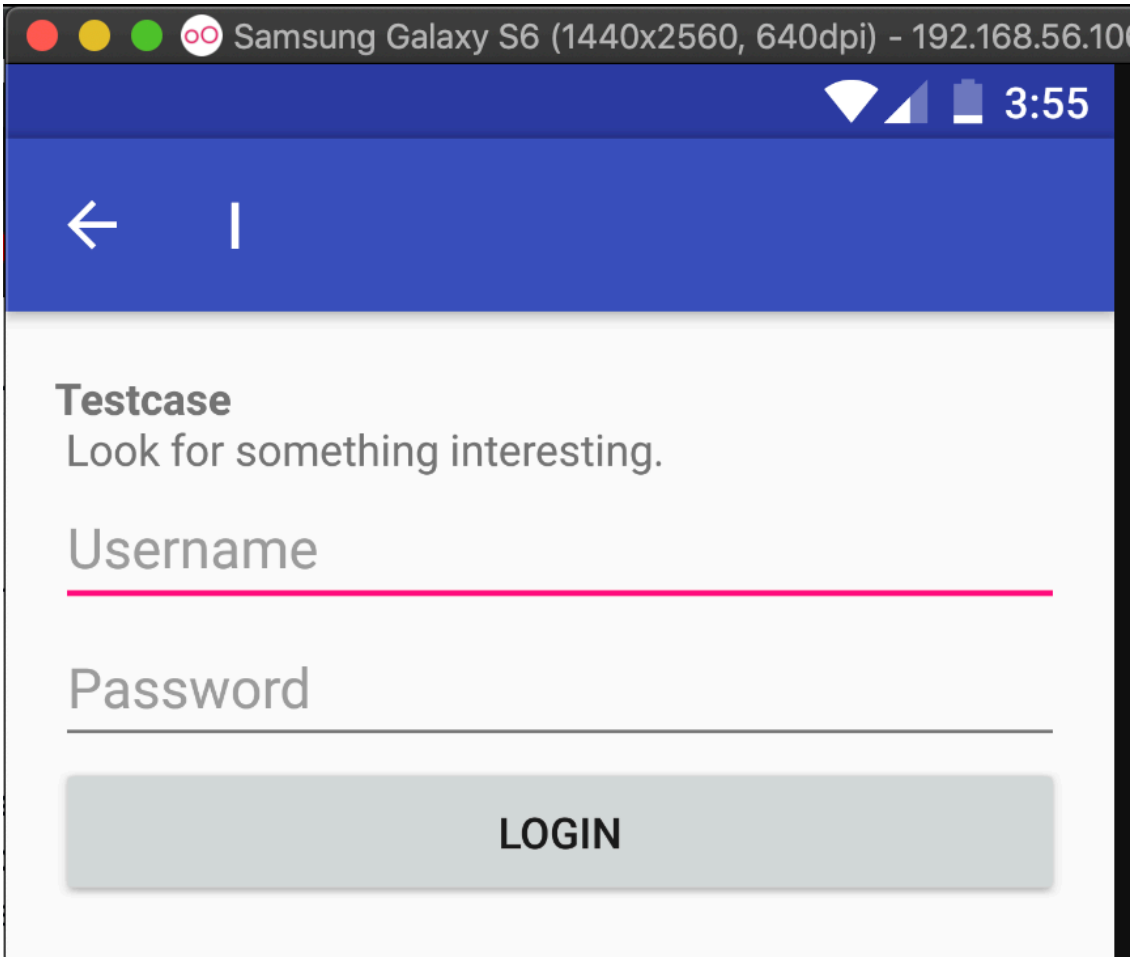
The second case turn out to be true

```
root@kali: ~/mobile/assessment/unzip                    sqlite3 privateNotSoSecure
sqlite3 privateNotSoSecure 126x30
 ⚡  ~/mobile/assessment  adb shell
root@vbox86p:/ # cd /data/data/sg.vp.owasp_mobile.funhack/databases
root@vbox86p:/data/data/sg.vp.owasp_mobile.funhack/databases # ls
College
College-journal
privateNotSoSecure
privateNotSoSecure-journal
root@vbox86p:/data/data/sg.vp.owasp_mobile.funhack/databases # exit
 ⚡  ~/mobile/assessment  p
 ✘ ⚡  ~/mobile/assessment  adb pull /data/data/sg.vp.owasp_mobile.funhack/databases/privateNotSoSecure

/data/data/sg.vp.owasp_mobile.funhack/databases/privateNotSoSecure: 1 file pulled. 6.9 MB/s (16384 bytes in 0.002s)
 ⚡  ~/mobile/assessment  ls
assesment  assesment.apk  privateNotSoSecure  unzip
 ⚡  ~/mobile/assessment  sqlite3 privateNotSoSecure
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> .tables
Accounts          android_metadata
sqlite> select * from Accounts
   ...> ;
admin|AdminPass
sqlite>
```

And we get the admin username and password

L

We find a login portal and we check the code we find some sort of logging that happens when we click the login button



sg.vp.owasp_mobile.funhack.l

```java
import android.widget.Toast;

public class l extends AppCompatActivity {
    String TAG = "l";
    EditText passwordText;
    EditText usernameText;

    /* access modifiers changed from: private */
    public void CreateLogs(String str, String str2) {
        String str3 = this.TAG;
        StringBuilder sb = new StringBuilder();
        sb.append("User successfully logged in. User: ");
        sb.append(str);
        sb.append(" Password: ");
        sb.append(str2);
        Log.e(str3, sb.toString());
        System.out.println("WTF, Logging Class should be used instead.");
        Toast.makeText(this, "Log output has been created", 1).show();
    }

    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_omtg__datast_002__logging);
        setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
        this.usernameText = (EditText) findViewById(R.id.loggingUsername);
        this.passwordText = (EditText) findViewById(R.id.loggingPassword);
        ((Button) findViewById(R.id.loginButton)).setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                l.this.CreateLogs(l.this.usernameText.getText().toString(), l.this.passwordText.getText().to
            }
        });
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }
}
```

Using the command # adb logcat
we find that the data is logged in clear text

```
04-23 15:56:29.010  2219  2219 E l       : User successfully logged in. User: ahmed Password: ahmedpassword
04-23 15:56:29.010  2219  2219 I System.out: WTF, Logging Class should be used instead.
04-23 15:56:29.055   275   667 D AudioFlinger: mixer(0xf1d40000) throttle end: throttle time(11)
04-23 15:56:29.061  2219  2233 V RenderScript: 0xe8e30000 Launching thread(s), CPUs 4
04-23 15:56:32.516  2219  2233 E Surface : getSlotFromBufferLocked: unknown buffer: 0xe9079620
```

Recommendation:
Remove any logs before the application is deployed specially sensitive information.
(Since logs only help when the developer is testing).

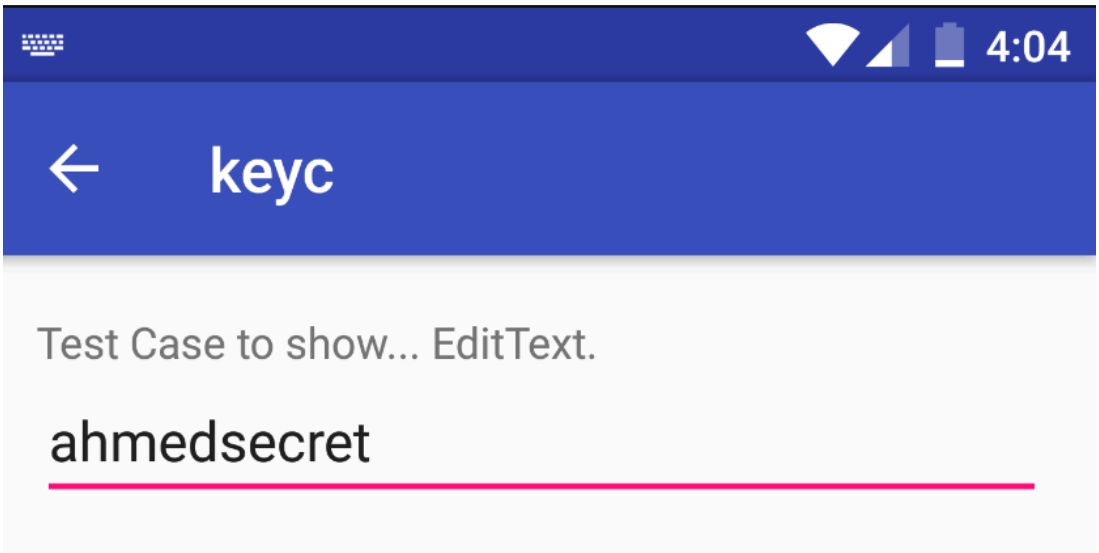
THIRDPRT

Don't Know how to do this.

sg.vp.owasp_mobile.funhack.l | sg.vp.owasp_mobile.funhack.thirdprt

```java
package sg.vp.owasp_mobile.funhack;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class thirdprt extends AppCompatActivity {
    /* access modifiers changed from: private */
    public void CrashApp() {
        throw new RuntimeException("This is a crash");
    }

    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_omtg__datast_004__3rd_party);
        setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        ((Button) findViewById(R.id.crashButton)).setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                thirdprt.this.CrashApp();
            }
```

KeyC



In order to exploit this we need to enable the virtual keyboard in the emulator (genymotion)
We get from the code that this is related to the keyboard cache

What we should be able to do is

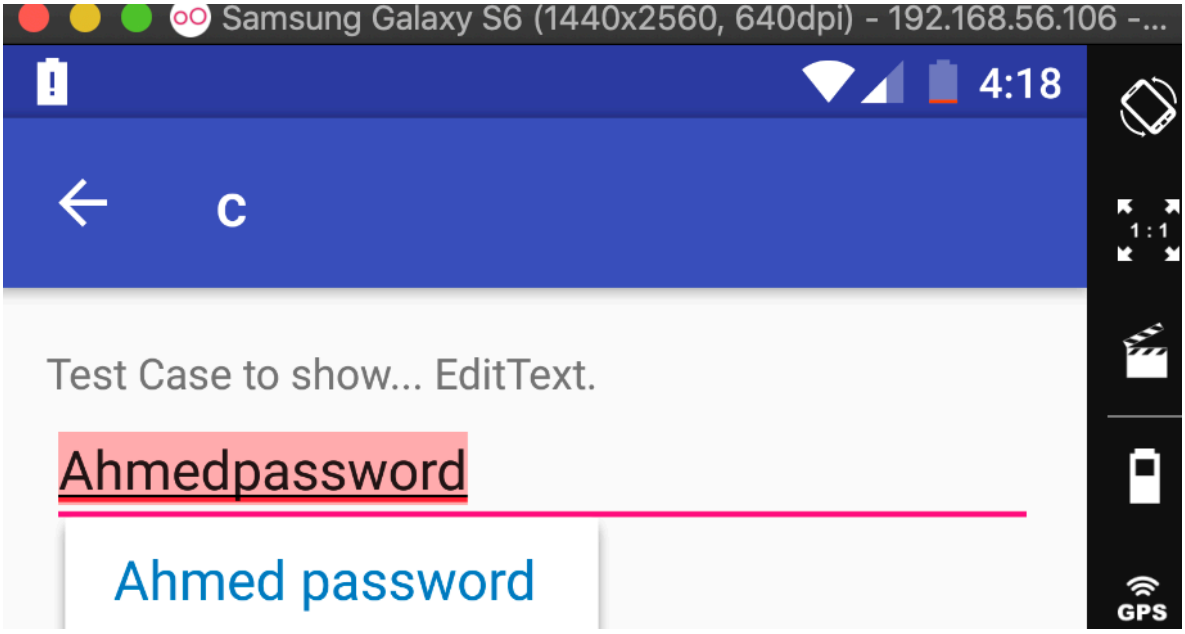
1. Enter any sensitive information
2. Select this info field and select the “Add to dictionary” option.

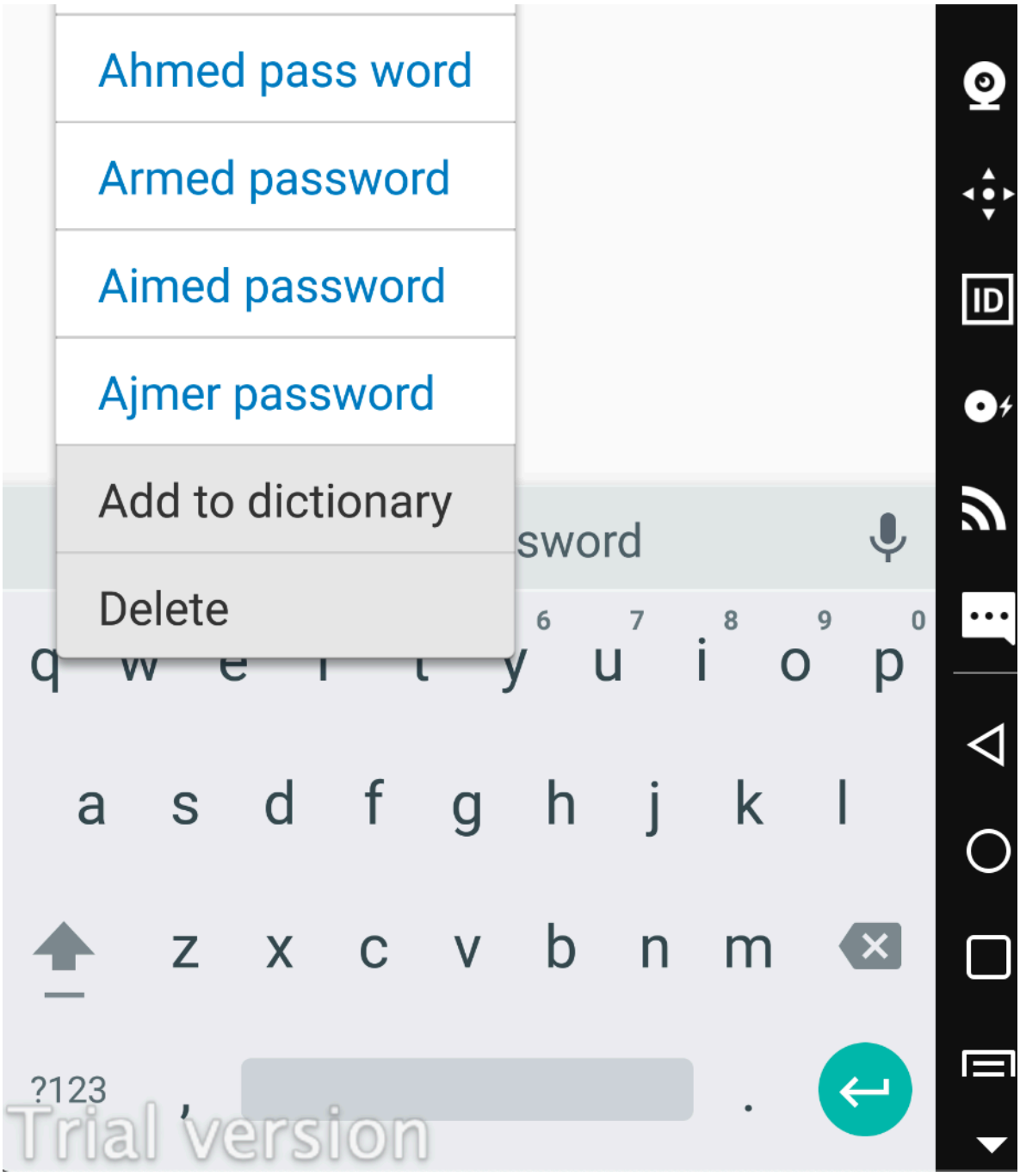
```
# adb pull /data/data/com.android.providers.userdictionary/databases/user_dict.db
# sqlite3 user_dict.db
# select * from words;
```

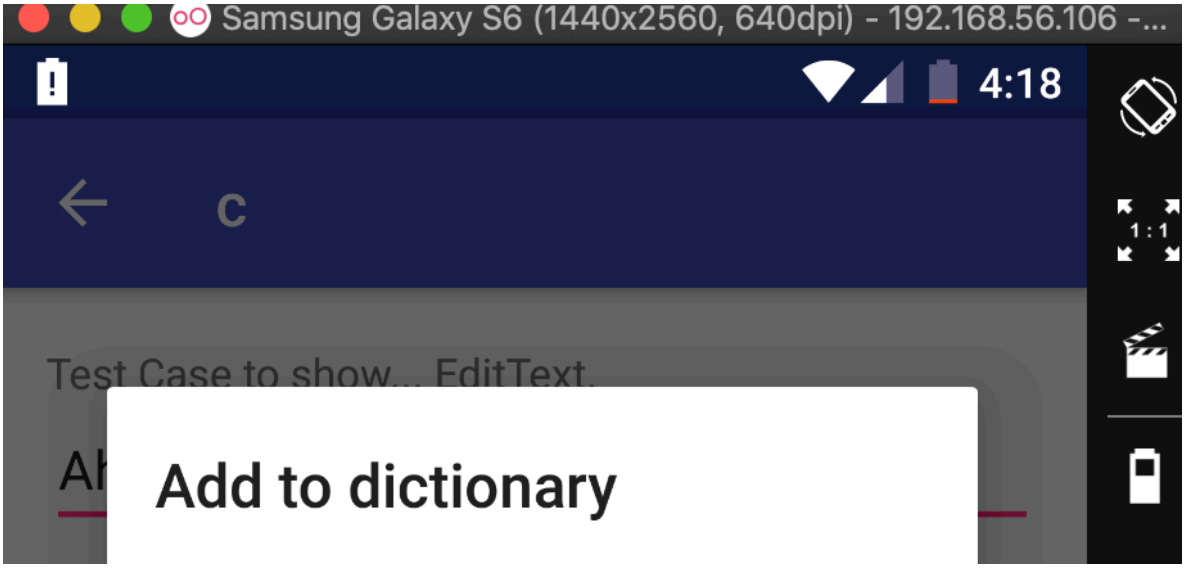
However it seems that it was mitigated here as we couldnot add it to the dictionary :') :')

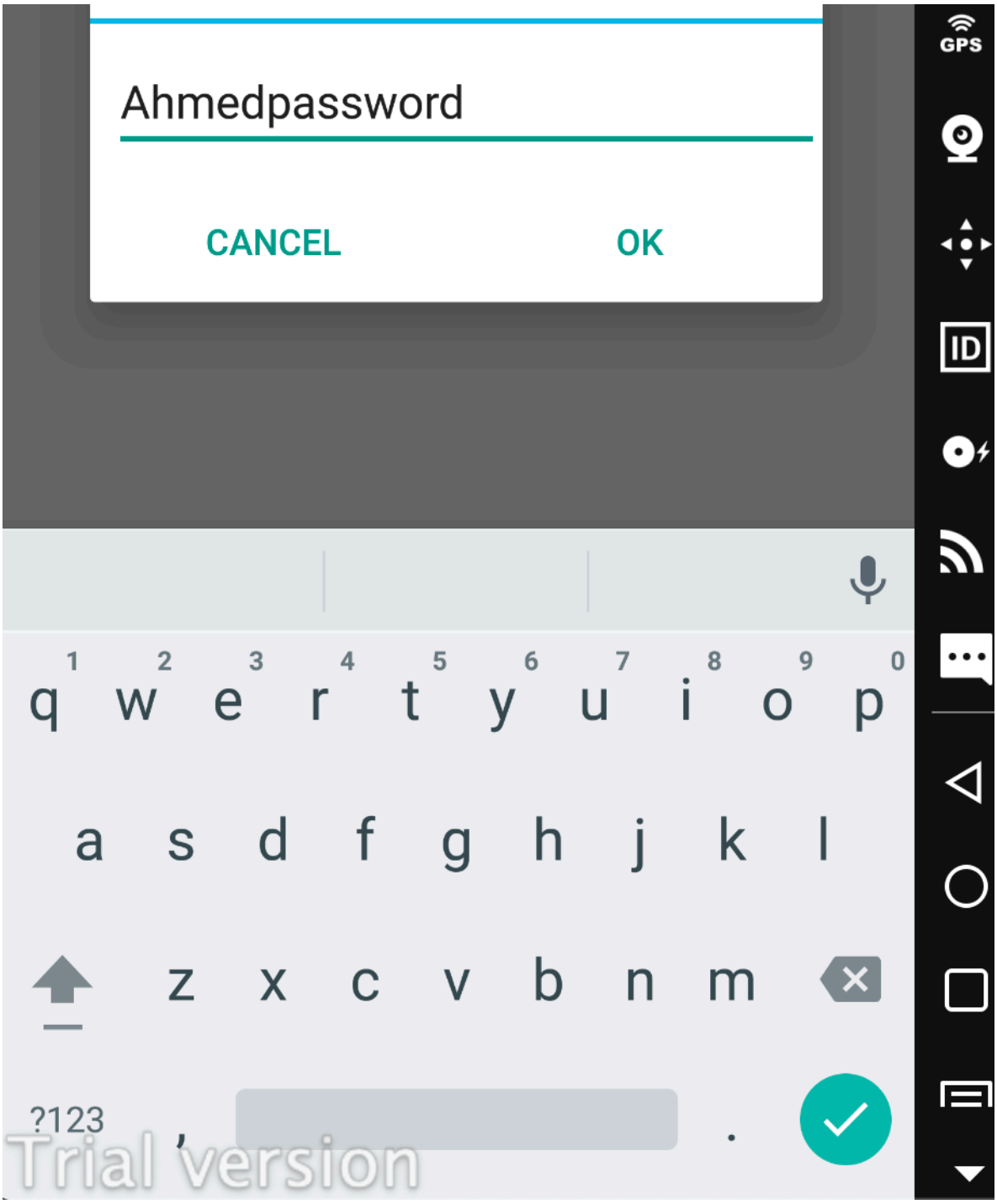
c

This probably should have before the last one , Here whatever was implemented inside the last activity wasnot implemented here

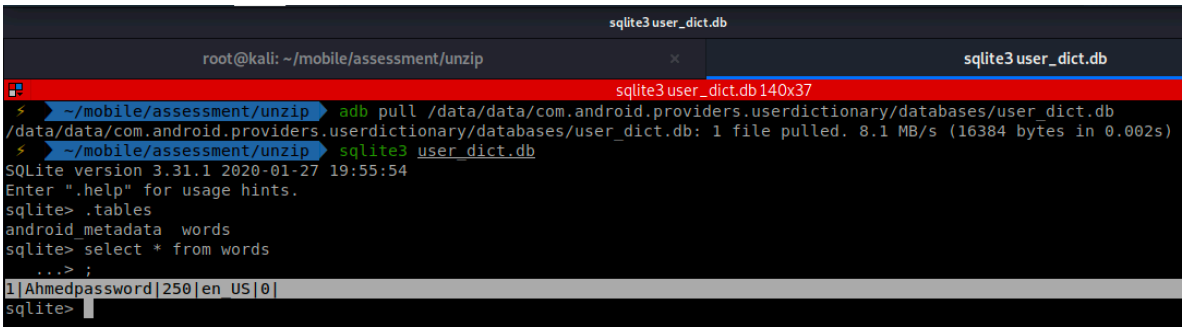



Ahmed pass word
Armed password
Aimed password
Ajmer password
Add to dictionary
Delete
sword
Trial version


Samsung Galaxy S6 (1440x2560, 640dpi) - 192.168.56.106 -...
4:18
Test Case to show... EditText
Add to dictionary



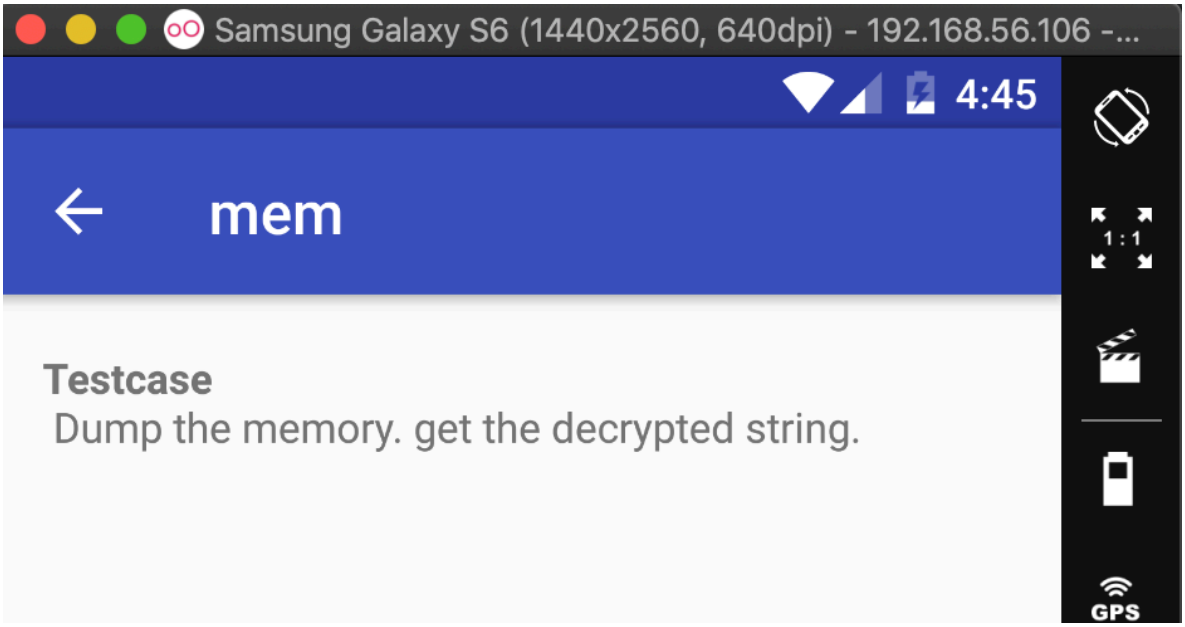
Here we can pull this from the dictionary with this



Recommendations:
Always disable adding any sensitive information from being added in the dictionary as they can be accessed by any other application.

MEM

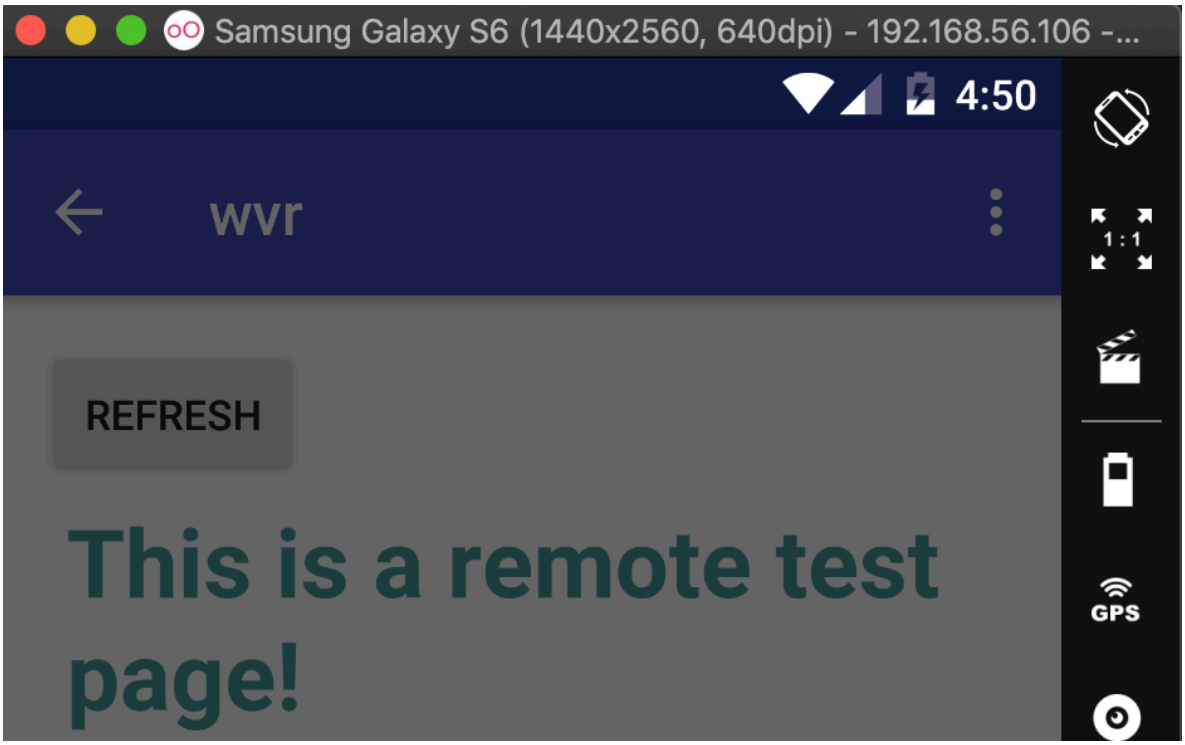
We try to do as instructed:

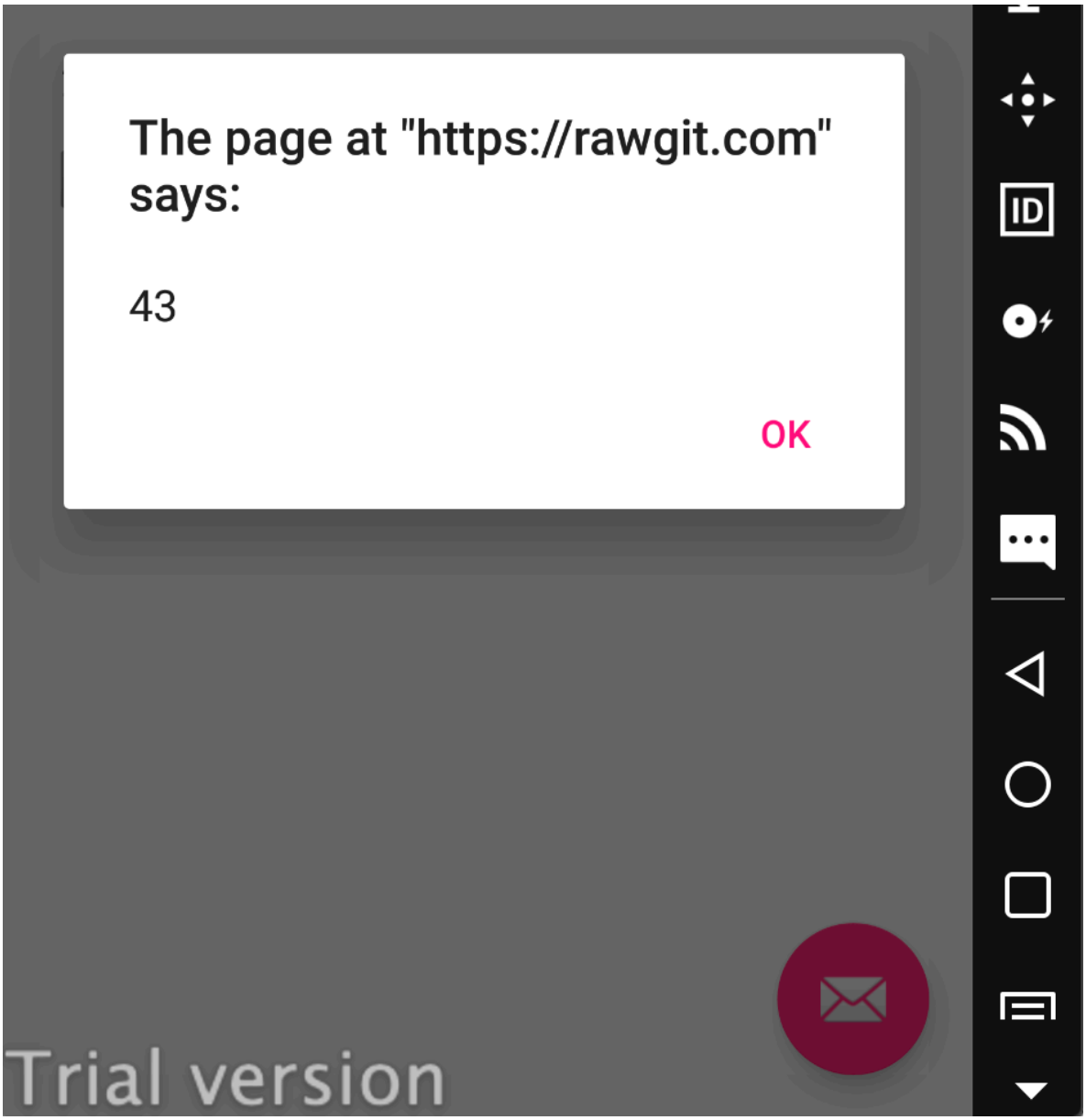


I think this needs android studio. Could not do it :'(

WVR

Once we open this activity we are greeted with XSS




The page at "https://rawgit.com" says:
43
OK
Trial version

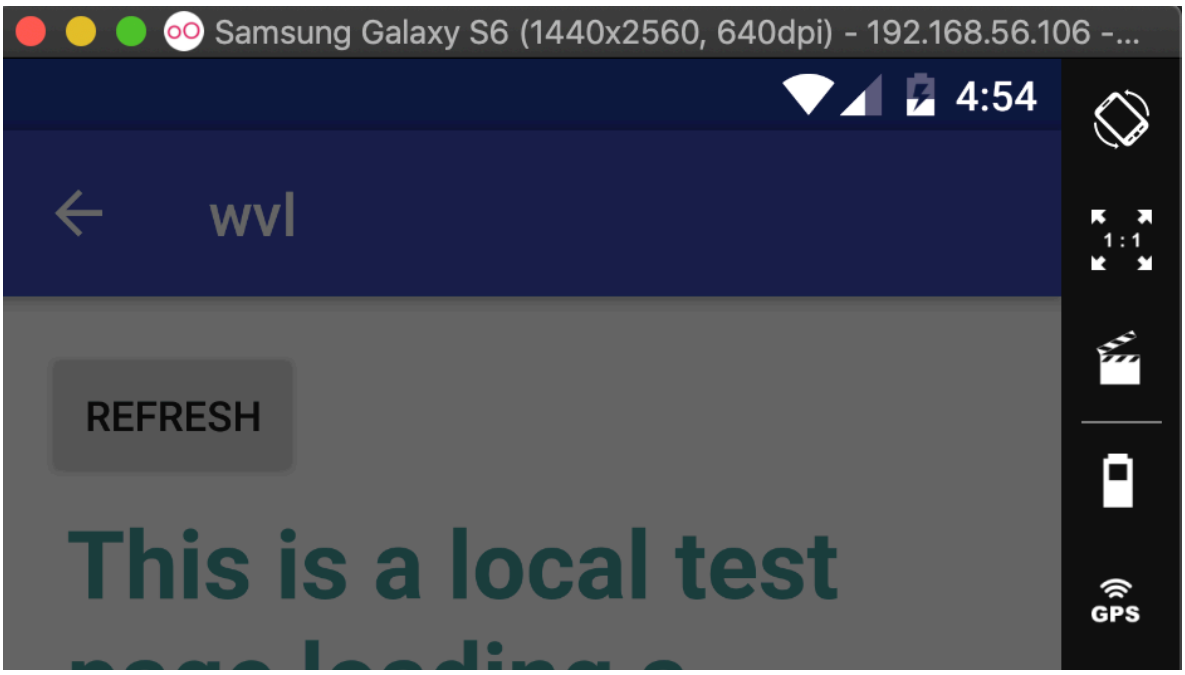
sg.vp.owasp_mobile.funhack.wvr

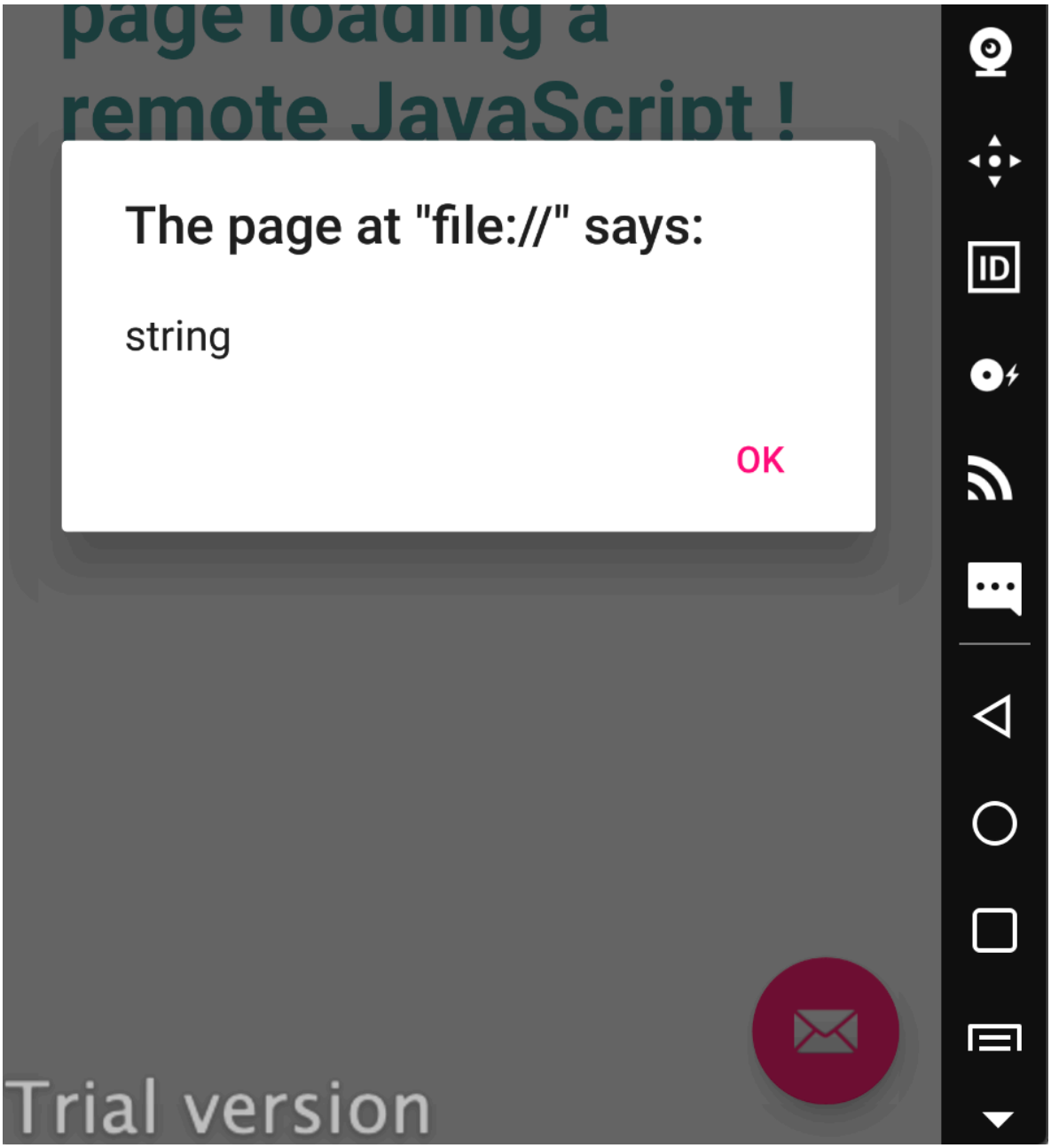
```java
package sg.vp.owasp_mobile.funhack;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebChromeClient;
import android.webkit.WebView;
import android.widget.Button;

public class wvr extends AppCompatActivity {
    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_omtg__env_005__web_view);
        setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        ((Button) findViewById(R.id.button1)).setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                ((WebView) wvr.this.findViewById(R.id.webView1)).reload();
            }
        });
        WebView webView = (WebView) findViewById(R.id.webView1);
        webView.setWebChromeClient(new WebChromeClient());
        webView.getSettings().setJavaScriptEnabled(true);
        webView.addJavascriptInterface(new JS(this), "Android");
        webView.loadUrl("https://rawgit.com/sushi2k/AndroidWebView/master/webview.htm");
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_my, menu);
        return true;
    }
```

This enables JS

Recommendation:
Remove this line of code as by default JS is disabled. This is to prevent any sort of exploits from remote websites.

WVL




page loading a
remote JavaScript !
The page at "file://" says:
string
OK
Trial version

sg.vp.owasp_mobile.funhack.wvl

```java
import android.webkit.JavaScriptInterface;
import android.webkit.WebChromeClient;
import android.webkit.WebView;
import android.widget.Button;

public class wvl extends AppCompatActivity {

    final class JavaScriptInterface {
        JavaScriptInterface() {
        }

        @JavascriptInterface
        public String getSomeString() {
            return "string";
        }
    }

    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_omtg__env_005__web_view__local);
        setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        ((Button) findViewById(R.id.button2)).setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                ((WebView) wvl.this.findViewById(R.id.webView2)).reload();
            }
        });
        WebView webView = (WebView) findViewById(R.id.webView2);
        webView.getSettings().setJavaScriptEnabled(true);
        webView.getSettings().setAllowFileAccessFromFileURLs(true);
        webView.setWebChromeClient(new WebChromeClient());
        webView.addJavascriptInterface(new JavaScriptInterface(), "jsinterface");
        webView.loadUrl("file:///android_asset/local.htm");
    }
}
```

This allows files to be Read from URLs

This is similar to Path traversal. We can see any files we have access to on the device from this URL

Recommendation:
Remove this line of code.
If needed try to whitelist only the inteded files.
Try not to take this input from the user.

ONEBP

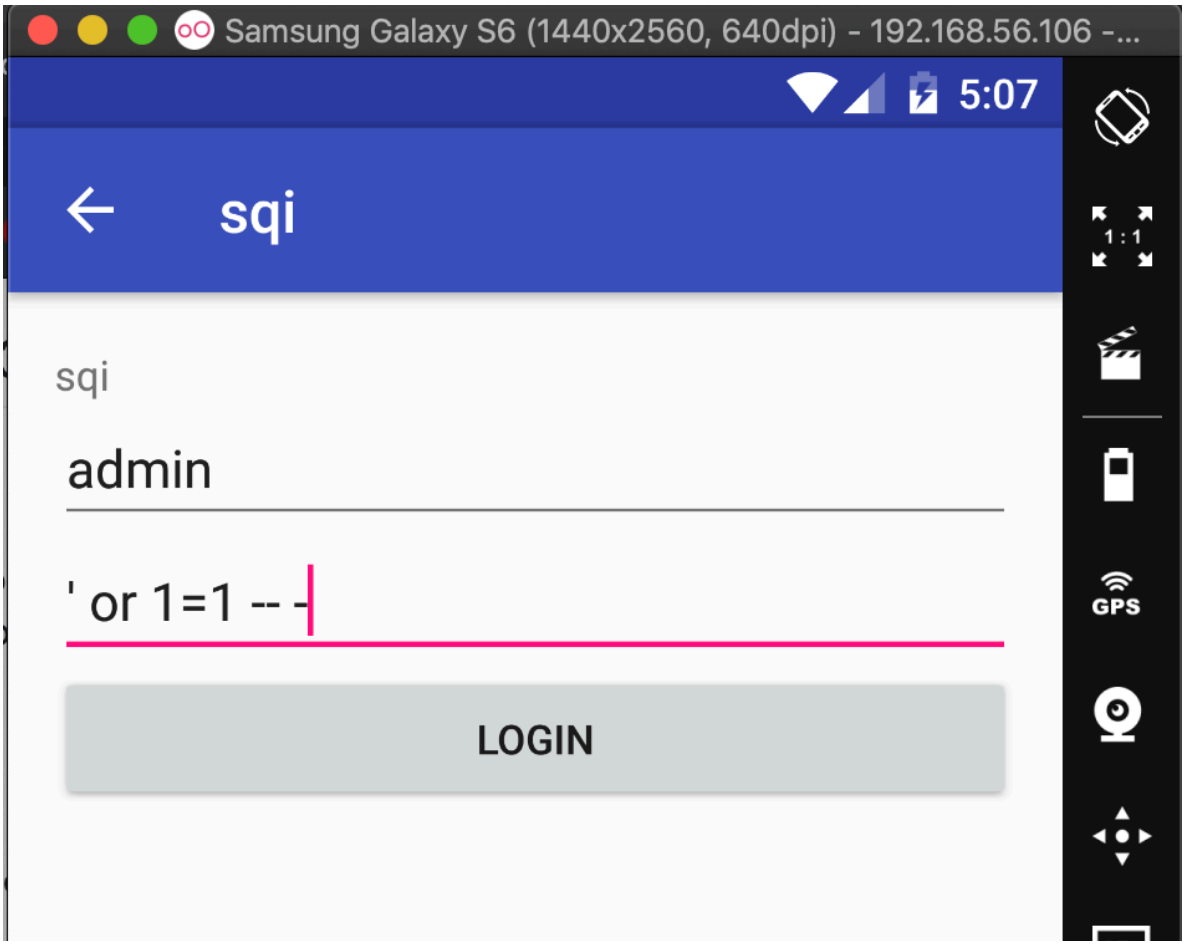
From the code this seems to be the best practice to do authentication with raw sql query.
A lot of checks and filters are being done on the input to prevent injections.

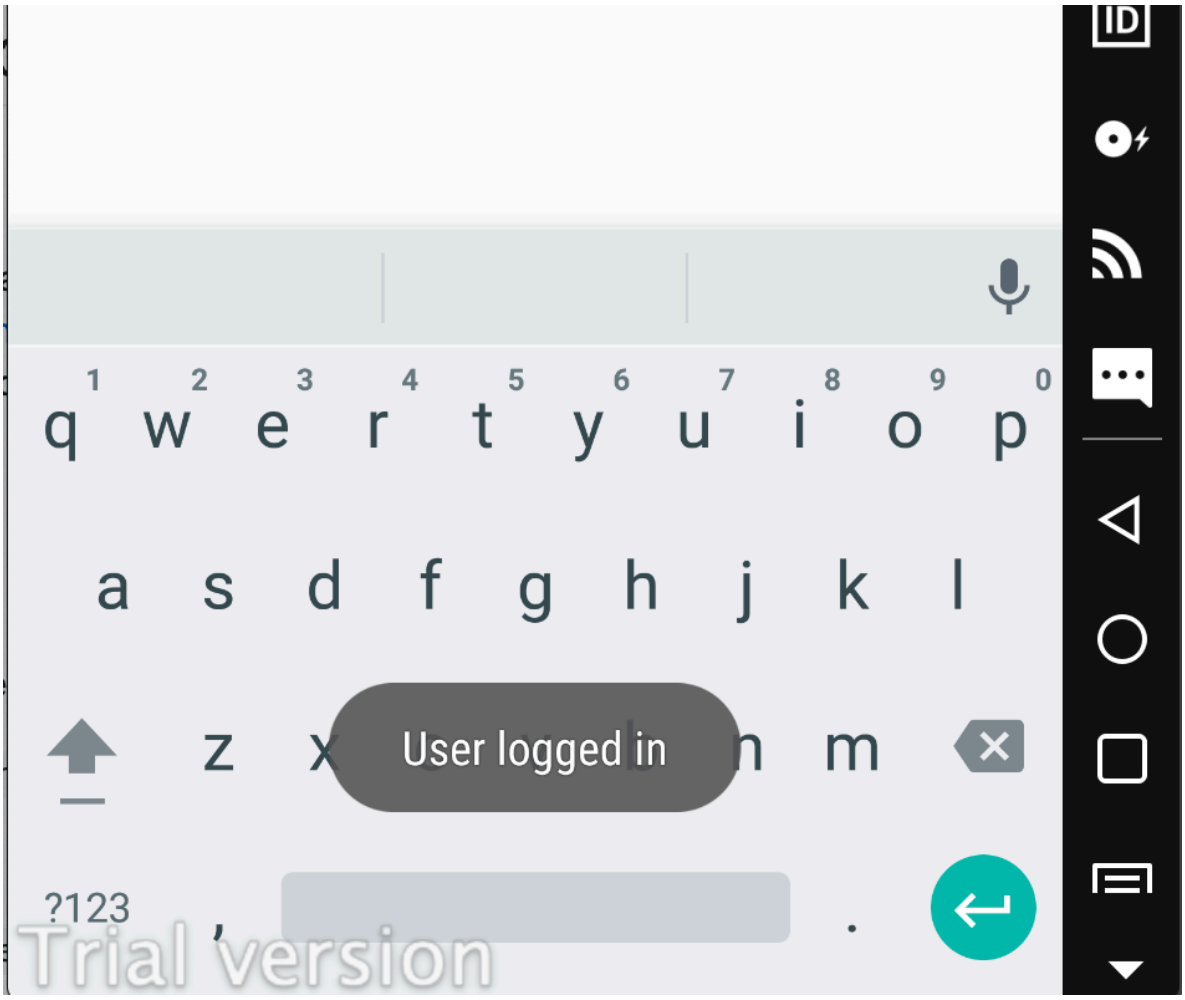
sg.vp.owasp_mobile.funhack.oneBP

```java
/* access modifiers changed from: private */
public boolean checkLogin(String str, String str2) {
    boolean z;
    Throwable th;
    th = null;
    boolean z2 = true;
    boolean z3 = false;
    try {
        Cursor rawQuery = openOrCreateDatabase("authentication-best-practice", 0, null).rawQuery("SELECT * F
        if (rawQuery != null) {
            try {
                if (!rawQuery.moveToFirst()) {
                    z2 = false;
                }
                try {
                    rawQuery.close();
                } catch (Throwable th2) {
                    th = th2;
                    z = z2;
                    if (rawQuery != null) {
                        if (th != null) {
                            try {
                                rawQuery.close();
                            } catch (Throwable th3) {
                                th.addSuppressed(th3);
                            }
                        } else {
                            rawQuery.close();
                        }
                    }
                    try {
                        throw th;
                    } catch (Exception e) {
                        e = e;
```

SQI

Here we can bypass authentication using SQLi





Recommendation:
Either follow the best practice to use the Raw sql query or use prepare statements (paramaterization)

PROVIDER

Here we go exploit content providers

We discover that the content provider is vulnerable to sql injection , we need to pass the parameters from the phone
using those commands o=

we insert a new value like this
content insert --uri content://sg.vp.owasp_mobile.provider.College/students --bind name:s:ahmed --bind grade:s:ahmed

we can dump the database through this
content query --uri content://sg.vp.owasp_mobile.provider.College/students --where "name='ahmed') OR 1=1--'"

```
adb shell 140x37
query --uri content://sg.vp.owasp_mobile.provider.College/students          <
Row: 0 _id=1, name=, grade=1
Row: 1 _id=2, name=, grade=1
.College/students --bind name:s:ahmed --bind grade:s:ahmed                  <
vp.owasp_mobile.provider.College/students --where "name='ahmed') OR 1=1--'"  <
Row: 0 _id=1, name=, grade=1
Row: 1 _id=2, name=, grade=1
Row: 2 _id=3, name=, grade=ahmed
Row: 3 _id=4, name=ahmed, grade=ahmed
root@vbox86p:/ #
```

Recommendations:
Same recommendations as last challenge

CI

Here we have code injection

sg.vp.owasp_mobile.funhack.ci

```java
package sg.vp.owasp_mobile.funhack;

import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import dalvik.system.DexClassLoader;

public class ci extends AppCompatActivity {
    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_omtg__coding_004__code__injection);
        setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        try {
            StringBuilder sb = new StringBuilder();
            sb.append(Environment.getExternalStorageDirectory());
            sb.append("/libcodeinjection.jar");
            Class loadClass = new DexClassLoader(sb.toString(), getDir("dex", 0).getAbsolutePa
            Log.e("Test", (String) loadClass.getMethod("returnString", new Class[0]).invoke(lo
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

This code is loaded and executed from the external storage

what we need to do is create a code with the same name and compile it with a java compile to jar. Then overwrite the file.

Recommendation :
Highly unprefered to include code from external sources inside your application , try to remove this.

SC

```java
sg.vp.owasp_mobile.funhack.sc
package sg.vp.owasp_mobile.funhack;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.webkit.WebView;

public class sc extends AppCompatActivity {
    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_omtg__netw_001__secure__channel);
        setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        WebView webView = (WebView) findViewById(R.id.webView2);
        ((WebView) findViewById(R.id.webView1)).loadUrl(getResources().getString(R.string.url_example));
        webView.loadUrl(getResources().getString(R.string.url_example_ssl));
    }
}
```

We need to check those two URLs

```
root@kali: ~/mobile/assessment/assesment/res/values 140x37
~/mobile/assessment/assesment/res/values  ls
attrs.xml  bools.xml  colors.xml  dimens.xml  drawables.xml  ids.xml  integers.xml  public.xml  strings.xml  styles.xm
~/mobile/assessment/assesment/res/values  cat strings.xml | grep url
<string name="url_example">http://example.com</string>
<string name="url_example_ssl">https://example.com</string>
~/mobile/assessment/assesment/res/values  \
```

We find that he uses insecure channels (http instead of https in the first webview)

SSLP & SSLPWC

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

Those two challenges are related to SSL pinning --> need further study.