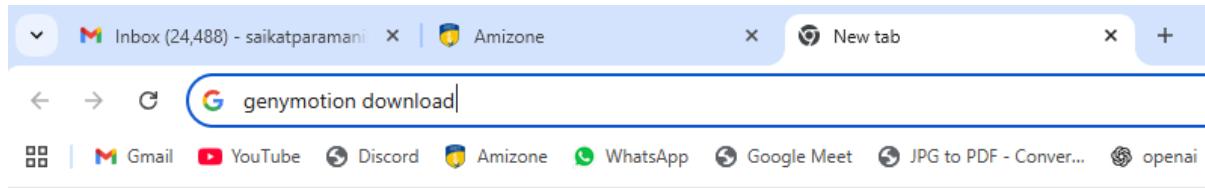
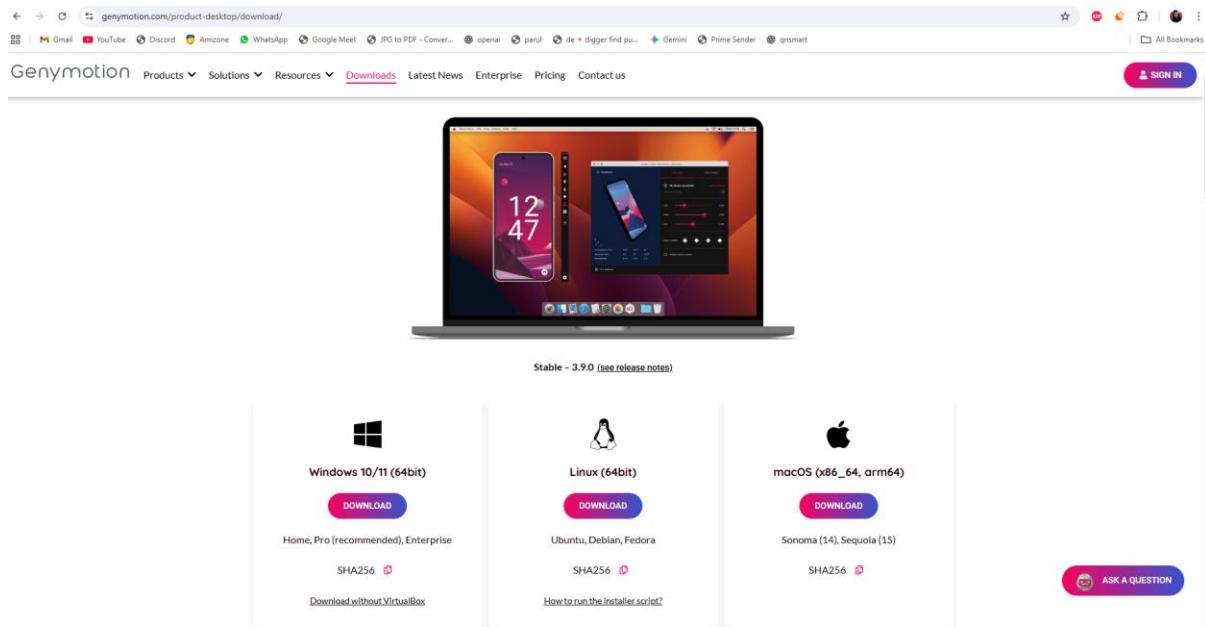


DIVA (Damn Insecure and Vulnerable App)



Download Genymotion

- Visit the official Genymotion download page:
<https://www.genymotion.com/product-desktop/download/>
- Choose the correct installer based on your operating system (Windows, Linux, macOS).
- For Windows, download the **Genymotion 64-bit version** unl

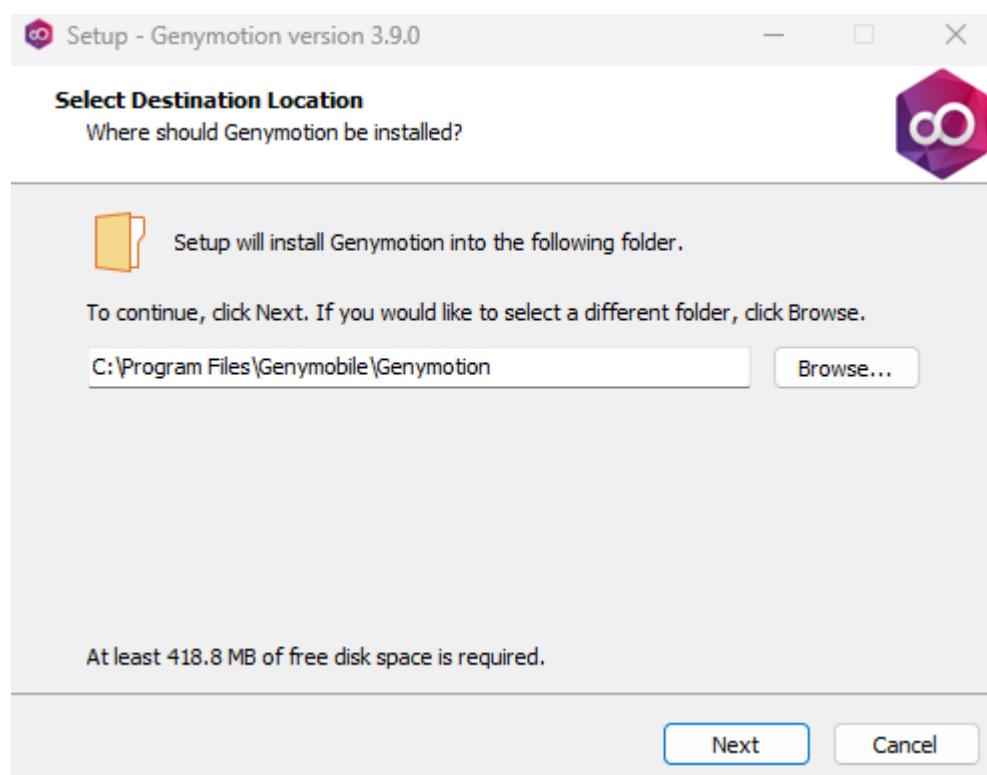
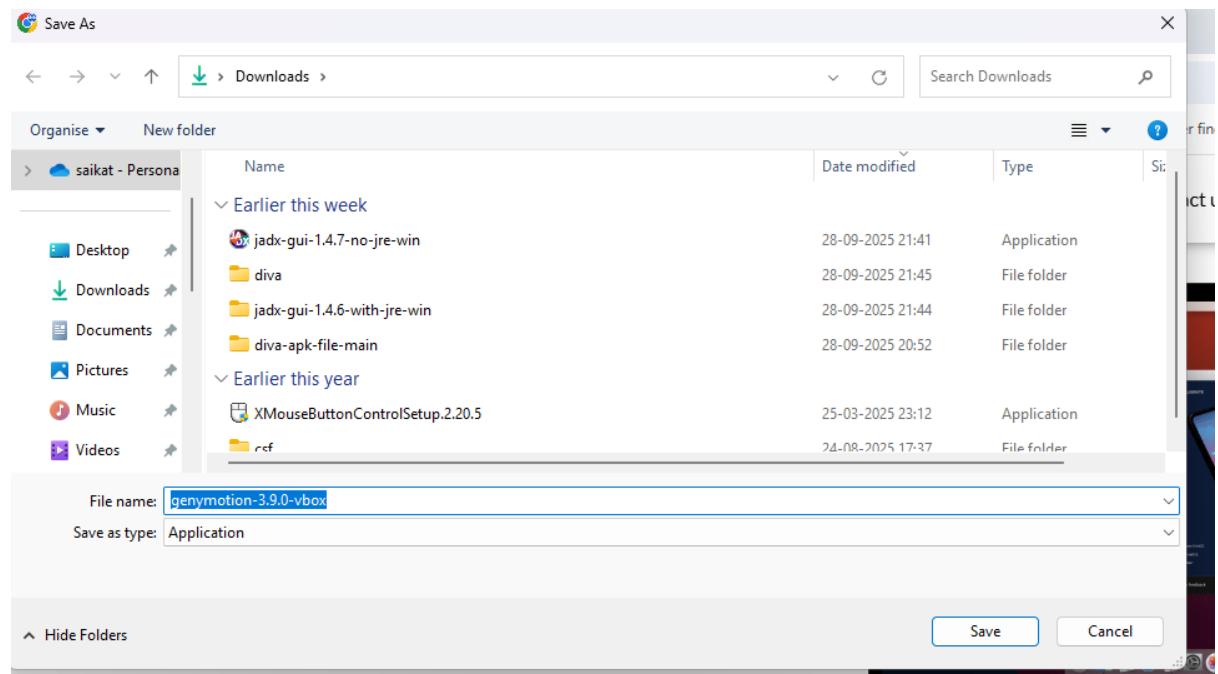


Install Genymotion

Run the downloaded installer (genymotion-3.9.0-vbox.exe).

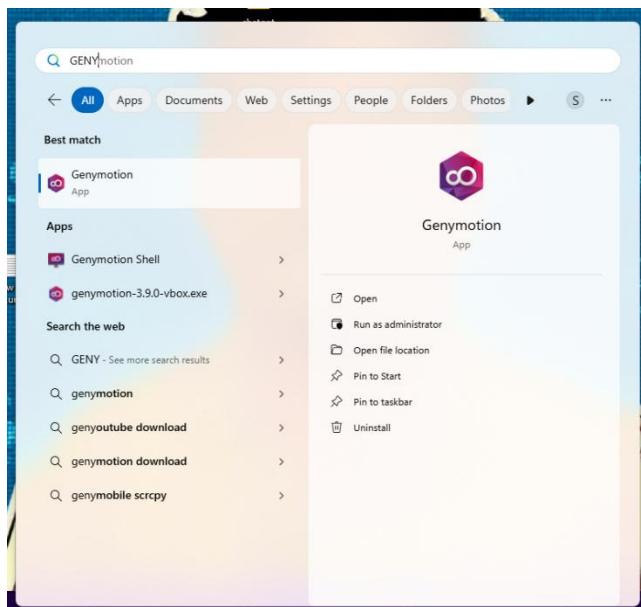
Select the installation destination (default: C:\Program Files\Genymobile\Genymotion).

Click Next and follow the installation wizard until setup completes.



Launch Genymotion

- After installation, search for Genymotion in the Windows start menu.
- Click to launch the emulator.



Genymotion First Launch & License Setup

When Genymotion is launched for the first time, it requires user login and license selection.

Step 4: Sign in to Genymotion

- You will be greeted with the Welcome to Genymotion screen.
- Enter your registered email and password to sign in. (If you don't have an account, create one on the Genymotion website.)

Welcome to Genymotion



 Enter your email

 Enter your password 

 View proxy options 

Select License Type

- Genymotion requires a license to use.
- For educational and practice purposes, choose Personal Use.
- Click Next to continue.

Genymotion requires a license



Use of Genymotion requires a license

Genymotion is a professional tool for which all kinds of profit-making businesses need a valid license. A very light version of Genymotion is available without a license, but strictly restricted to a personal use.

[Buy a license \(if you don't already have one\)](#)

I have a license

Personal Use

Creating and Configuring a Virtual Device

After logging in and selecting license type, you must create an Android virtual device inside Genymotion.

Step 6: Create a Virtual Device

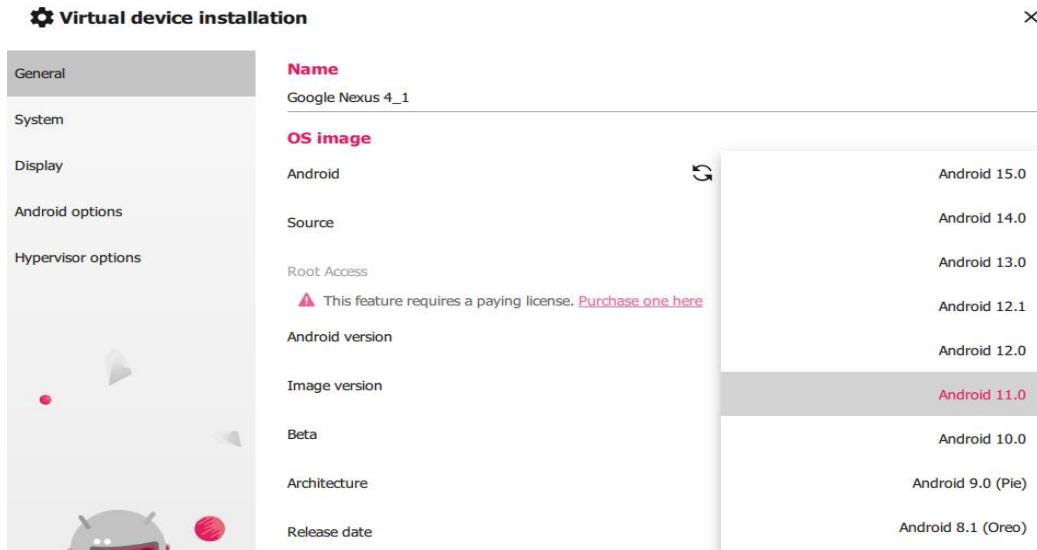
- From the main Genymotion window, click the Create button.
- A list of device templates appears (e.g., Google Nexus, HTC One, Motorola Moto X, etc.).
- Select a device template according to your preference (commonly Google Nexus 4/5).



Virtual device installation						
Filters	Type	Name	Display size	Resolution	Density	Source
<input type="text"/> Search		Custom Phone	4.7 inches	768 x 1280	320 - XHDPI	Genymotion
Form factor		Google Nexus 4	4.7 inches	768 x 1280	320 - XHDPI	Genymotion
Density		HTC One	4.7 inches	1080 x 1920	480 - XXHDPI	Genymotion
Size		Motorola Moto X	4.7 inches	720 x 1280	320 - XHDPI	Genymotion
Source		Samsung Galaxy S3	4.8 inches	720 x 1280	320 - XHDPI	Genymotion
		Google Nexus 5	4.95 inches	1080 x 1920	480 - XXHDPI	Genymotion

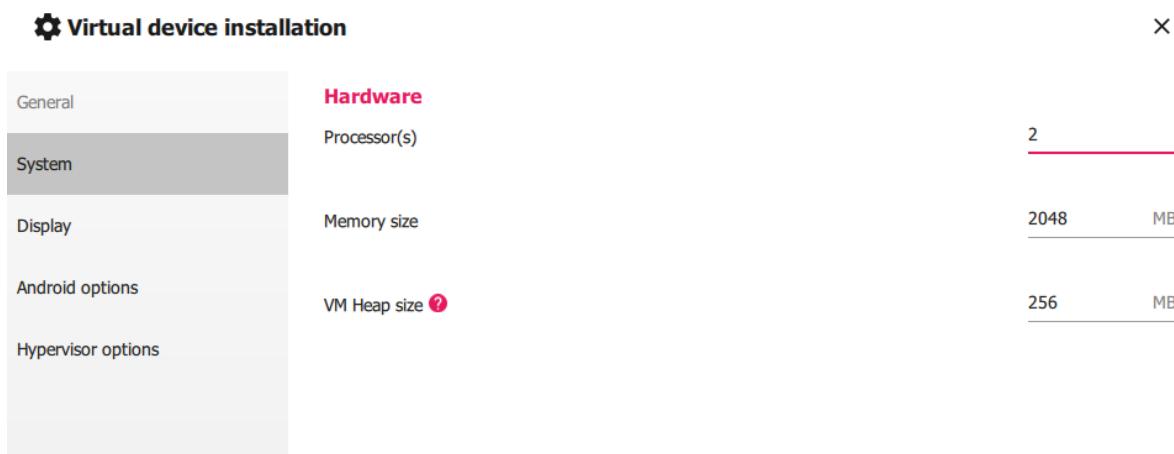
Select Android Version

- After choosing a device, select the Android OS version.
- Example: Selecting Android 11.0 for a Google Nexus 4 virtual device.



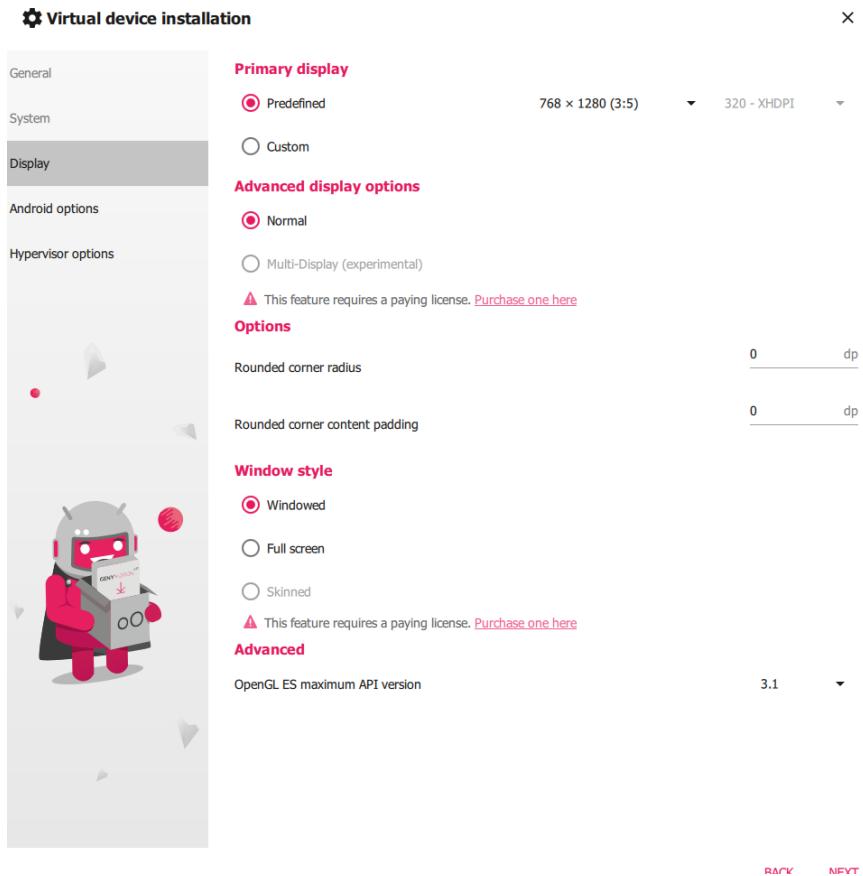
Configure System Resources

- Assign system hardware resources to the virtual device:
 - Processor(s): 2
 - Memory size: 2048 MB
 - VM Heap size: 256 MB
- These can be adjusted depending on your system capabilities.



Configure Display Options

- Keep everything Default
- Click on next



Configure Android Options

- Select Android options such as:

- Show Android navigation bar
- Use virtual keyboard (optional)

 Virtual device installation X

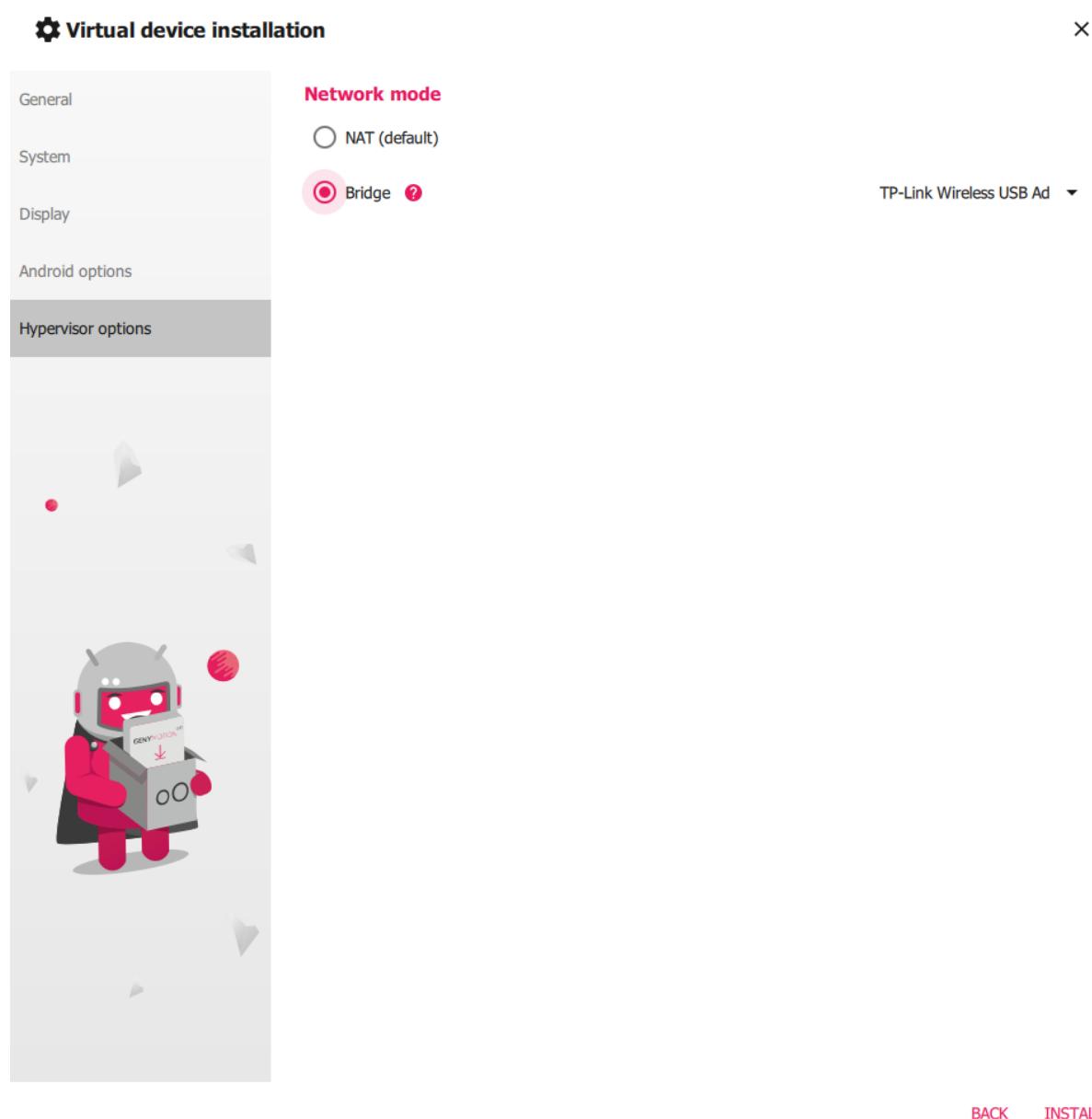
General	Input options
System	Show Android navigation bar <input checked="" type="checkbox"/>
Display	Use virtual keyboard for text input <input type="checkbox"/>
Android options	
Hypervisor options	



BACK NEXT

Configure Network (Hypervisor Options)

- Select the network mode:
 - Click on Bridge
 - Click on next



Install the Virtual Device

- After completing configuration, click Install.
- The virtual device will download and install. Progress can be seen at the bottom.
- Once complete, the new virtual device will be listed in Genymotion.

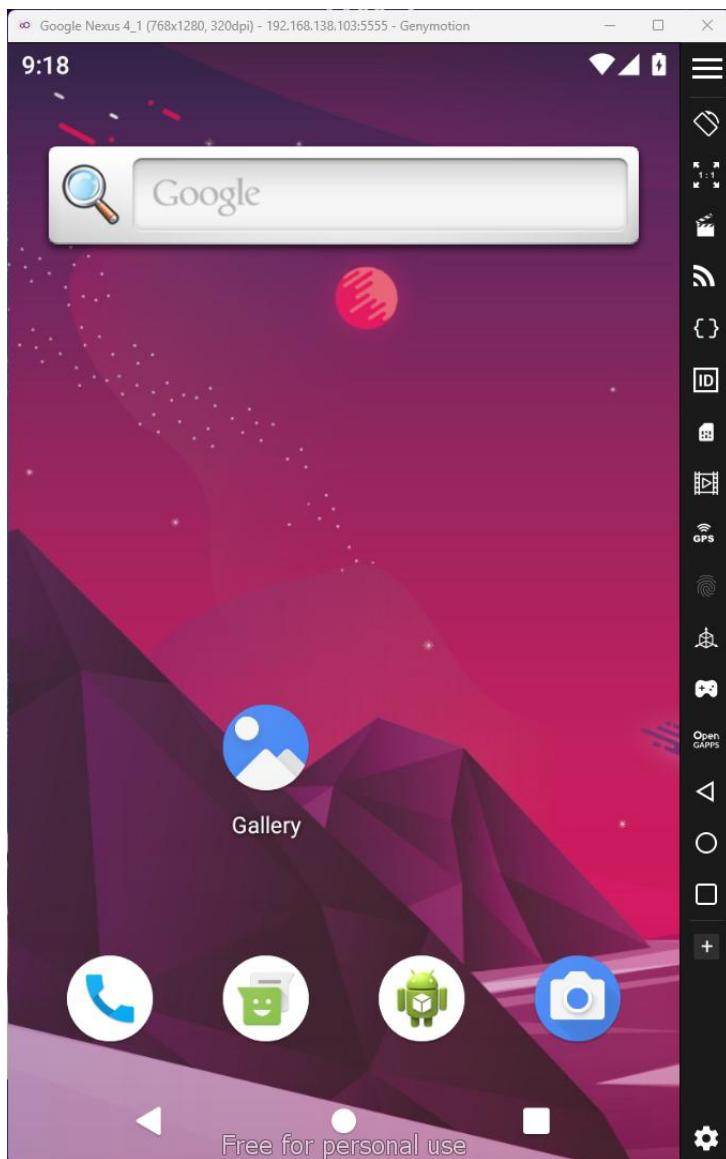
Type	Name	Android API	Resolution	Density	Size on disk	Source	Status	Actions
Google Nexus 4	Google Nexus 4_1				69MB / 581MB			X

Run the Virtual Android

- Click on play button under action

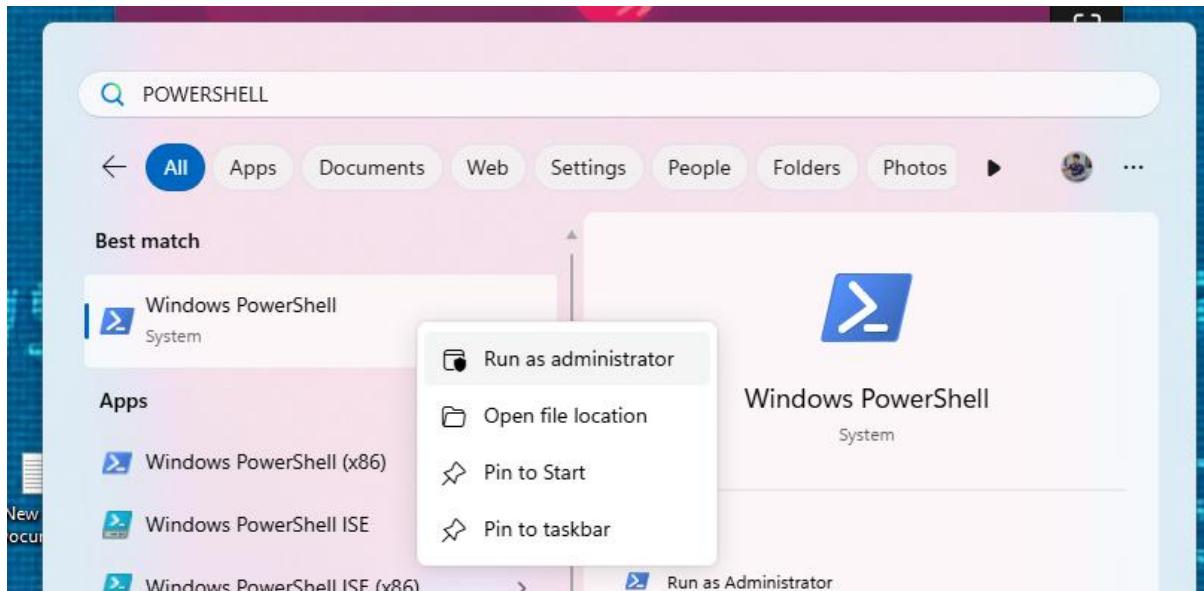
Type	Name	Android API	Resolution	Density	Size on disk	Source	Status	Actions
Google Nexus 4_1	11.0 - API 30	768 x 1280	320 - XHDPI	1.32 GB	Genymotion	Off		

Android will open



Open powershell

- Run as administrator



Verify ADB Installation

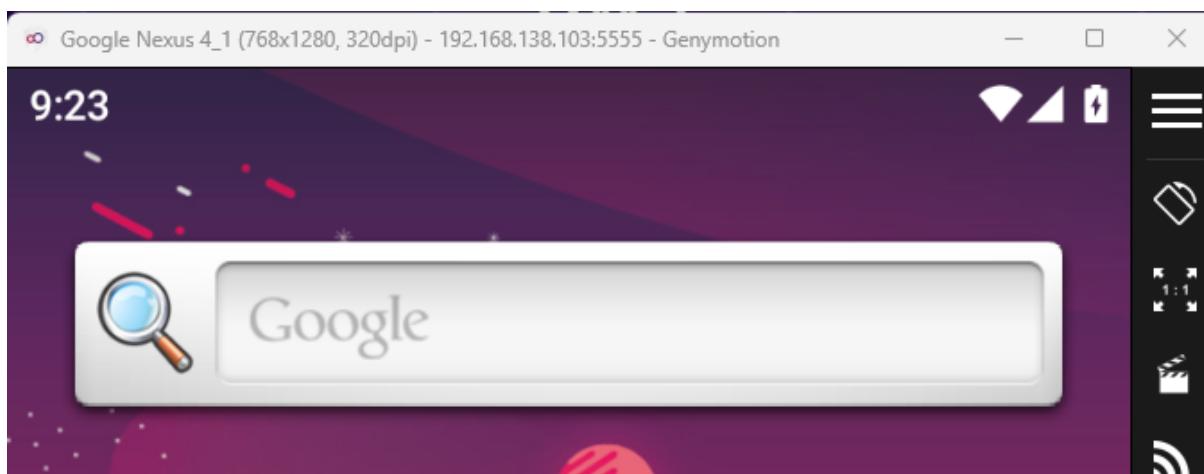
- Run the following command to check ADB version:
- & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" version

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" version
Android Debug Bridge version 1.0.41
Version 29.0.2-105
Installed as C:\Program Files\Genymobile\Genymotion\tools\adb.exe
PS C:\WINDOWS\system32>
```

Note the ip of the android from the emulator title bar (eg : 192.168.138.103:5555)



Connect ADB to the Device

- Copy the device IP and port shown in the emulator title bar (e.g., 192.168.138.103:5555).

Run: & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" connect

192.168.138.103:5555

```
PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" connect 192.168.138.103:5555
already connected to 192.168.138.103:5555
PS C:\WINDOWS\system32>
```

Verify Device Connection

- **Run:** & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" devices

```
PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" devices
List of devices attached
192.168.138.103:5555      device
```

Downloading and Extracting the DIVA APK

Next, we need to download the vulnerable APK file for DIVA.

Search and Download DIVA APK

- Open a web browser and search for diva apk github.
- Click on the GitHub repository: *0xArab/diva-apk-file*.
- On the repository page, click Download ZIP.



 GitHub
<https://github.com> › [diva-apk-file](#) ::

[OxArab/diva-apk-file: DIVA \(Damn insecure and vulnerable ...\)](#)

DIVA (Damn insecure and vulnerable App) is an App intentionally designed to be insecure. We are releasing the Android version of Diva.

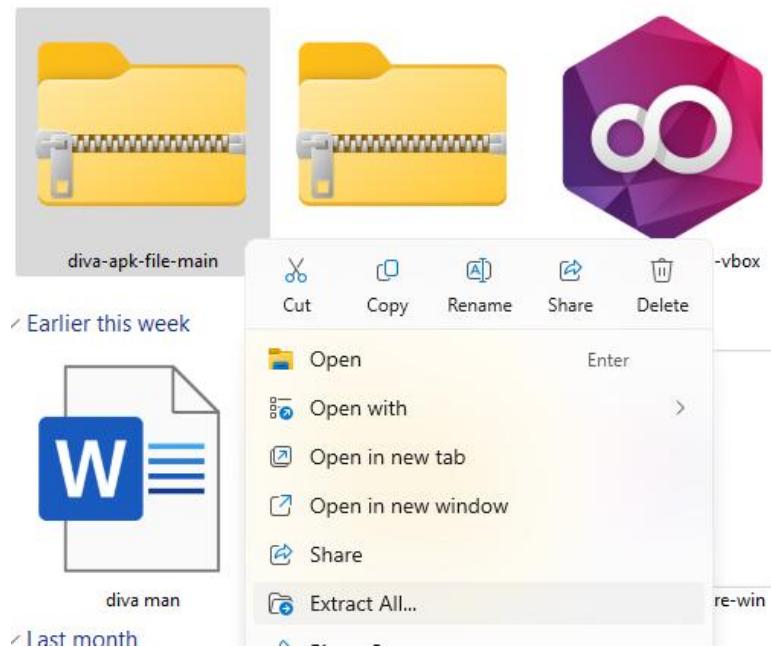
About

DIVA (Damn insecure and vulnerable App) is an App intentionally designed to be insecure

- Readme
- GPL-3.0 license
- Activity
- 28 stars
- 2 watching

Extract the ZIP File

- Go to the Downloads folder and locate the file `diva-apk-file-main.zip`.
- Right-click on it and select Extract All.
- This will create a folder containing the DIVA APK file that we can later install on the emulator.

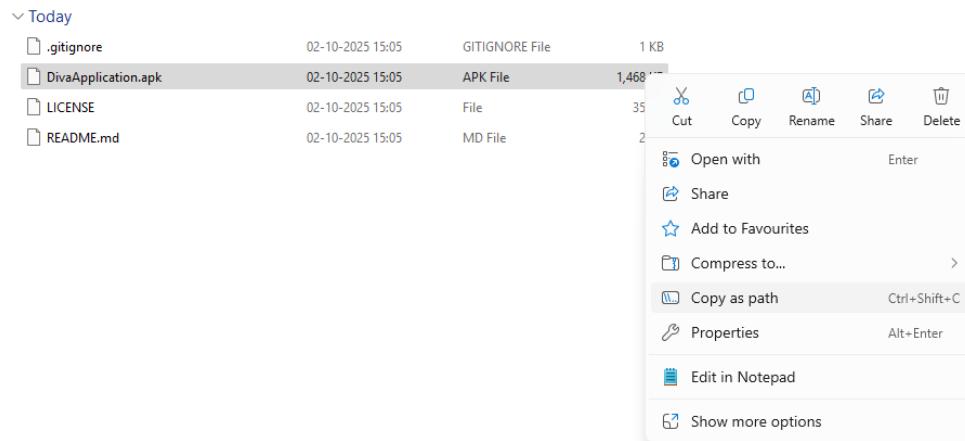


Installing the DIVA APK on Emulator

Once the APK file is extracted, we install it into the Genymotion virtual device using ADB.

Step 20: Locate the APK File

- Inside the extracted folder, locate DivaApplication.apk.
- Right-click on the file and select Copy as path to copy its full file location.



Install APK using ADB

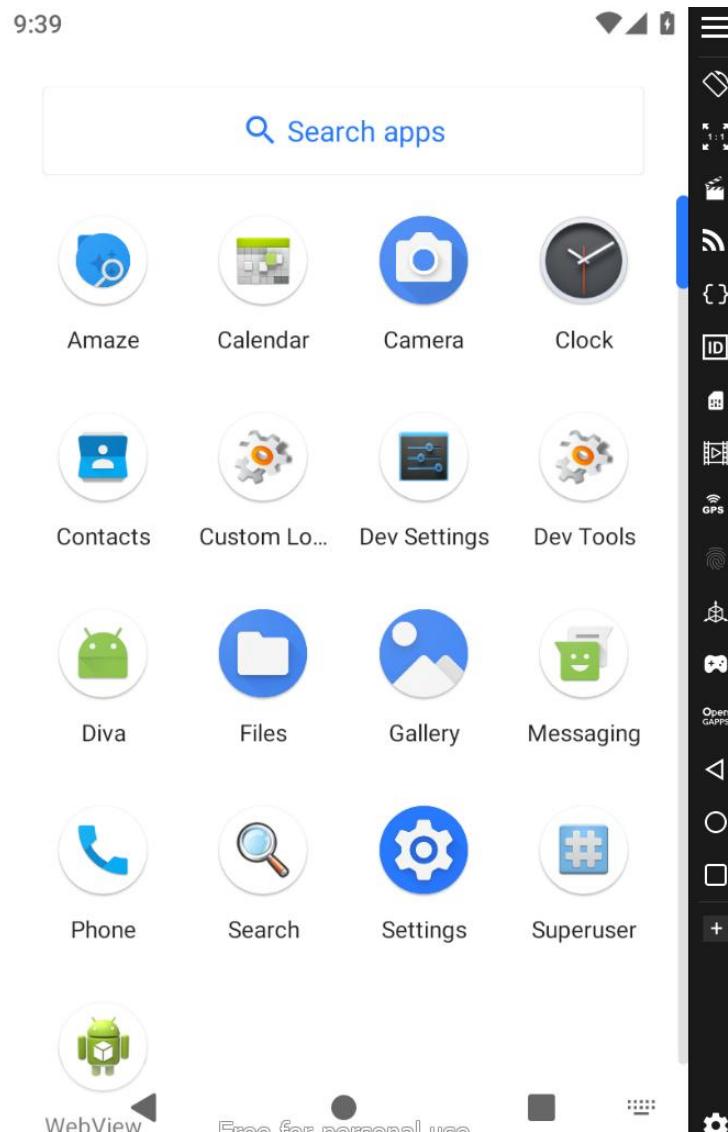
- Open PowerShell as Administrator.
- Use the following command format to install:

Run: & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" install
"C:\Users\saika\Downloads\diva-apk-file-main\diva-apk-file-
main\DivaApplication.apk"

```
PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" install  
"C:\Users\saika\Downloads\diva-apk-file-main\diva-apk-file-main\DivaApplication.apk"  
Performing Streamed Install  
Success  
PS C:\WINDOWS\system32>
```

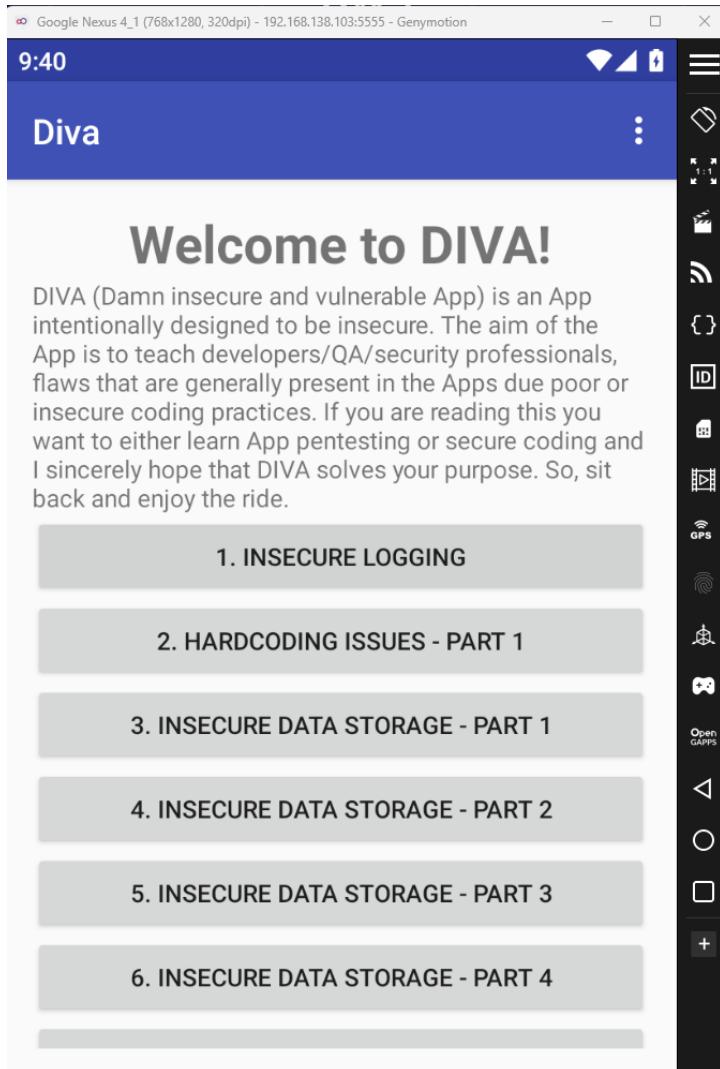
Verify Installation

- Open the app drawer inside the emulator.
- You should see the Diva app icon.



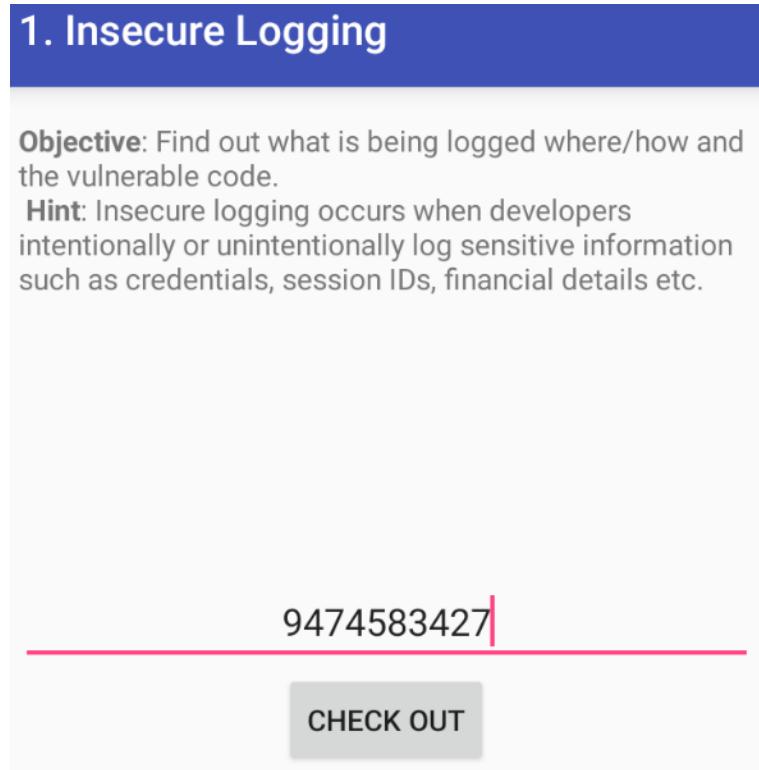
Open DIVA App

- Tap on the Diva icon.
- The welcome screen of DIVA will appear, showing a list of vulnerable challenges.



Open Insecure Logging Challenge

- From the DIVA app home screen, tap 1. Insecure Logging.
- You will see an input field where you can enter a value (e.g., a credit card number or ID) and press CHECK OUT.



Clear Previous Logs

- Before testing, clear the existing logcat logs to avoid confusion.

Run: & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" logcat -c

```
PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" logcat -c
PS C:\WINDOWS\system32>
```

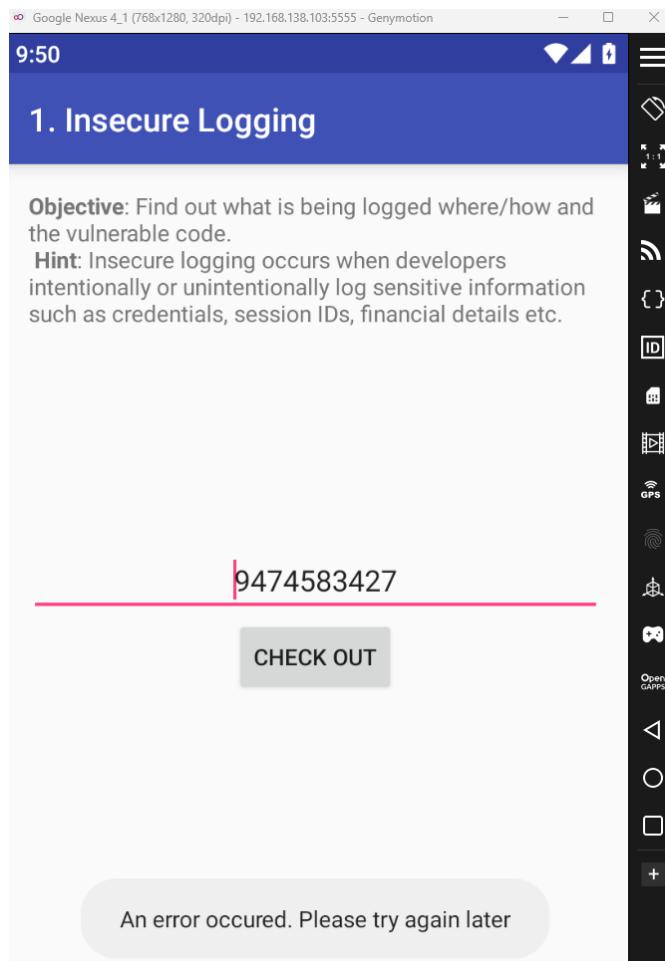
Monitor Logs

- Start monitoring logs in real-time:
- Run: & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" logcat

```
PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" logcat
----- beginning of main
10-02 09:48:57.905 634 737 D InputDispatcher: Waiting to send key to Window[9cb40ea u0 jakhar.aseem.diva/jakhar.aseem.diva.LogActivity] because there are unprocessed events that may cause focus to change
----- beginning of system
10-02 09:48:58.384 634 634 W WindowManager: removeWindowToken: Attempted to remove non-existing token: android.os.Binder@66725f1
10-02 09:48:59.400 634 737 D InputDispatcher: Waiting to send key to Window[9cb40ea u0 jakhar.aseem.diva/jakhar.aseem.diva.LogActivity] because there are unprocessed events that may cause focus to change
10-02 09:49:00.005 861 861 D KeyguardClockSwitch: Updating clock: 9D49
----- beginning of kernel
10-02 09:49:00.748 0 0 D logd : Skipping 18446744073709551471 entries from slow reader, pid 674, from LogBuffer::kickMe()
10-02 09:49:05.191 0 0 D logd : logdr: UID=0 GID=0 PID=2913 b tail=0 logMask=99 pid=0 start=0ns timeout=0ns
```

Trigger Logging Vulnerability

- Go back to the DIVA Insecure Logging screen.
- Enter a value (e.g., 9474583427) and press CHECK OUT.
- The app may show an error message on screen (e.g., *An error occurred. Please try again later*).



Analyze Logs

- Check the PowerShell logcat output , Press ctrl+f and type credit to find credit card details.
- You will find sensitive input (like the number you entered) logged in plain text, often with debug/error messages.
- Example log snippet:

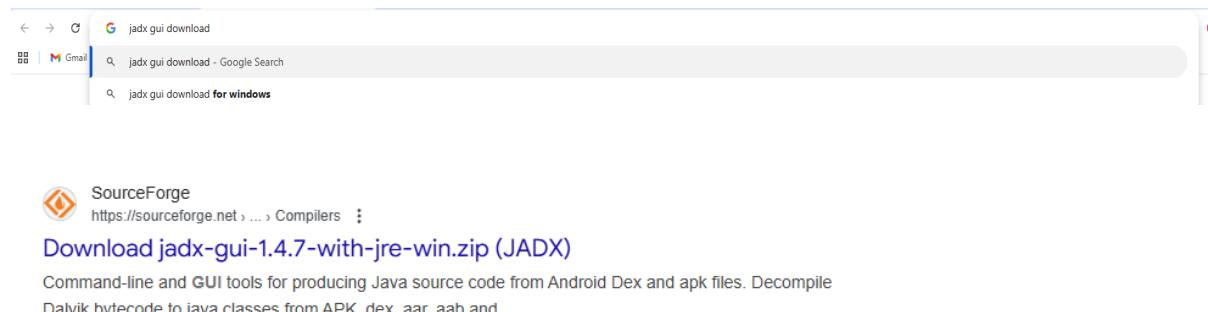
```
PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" logcat
----- beginning of main
10-02 09:48:57.905 634 737 D InputDispatcher: Waiting to send key to Window{9cb40ea u0 jakhar.aseem.diva/jakhar.aseem.diva.LogActivity} because there are unprocessed events
----- beginning of system
10-02 09:48:58.384 634 634 W WindowManager: removeWindowToken: Attempted to remove non-existing token: android.os.Binder@66725f1
10-02 09:48:59.400 634 737 D InputDispatcher: Waiting to send key to Window{9cb40ea u0 jakhar.aseem.diva/jakhar.aseem.diva.LogActivity} because there are unprocessed events
10-02 09:49:00.005 861 861 D KeypadClockSwitch: Updating clock: 9:49
----- beginning of kernel
10-02 09:49:00.748 0 0 D logd : Skipping 184467440873709551471 entries from slow reader, pid 674, from LogBuffer::kickMe()
10-02 09:49:05.191 0 0 D logd : logd: UID=0 GID=0 PID=2913 b tail=0 logMask=99 pid=0 start=0ns timeout=0ns
10-02 09:49:45.275 2796 2796 E diva-log: Error while processing transaction with credit card: 9474583427
10-02 09:49:45.277 634 649 I system_server: oneway function results will be dropped but finished with status OK and parcel size 4
```

Setting up JADX for Static Analysis

JADX is a tool used to decompile APK files and view the source code. We will use it to analyze DIVA's vulnerable code.

Download JADX

- Open a web browser and search for jadx gui download.
- Select the SourceForge link to download the latest version (e.g., jadx-gui-1.4.7-with-jre-win.zip).

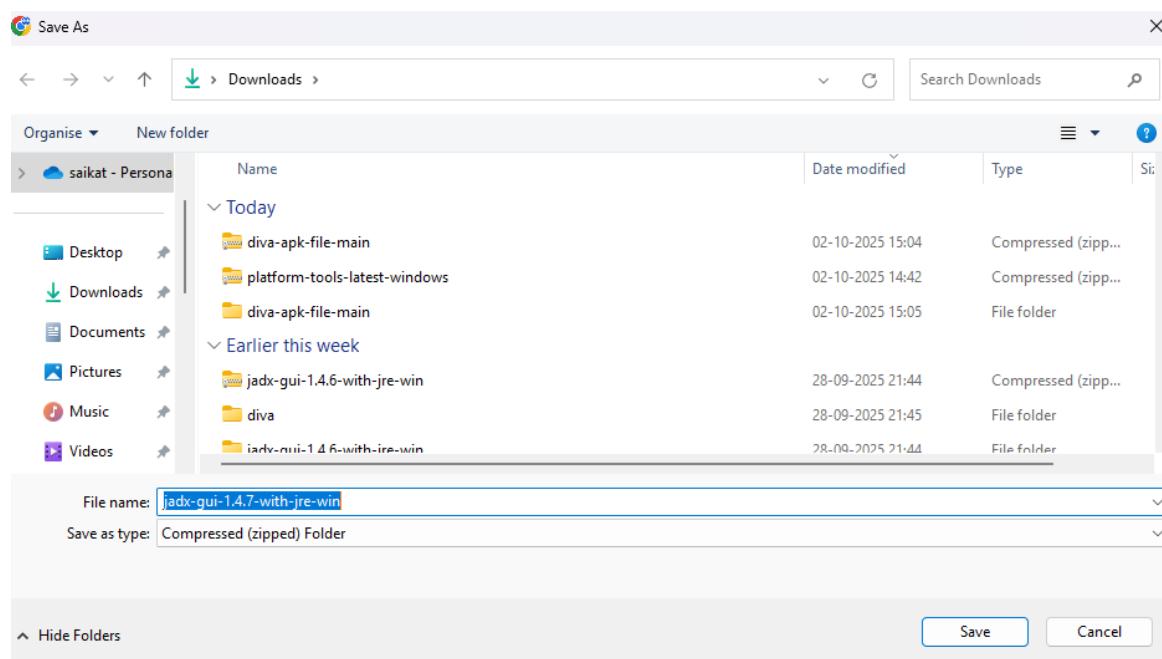


The screenshot shows a web browser window with the following details:

- Address bar: jadx gui download
- Search results:
 - jadx gui download - Google Search
 - jadx gui download for windows
- SourceForge page:
 - Page title: Download jadx-gui-1.4.7-with-jre-win.zip (JADX)
 - Description: Command-line and GUI tools for producing Java source code from Android Dex and apk files. Decompile Dalvik bytecode to java classes from APK, dex, aar, aab and ...

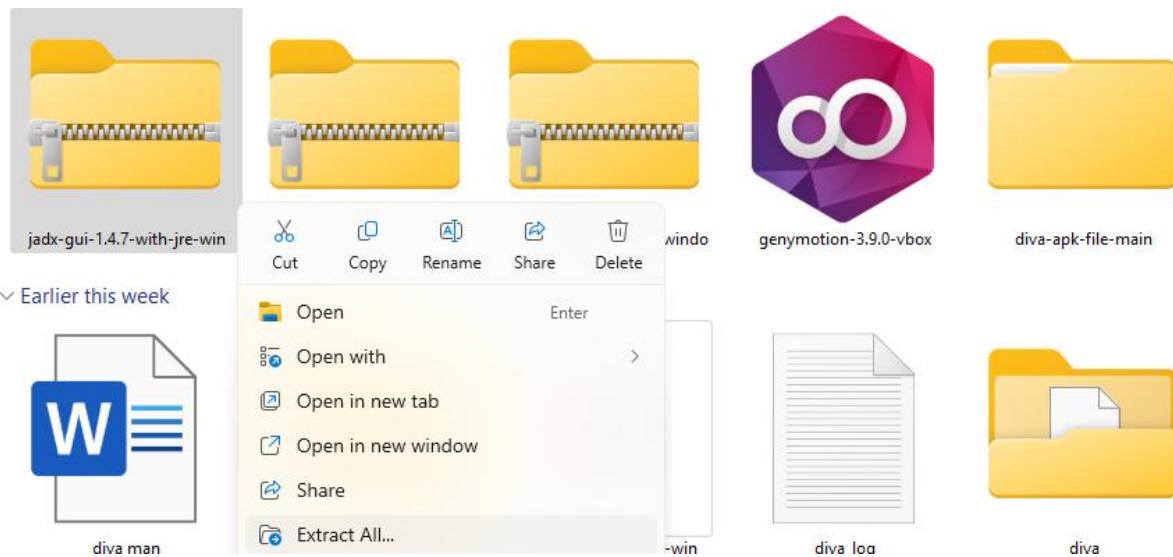
Save the JADX File

- Choose the Downloads folder as the save location.
- Save the compressed ZIP file.



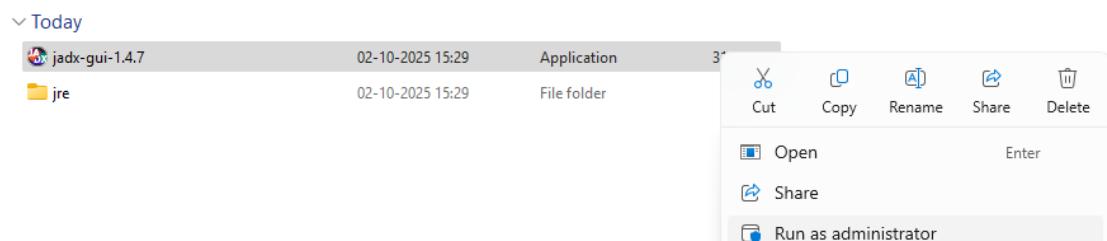
Extract the JADX ZIP

- Right-click on the downloaded file and choose Extract All.
- This creates a folder containing the JADX executable.



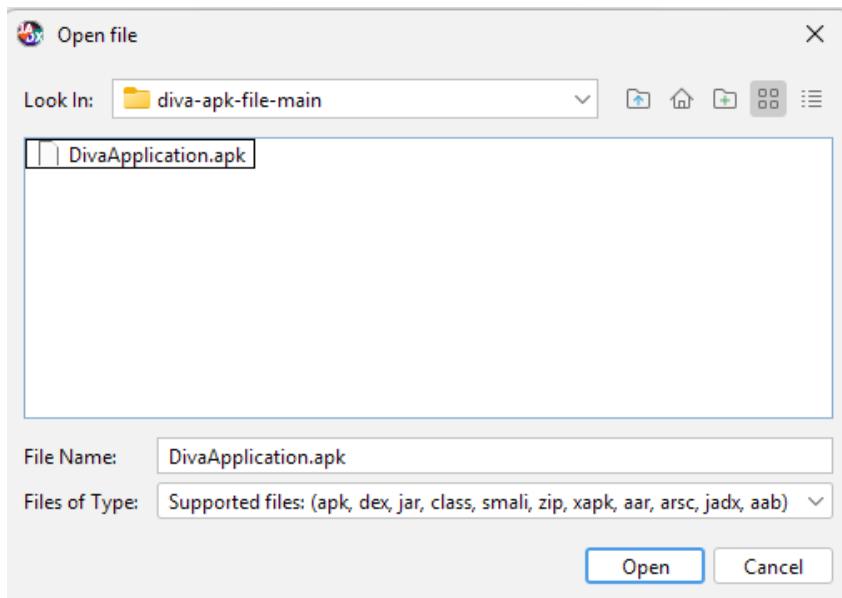
Run JADX

- Open the extracted folder and locate `jadx-gui-<version>` (e.g., `jadx-gui-1.4.7`).
- Right-click and select Run as Administrator.



Load DIVA APK in JADX

- In JADX, go to File > Open File.
- Browse to the folder where the DIVA APK (DivApplication.apk) is stored.
- Select the APK and click Open.



Static analysis of code why Insecure Login in DIVA is vulnerable

```
LogActivity x
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 {
    /* JAD INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityOnDonut, android.app.Activity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_log);
    }

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

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

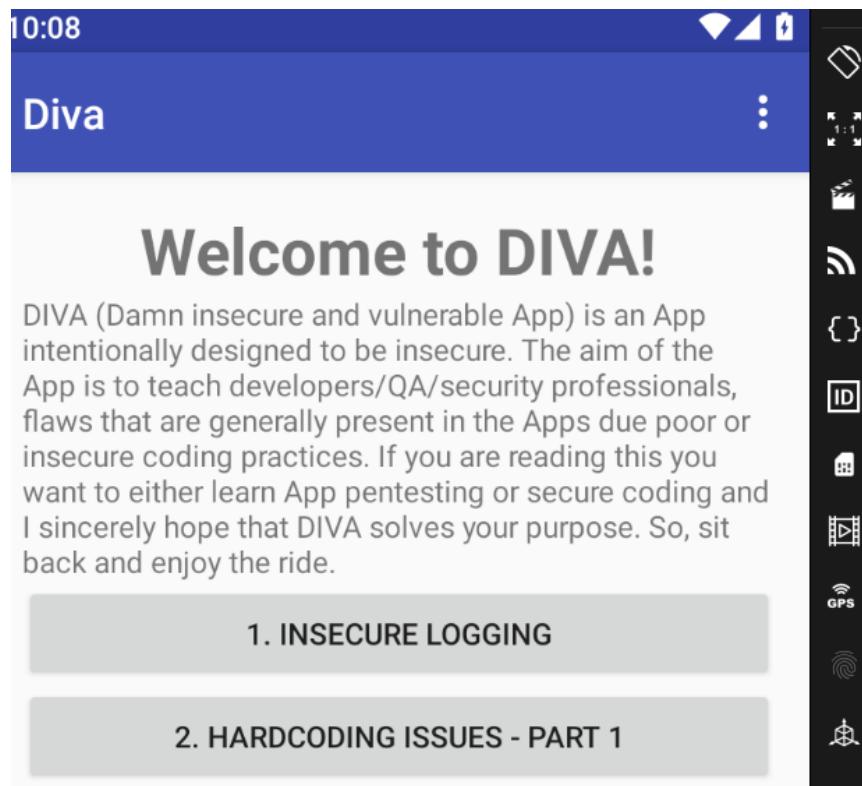
The code editor shows the Java source code for LogActivity.java. The code handles a credit card input field and processes it. A specific line of code is highlighted in yellow: `Log.e("diva-log", "Error while processing transaction with credit card: " + ctxt.getText().toString());`. This line logs the user's credit card number directly to the system log, which is a security vulnerability.

The code

`Log.e("diva-log", "Error while processing transaction with credit card: " + ctxt.getText().toString());` is insecure because it logs sensitive data (credit card number) directly into system logs. Logs can be accessed by attackers (via adb logcat, rooted devices, or other apps), which leads to data leakage and privacy violations.

Hardcoding Issues – Part 1

This option demonstrates how developers sometimes hardcode sensitive information inside the source code. The goal is to identify and extract it.



Entering random input – Access Denied

When we try random values in the app, it shows *Access Denied*, confirming that the app only accepts a specific secret key.

2. Hardcoding Issues - Part 1

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

A text input field contains the value "vkjbvlkjbsilvbkvblk". Below the input field is a gray button labeled "ACCESS". At the bottom of the screen, a message box displays the text "Access denied! See you in hell :D".

Code view in JADX – HardcodeActivity.java

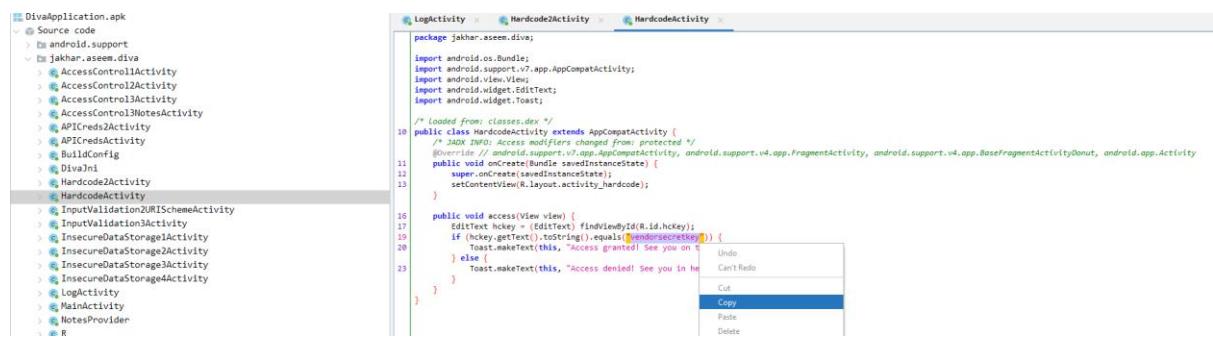
By decompiling the app using JADX, we find in HardcodeActivity.java that the secret key is hardcoded as:

```
if (key.getText().toString().equals("vendorsecretkey")) {
```

```
    Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
```

```
}
```

This means the secret value vendorsecretkey is directly written in the code.



```
LogActivity x Hardcode2Activity x HardcodeActivity x
package jakhar.aesem.dive;
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 {
    /* Java source and dex files changed from protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityDonut, android.app.Activity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hardcode);
    }

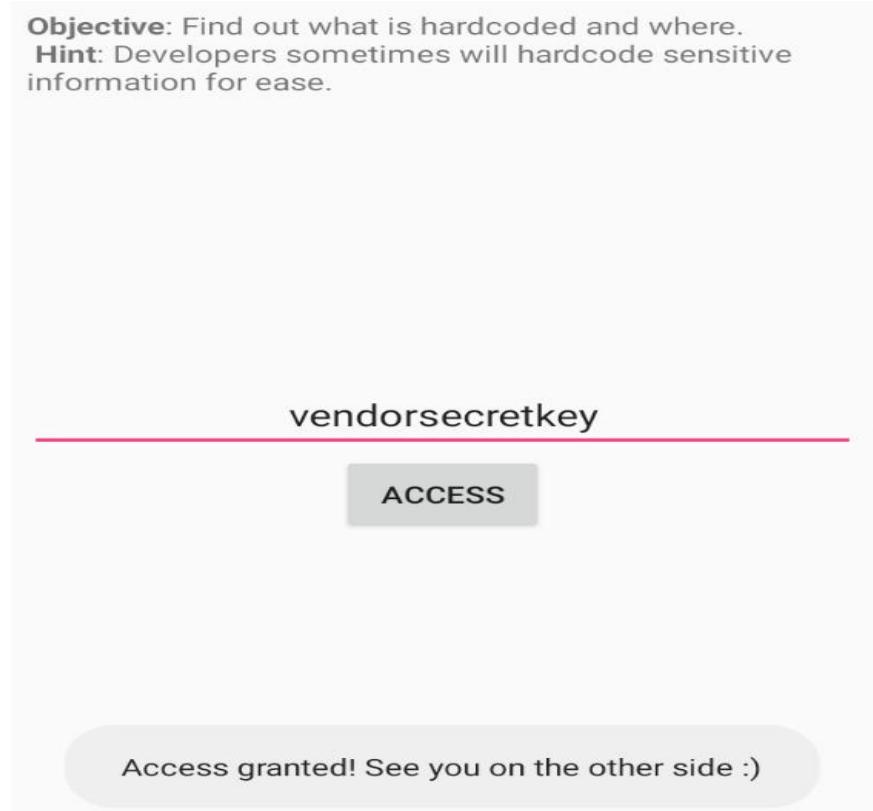
    public void access(View view) {
        EditText hckey = findViewById(R.id.hckey);
        if (hckey.getText().toString().equals("vendorsecretkey")) {
            Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
        } else {
            Toast.makeText(this, "Access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

Entering vendorsecretkey – Access Granted

After discovering the hardcoded key in the code, entering vendorsecretkey in the app grants access successfully. This confirms the vulnerability.

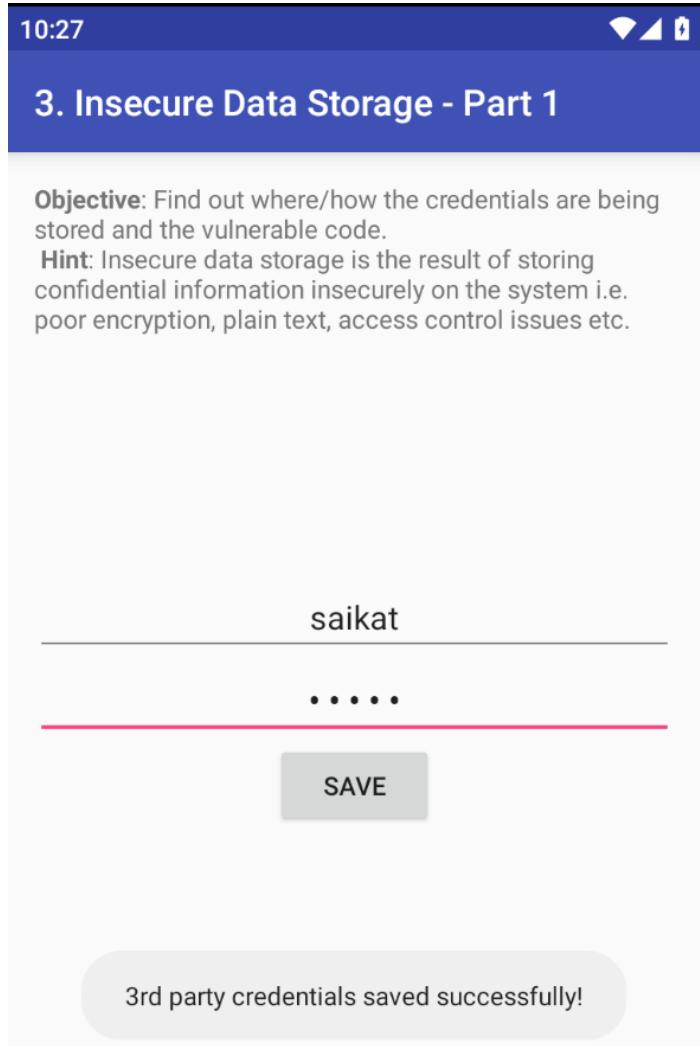
Objective: Find out what is hardcoded and where.

Hint: Developers sometimes will hardcode sensitive information for ease.



Insecure Data Storage – Part 1

In this challenge, we enter a username (saikat) and password (12345) in the DIVA app. The app shows a message: “*3rd party credentials saved successfully!*” indicating the credentials were stored on the device.



To locate where the app stores data, we first confirm the package name using:

Run: adb shell pm list packages | findstr diva

```
PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" shell pm list packages | findstr diva
package:jakhar.aseem.diva
```

Navigating to shared_prefs with ADB

Run one by one:

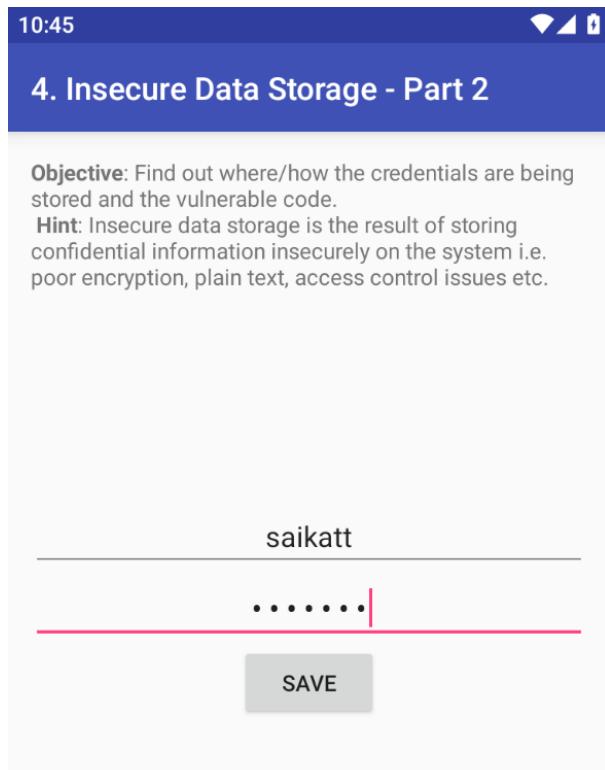
1. & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" shell
2. run-as jakhar.aseem.diva
3. cd /data/data/jakhar.aseem.diva/shared_prefs
4. ls -l
5. cat *.xml

```
PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" shell
vbox86p:/ # run-as jakhar.aseem.diva
vbox86p:/data/user/0/jakhar.aseem.diva $ cd /data/data/jakhar.aseem.diva/shared_prefs
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs $ ls -l
total 8
-rw-rw---- 1 u0_a129 u0_a129 153 2025-10-02 10:27 jakhar.aseem.diva_preferences.xml
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs $ cat *.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">12345</string>
    <string name="user">saikat</string>
</map>
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs $
```

This proves the credentials are saved in plain text. The app is vulnerable because it stores sensitive credentials in clear text. An attacker with device access could easily steal this data. The secure fix is to use encrypted storage or the Android Keystore system.

Insecure Data Storage – Part 2

In this challenge, we input username (saikatt) and password. The app confirms the credentials are saved. Our task is to check *where* and *how* they are stored on the device.



ADB shell – locating database storage (Run one by one)

1. & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" shell
2. run-as jakhar.aseem.diva
3. ls -la
4. cd databases
5. ls -la
6. sqlite3 ids2
7. .tables
8. Select * FROM myusers;

```

PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" shell
vbox86p:/ # run-as jakhar.aseem.diva
vbox86p:/data/user/0/jakhar.aseem.diva $ ls -la
total 56
drwxr-x--x 6 u0_a129 u0_a129 4096 2025-10-02 10:27 .
drwxrwx--x 167 system system 12288 2025-10-02 09:38 ..
drwxrws--x 2 u0_a129 u0_a129_cache 4096 2025-10-02 09:38 cache
drwxrws--x 2 u0_a129 u0_a129_cache 4096 2025-10-02 09:38 code_cache
drwxrwx--x 2 u0_a129 u0_a129 4096 2025-10-02 10:42 databases
drwxrwx--x 2 u0_a129 u0_a129 4096 2025-10-02 10:27 shared_prefs
vbox86p:/data/user/0/jakhar.aseem.diva $ cd databases
vbox86p:/data/user/0/jakhar.aseem.diva/databases $ ls -la
total 68
drwxrwx--x 2 u0_a129 u0_a129 4096 2025-10-02 10:42 .
drwxr-x--x 6 u0_a129 u0_a129 4096 2025-10-02 10:27 ..
-rw-rw---- 1 u0_a129 u0_a129 20480 2025-10-02 09:40 divanotes.db
-rw-rw---- 1 u0_a129 u0_a129 0 2025-10-02 09:40 divanotes.db-journal
-rw-rw---- 1 u0_a129 u0_a129 16384 2025-10-02 10:42 ids2
-rw-rw---- 1 u0_a129 u0_a129 0 2025-10-02 10:42 ids2-journal
vbox86p:/data/user/0/jakhar.aseem.diva/databases $ sqlite3 ids2
SQLite version 3.28.0 2020-05-06 18:46:38
Enter ".help" for usage hints.
sqlite> .tables
android_metadata myuser
sqlite> select * FROM myuser;
saikatt|1234555

```

By opening the database with SQLite (using commands like `sqlite3`), we can see that the username and password are stored in **plain text** inside the database.

Input Validation Issue Part- 1

The app takes the text you type and directly concatenates it into an SQL query. Because there is no sanitization or parameterization, a malicious input can change the meaning of the SQL statement and return all records instead of just the intended one. This is classic SQL injection.

Run: '1' or '1'='1'--

7. Input Validation Issues - Part 1

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.

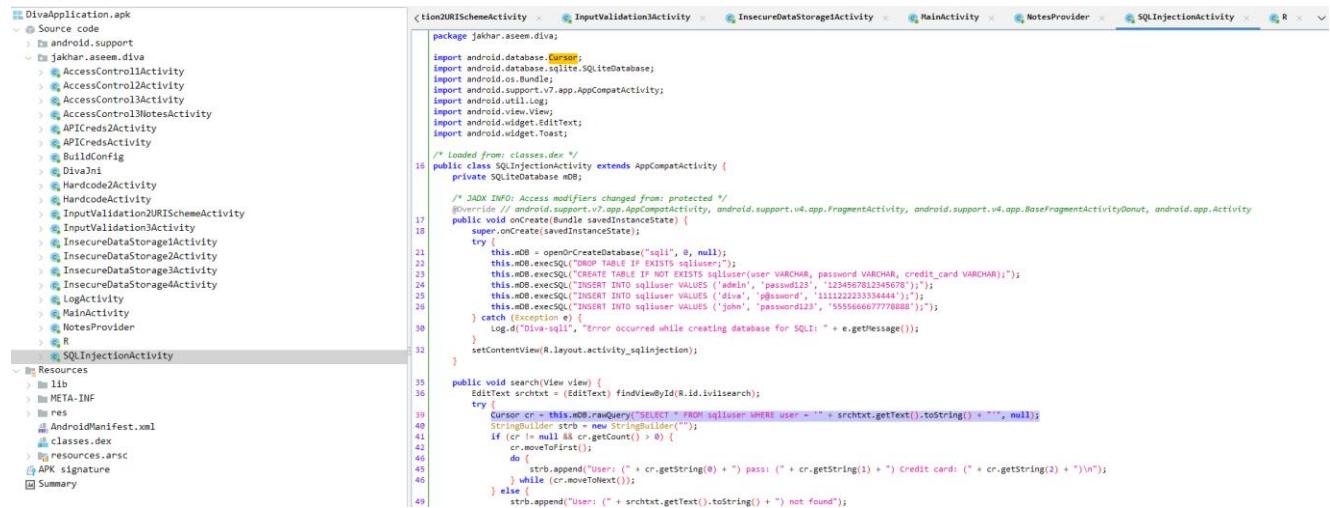
Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. For ease of testing there are three users already present in the database, for example one of them is admin, you can try searching for admin to test the output.

1'or'1'='1'--

SEARCH

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

Viewing vulnerable code in JADX



```
15 // Loaded from: classes.dex /*  
16 public class SQLInjectionActivity extends AppCompatActivity {  
17     private SQLiteDatabase mDB;  
18     /* JAD: INQ: Access modifiers changed from: protected */  
19     @Override // android.support.v4.app.FragmentActivity, android.support.v4.app.FragmentActivityOnDonut, android.app.Activity  
20     protected void onCreate(Bundle savedInstanceState) {  
21         super.onCreate(savedInstanceState);  
22         try {  
23             this.mDB = openOrCreateDatabase("sql1", 0, null);  
24             this.mDB.execSQL("DROP TABLE IF EXISTS sqluser");  
25             this.mDB.execSQL("CREATE TABLE IF NOT EXISTS sqluser(user VARCHAR, password VARCHAR, credit_card VARCHAR);");  
26             this.mDB.execSQL("INSERT INTO sqluser VALUES ('admin', 'password123', '1234567812345678');");  
27             this.mDB.execSQL("INSERT INTO sqluser VALUES ('diva', 'p@ssw0rd', '1111222233334444');");  
28             this.mDB.execSQL("INSERT INTO sqluser VALUES ('johh', 'password123', '5555666677778888');");  
29         } catch (Exception e) {  
30             Log.d("Diva-sql1", "Error occurred while creating database for SQLI " + e.getMessage());  
31         }  
32         setContentView(R.layout.activity_sqlinjection);  
33     }  
34     public void search(View view) {  
35         EditText srctxt = (EditText) findViewById(R.id.id_ivlsearch);  
36         try {  
37             Cursor cr = this.mDB.rawQuery("SELECT * FROM sqluser WHERE user = '" + srctxt.getText().toString() + "'", null);  
38             StringBuilder strb = new StringBuilder("");  
39             if (cr != null && cr.getCount() > 0) {  
40                 cr.moveToFirst();  
41                 do {  
42                     strb.append("User: (" + cr.getString(0) + ") pass: (" + cr.getString(1) + ") Credit card: (" + cr.getString(2) + ")\n");  
43                 } while (cr.moveToNext());  
44             } else {  
45                 strb.append("User: (" + srctxt.getText().toString() + ") not found");  
46             }  
47         } catch (Exception e) {  
48             Log.d("Diva-sql1", "Error occurred while searching for SQLI " + e.getMessage());  
49         }  
50     }  
51 }
```

In the source code (InputValidation1Activity.java), the following line shows the vulnerability: **Cursor cr = mDB.rawQuery("SELECT * FROM sqluser WHERE user = "" + srctxt.getText().toString() + "", null);**

The input from the user is directly concatenated into the SQL query without sanitization or prepared statements, making the app vulnerable to SQL Injection.

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.

prefs/jakhar.aseem.diva_preferences.xml

VIEW

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼ <map>  
  <string name="password">12345</string>  
  <string name="user">saikat</string>  
</map>
```

Access Control Issues – Part 1

The challenge states that API credentials can be viewed by clicking a button inside the app. The task is to try and access those credentials from outside the app, simulating how attackers exploit weak access controls.

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.

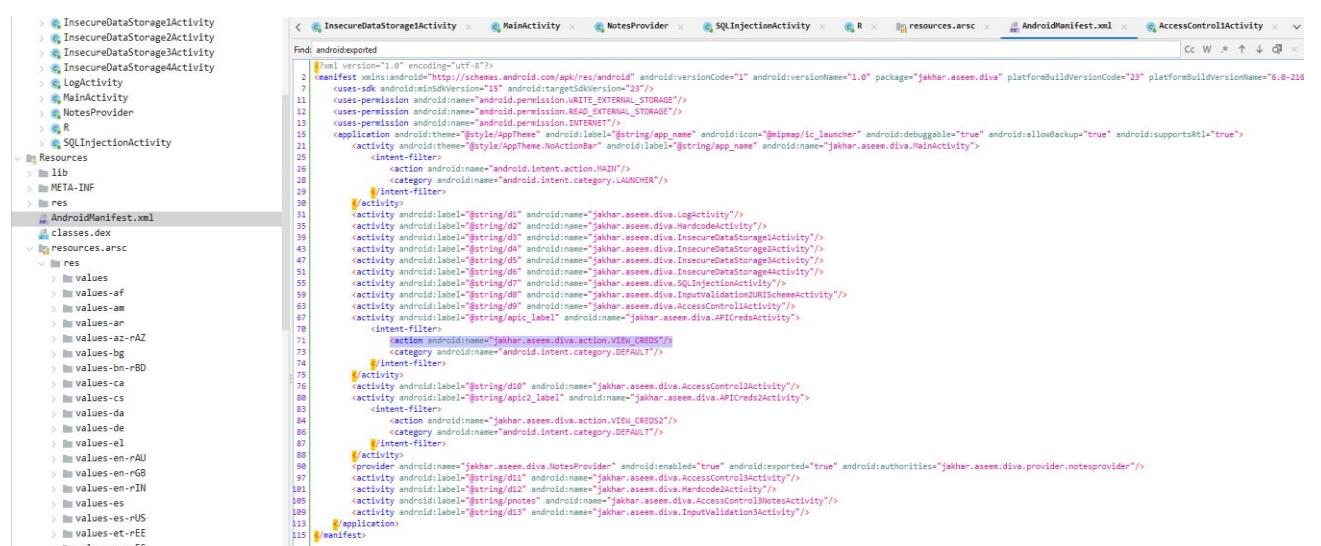
Hint: Components of an app can be accessed from other apps or users if they are not properly protected. Components such as activities, services, content providers are prone to this.

VIEW API CREDENTIALS

AndroidManifest.xml showing exported activities

Looking at the app's manifest file, we see that the AccessControl1Activity is marked as `exported = true`.

This means other apps or even ADB commands can start this activity without restriction, exposing sensitive functionality.



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="jakhar.aseem.diva" platformBuildVersionCode="23" platformBuildVersionName="6.0-216">
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <application android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:debuggable="true" android:allowBackup="true" android:supportsRtl="true">
        <activity android:name=".MainActivity" android:label="@string/app_name" android:theme="@style/AppTheme.NoActionBar" android:label="@string/app_name" android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        <activity android:label="@string/d1" android:name=".LogActivity"/>
        <activity android:label="@string/d2" android:name=".InsecureDataStorage3Activity"/>
        <activity android:label="@string/d3" android:name=".InsecureDataStorage4Activity"/>
        <activity android:label="@string/d4" android:name=".NotesProvider"/>
        <activity android:label="@string/d5" android:name=".R"/>
        <activity android:label="@string/d6" android:name=".SQLInjectionActivity"/>
        <activity android:label="@string/d7" android:name=".AccessControl1Activity" android:exported="true" android:label="@string/d7" android:label="jakhar.aseem.diva.AccessControl1Activity">
            <intent-filter>
                <action android:name="jakhar.aseem.diva.action.VIEW_CRED05"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        <activity android:label="@string/d8" android:name=".AccessControl2Activity"/>
        <activity android:label="@string/api1_label" android:name=".APIcreds2Activity">
            <intent-filter>
                <action android:name="jakhar.aseem.diva.action.VIEW_CRED52"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        <activity android:name="jakhar.aseem.diva.NotesProvider" android:enabled="true" android:exported="true" android:authorities="jakhar.aseem.diva.provider.notesprovider"/>
        <provider android:name="jakhar.aseem.diva.NotesProvider" android:enabled="true" android:exported="true" android:authorities="jakhar.aseem.diva.provider.notesprovider"/>
        <activity android:label="@string/d11" android:name="jakhar.aseem.diva.AccessControl3Activity"/>
        <activity android:label="@string/d12" android:name="jakhar.aseem.diva.HandcodeActivity"/>
        <activity android:label="@string/pronet" android:name="jakhar.aseem.diva.AccessControl3NotesActivity"/>
        <activity android:label="@string/d13" android:name="jakhar.aseem.diva.InputValidation3Activity"/>
    </application>
</manifest>
```

Decompiled code of AccessControl1Activity

In the source code, the activity listens for an intent action:

`jakhar.aseem.diva.action.VIEW_CREDS`

When this intent is triggered, the app directly displays API credentials on the screen. Since there are no permission checks, any external app or user can call this intent.



```
/*
 * This file was generated by the Java(TM) Persistence API tools.
 */
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;

public class AccessControl1Activity extends AppCompatActivity {
    /* Loaded from class file */
    /* JADK INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityDonut, android.app.Activity
    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 (getApplicationContext().getPackageManager() != null) {
            startActivity(i);
            return;
        }
        Toast.makeText(this, "Error while getting API details", 0).show();
        Log.e("Diva-act1", "Couldn't resolve the Intent VIEW_CREDS to our activity");
    }
}
```

Using ADB to trigger the activity externally

Using the following ADB command:

Run : adb shell am start -a jakhar.aseem.diva.action.VIEW_CREDS

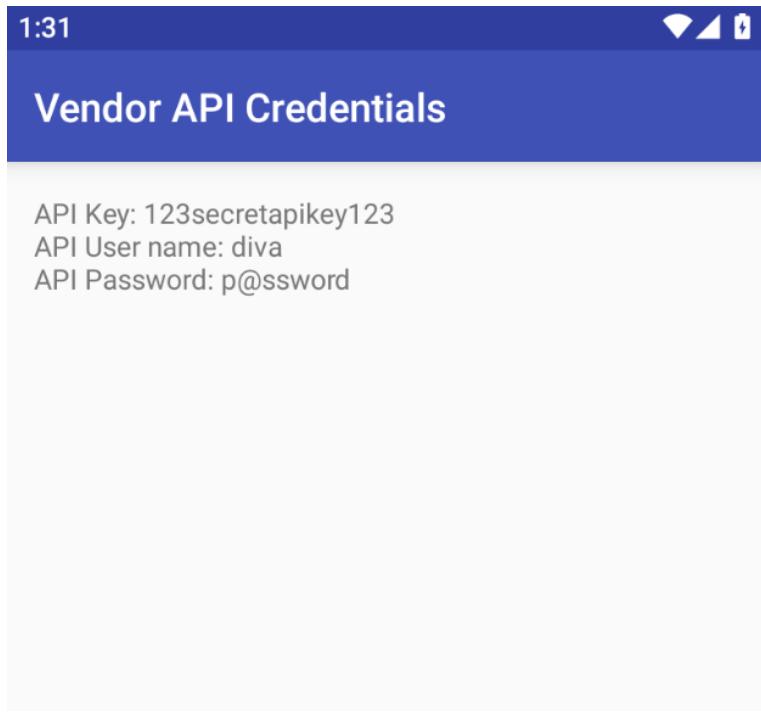
```
PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" shell am start -a jakhar.aseem.diva.action.VIEW_CREDS
Starting: Intent { act=jakhar.aseem.diva.action.VIEW_CREDS }
PS C:\WINDOWS\system32>
```

API credentials displayed

As a result, the application displays sensitive vendor API credentials:

- API Key: 123secretapikey123
- Username: diva
- Password: p@ssword

This confirms that the activity is insecurely exposed.



Access Control Issues – Part 2

This challenge is about accessing TVEETER API credentials. Normally, users must register and provide a PIN to view them. The goal is to bypass the PIN check and access the credentials from outside the app.

A screenshot of a mobile application section titled "10. Access Control Issues - Part 2". The title is displayed in white text on a dark blue header bar. The main content area is white and contains the following text:

Objective: You are able to access the Third Party app TVEETER API credentials after you have registered with Tveeter. The App requests you to register online and the vendor gives you a pin, which you can use to register with the app. Now, try to access the API credentials from outside the app without knowing the PIN. This is a business logic problem so you may need to see the code.

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.

Register Now. Already Registered.

VIEW TVEETER API CREDENTIALS

Viewing AccessControl2Activity code

The code for AccessControl2Activity shows that it expects an intent with a parameter called check_pin. If check_pin is false, the activity still proceeds to display the API credentials.

This is a business logic flaw — the app trusts the external intent parameter without verification.



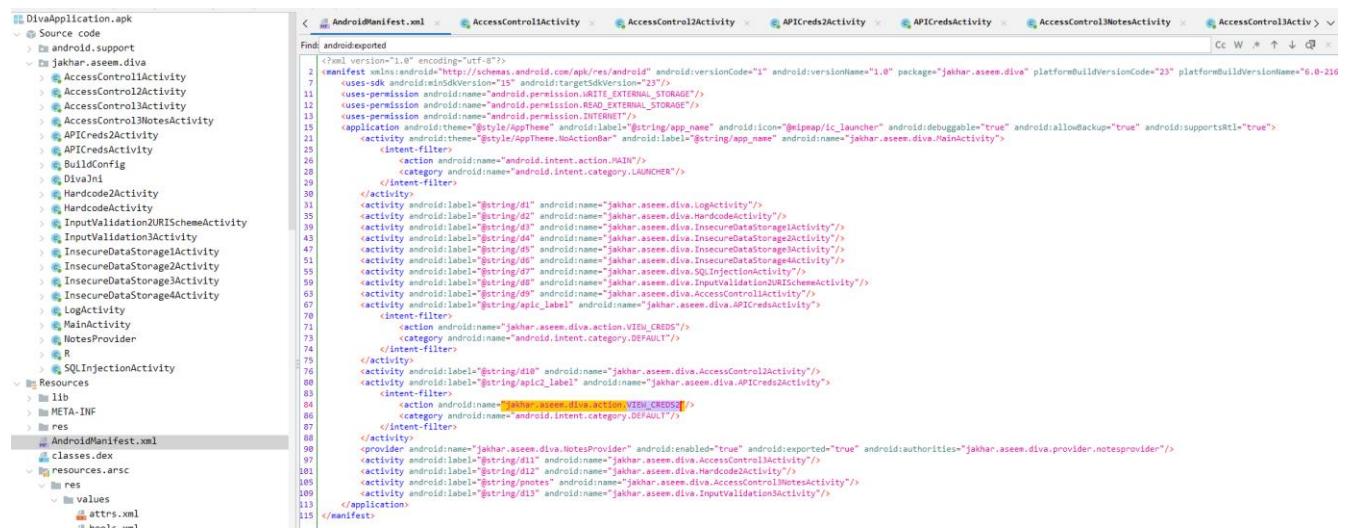
```
/* Loaded from: classes.dex */
public class AccessControl2Activity extends AppCompatActivity {
    /* JAKHAR_APP_ID_ACCESS modifers are taken from protected */
    /* package android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityOnout, android.app.Activity */
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_access_control);
    }

    public void viewACredentials(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("check_pin", chk_pin);
        i.setCategory("jakhar.aseem.diva.access_control");
        if (i.resolveActivity(getApplicationContext()) != null) {
            startActivity(i);
            return;
        }
        Toast.makeText(this, "Error while getting Tweeter API details", 0).show();
        Log.e("Div-a-cti", "Couldn't resolve the Intent VIEW_CREDS2 to our activity");
    }
}
```

AndroidManifest.xml showing exported activity

In the manifest file, AccessControl2Activity is marked with an intent-filter that listens for the action: **jakhar.aseem.diva.action.VIEW_CREDS2**

Since it is exported and doesn't require permissions, any external app or ADB command can trigger it.



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="jakhar.aseem.diva" platformBuildVersionCode="23" platformBuildVersionName="6.0-216">
    <uses-sdk android:minSdkVersion="15" android:targetSdkVersion="23" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:debuggable="true" android:allowBackup="true" android:supportsRtl="true">
        <activity android:theme="@style/AppTheme.NoActionBar" android:label="@string/app_name" android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:label="@string/d1" android:name=".LogActivity" />
        <activity android:label="@string/d2" android:name=".AccessControl2Activity" />
        <activity android:label="@string/d3" android:name=".AccessControl3Activity" />
        <activity android:label="@string/d4" android:name=".APIcredsActivity" />
        <activity android:label="@string/d5" android:name=".APIcreds2Activity" />
        <activity android:label="@string/d6" android:name=".AccessControlNotesActivity" />
        <activity android:label="@string/d7" android:name=".AccessControl3NotesActivity" />
        <activity android:label="@string/d8" android:name=".AccessControl2Activity" />
        <activity android:label="@string/api2_label" android:name=".APIcredsActivity" />
        <intent-filter>
            <action android:name="jakhar.aseem.diva.action.VIEW_CREDS2" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
        <activity android:label="@string/d9" android:name=".AccessControl3Activity" />
        <activity android:label="@string/d10" android:name=".APIcreds2Activity" />
        <intent-filter>
            <action android:name="jakhar.aseem.diva.action.VIEW_CREDS2" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
        <activity android:label="@string/d11" android:name=".NotesProvider" android:exported="true" android:enabled="true" android:authorities="jakhar.aseem.diva.provider.notesprovider" />
        <activity android:label="@string/d12" android:name=".AccessControl2Activity" />
        <activity android:label="@string/d13" android:name=".AccessControl3Activity" />
        <activity android:label="@string/pnotes" android:name=".AccessControlNotesActivity" />
        <activity android:label="@string/d15" android:name=".InputValidationActivity" />
    </application>
</manifest>
```

Using ADB to exploit the activity

```
PS C:\WINDOWS\system32> & "C:\Program Files\Genymobile\Genymotion\tools\adb.exe" shell am start -a jakhar.aseem.diva.action.VIEW_CREDS2 --ez check_pin false
Starting: Intent { act=jakhar.aseem.diva.action.VIEW_CREDS2 (has extras) }
```

We send the following ADB command to directly call the activity and pass a malicious parameter:

Run: adb shell am start -a jakhar.aseem.diva.action.VIEW_CREDS2 --ez check_pin false

The --ez check_pin false argument tricks the app into believing registration was already done, so it displays the credentials.

TVEETER API credentials displayed

The app exposes sensitive third-party credentials without requiring the actual PIN:

- TVEETER API Key: secrettveeterapikey
- Username: diva2
- Password: p@ssword2

This proves the access control weakness.

