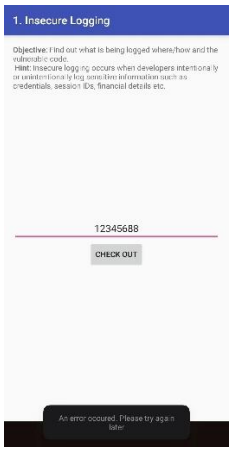
I. Insecure Logging Lab

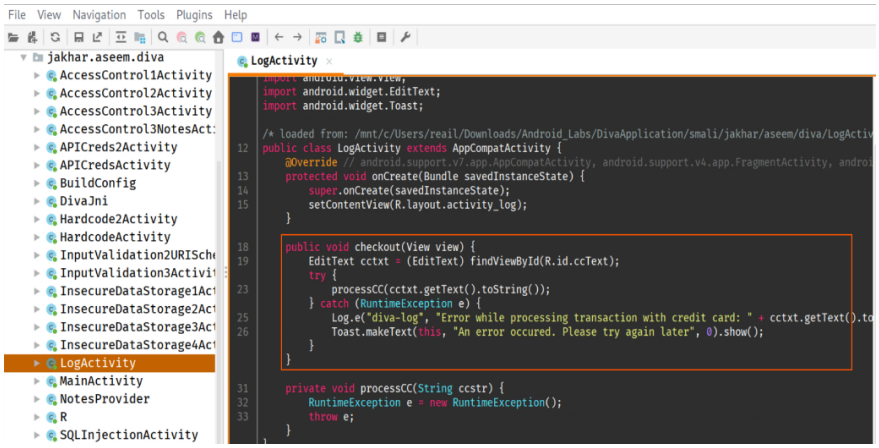
For to Understand this lab, we require two modules as pre-requisite first ADB (helps connect and debug android devices), and platform tools (provides necessary tools for android SDK)

Objective: Identifying insecure logging practices within the application.

Steps:

1. **Extract Application APK:** Using apktool, extract the contents of DivaApplication.apk into a new directory named DivaApplication. This allows for further analysis without needing the original .apk file.

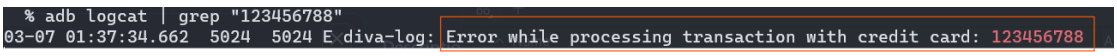




2. **Identifying via ADB Logs:** Use the following command in Git Bash to capture logs from the emulator:

   ```
   adb logcat | grep "123456788"
   ```

   This command searches for lines containing “123456788” in logcat output, which may indicate exposed sensitive information.



3. **Verify and Analyze Results:** After executing the above command, check if the string is found in the logs. If it is there then, it confirms insecure logging practices.

4. **Conclusion:** Secure logging practices involve encrypting sensitive information in logs and avoiding the exposure of Credit Card details, API keys and other credentials.

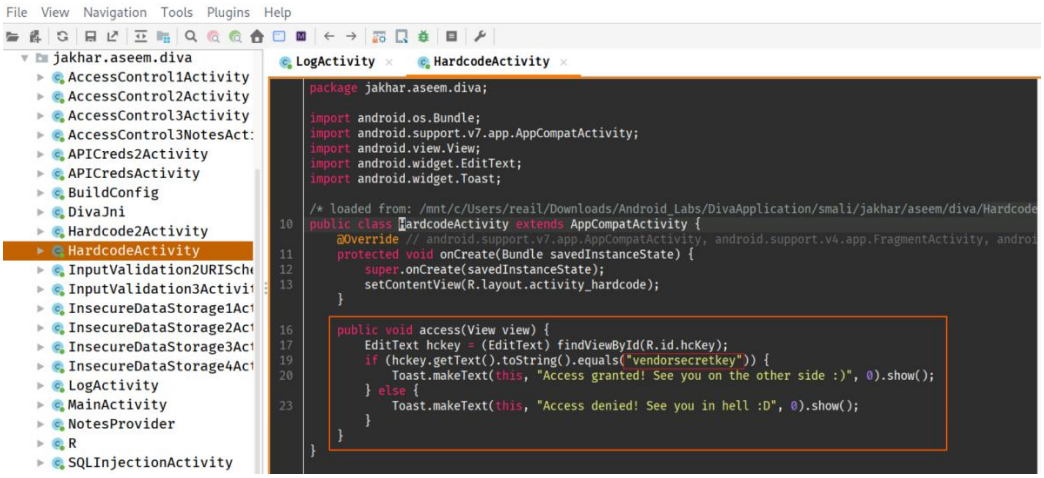
II. Hardcoding Issues Lab

Objective: Identify hardcoded credentials within the decompiled application.

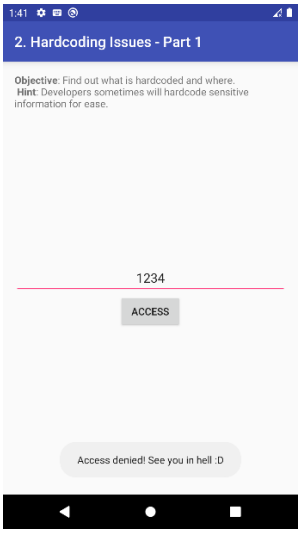
1. **Extracted Application APK:** Repeat step 1 from Lab 1 for decompiling apk.
2. **Identify Hardcoded Values Using Jadx-gui:** Use the following command in kali wsl:

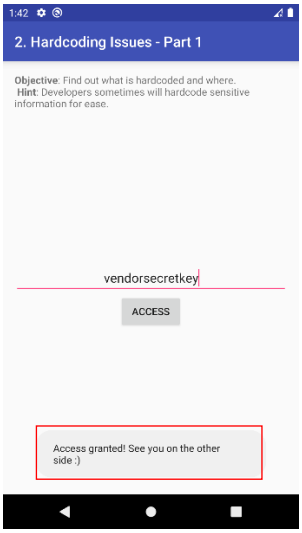
   sudo jadx-gui

   This command opens a GUI where we can **open the decompiled folder** and look for hardcoded lines starting with any key.



3. **Verify and Analyze Results:** Check for specific hardcoded strings or patterns that might indicate security risks, and here it is ***"vendorsecretkey"***.



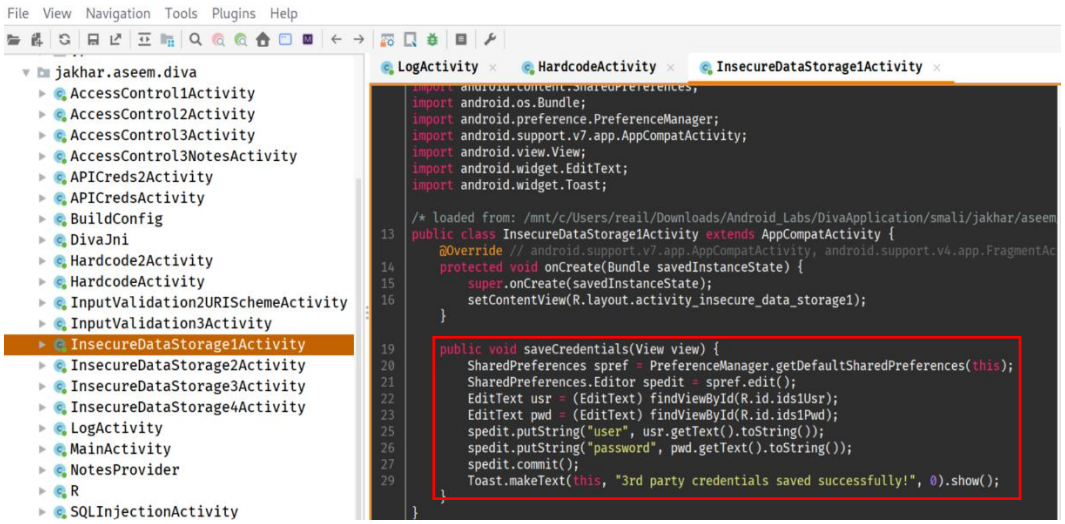


4. **Conclusion:** Hardcoded credentials can be easily extracted by attackers, necessitating the use of secure storage mechanisms like SharedPreferences with proper encryption.

III. Insecure Storage – I

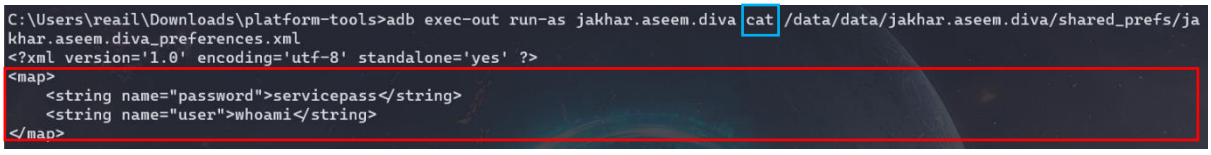
Objective: Identify insecure plain-text data stored within the application.

1. **Extract Application APK:** Repeat step 1 for this lab to ensure consistent analysis.

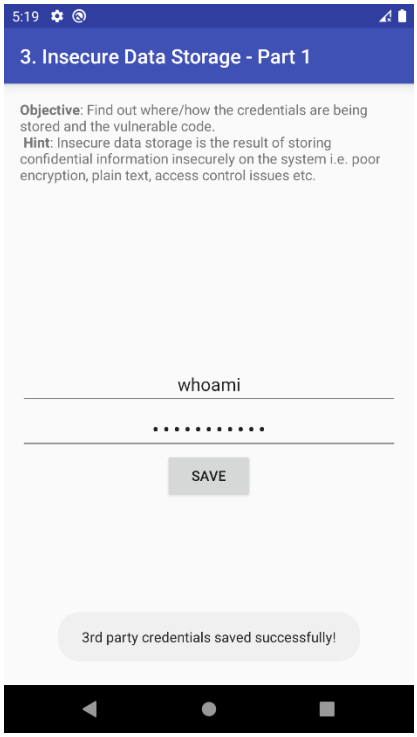


2. **Identify Sensitive Data in ADB Shell:** Use Git Bash with these following commands:

Again, open the GUI into **the decompiled folder** and look specific patterns related to sensitive data storage (e.g., SharedPreferences, internal storage). And ls inside **shared_prefs**.



3. **Verify and Analyze Results:** We can clearly see that **SharedPreferences** is used to store sensitive credentials that is user and password.



We navigated to the folder that is ***data/data*** and ***ls*** to see all the files inside that directory.

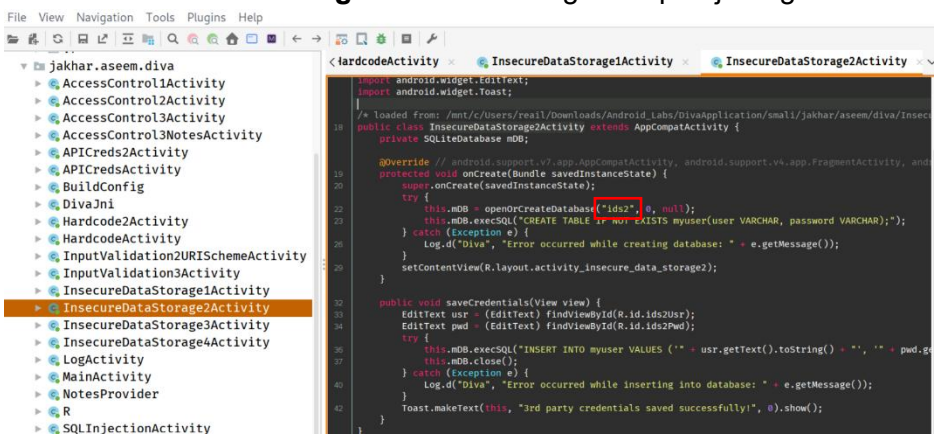
Next we will navigate to the folder ***jakhar.aseem.diva***, we go further inside that folder and ***ls*** again we will find **shared_prefs**, inside this the sensitive data is stored in XML format, which we then **cat**.

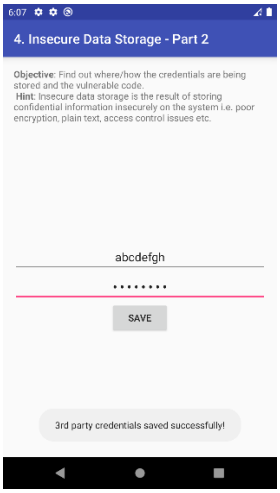
4. **Conclusion:** Storing credentials in XML without proper file permissions and secure storage mechanisms which can prevent unauthorized access or modification of sensitive data considered as an insecure way of storing data.

IV. Insecure Storage – II

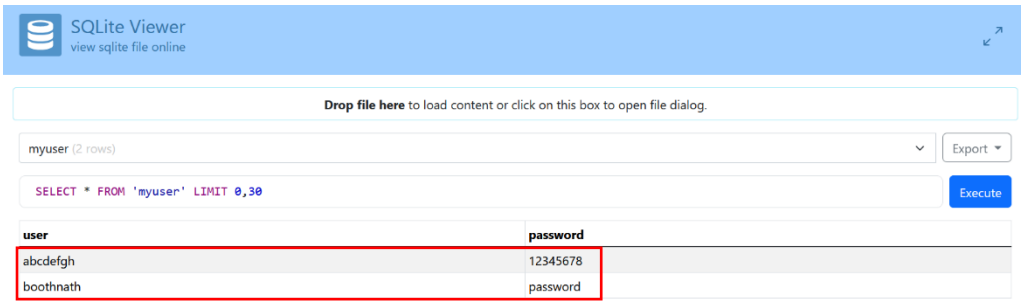
Objective: Identify insecure SQL data storage within the application database.

1. **Extract Application APK:** Repeat step 1 again for this lab for analysis.
2. **Identify Sensitive Table in Jadx-gui:** Use kali kex gui to open jadx-gui in **sudo**:






```
C:\Users\reail\Downloads\platform-tools>adb exec-out run-as jakhar.aseem.diva cp /data/data/jakhar.aseem.diva/databases/ids2
 /sdcard/ids2

C:\Users\reail\Downloads\platform-tools>adb pull /sdcard/ids2
/sdcard/ids2: 1 file pulled, 0 skipped. 9.3 MB/s (16384 bytes in 0.002s)
```

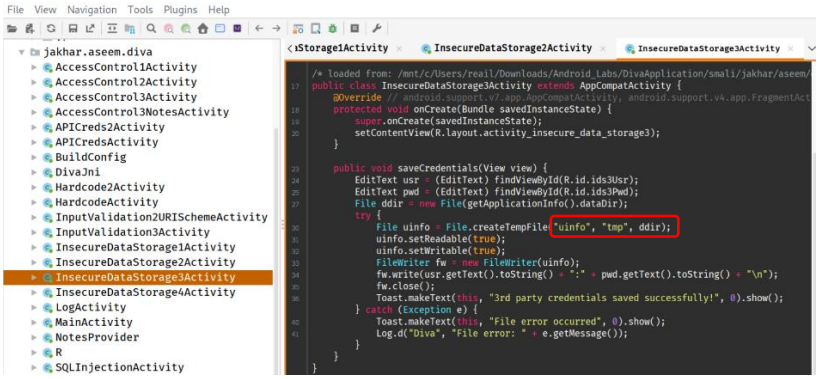


3. **Verify and Analyze Results:** We can clearly see credentials inside SQLite Viewer, and SQLite is used to store the user and password.
4. **Conclusion:** Storing credentials in SQLite without proper file permissions and secure storage mechanisms can lead to exposure or modification of sensitive data.

V. Insecure Storage – III

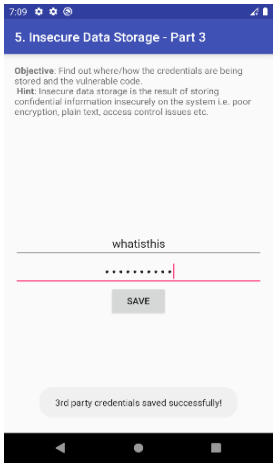
Objective: Identify insecure temp file data storage within the application.

1. **Extract Application APK:** Repeat step 1 again for this lab for analysis.



2. **Identify Sensitive tempfile in package dir:** Use kali kex gui to open jadx-gui in **sudo**:

   sudo jadx-gui -> open -> ~/path_to/DivaApplication (folder)




```
C:\Users\reail\Downloads\platform-tools>adb exec-out run-as jakhar.aseem.diva ls -al /data/data/jakhar.aseem.diva/
total 68
drwxr-x--x   6 u0_a134 u0_a134          4096 2025-03-07 07:09 .
drwxrwx--x 168 system  system          8192 2025-03-07 01:34 ..
drwxrws--x   2 u0_a134 u0_a134_cache 4096 2025-03-07 01:34 cache
drwxrws--x   2 u0_a134 u0_a134_cache 4096 2025-03-07 01:34 code_cache
drwxrwx--x   2 u0_a134 u0_a134          4096 2025-03-07 05:20 databases
lrwxrwxrwx   1 root    root            62 2025-03-07 05:16 lib -> /data/app/jakhar.aseem.diva-KL3-cpzgm7_u5y0En8fNgA==/lib/x
86
drwxrwx--x   2 u0_a134 u0_a134          4096 2025-03-07 05:19 shared_prefs
-rw-------   1 u0_a134 u0_a134            22 2025-03-07 07:09 uinfo8499528073792599641tmp
```

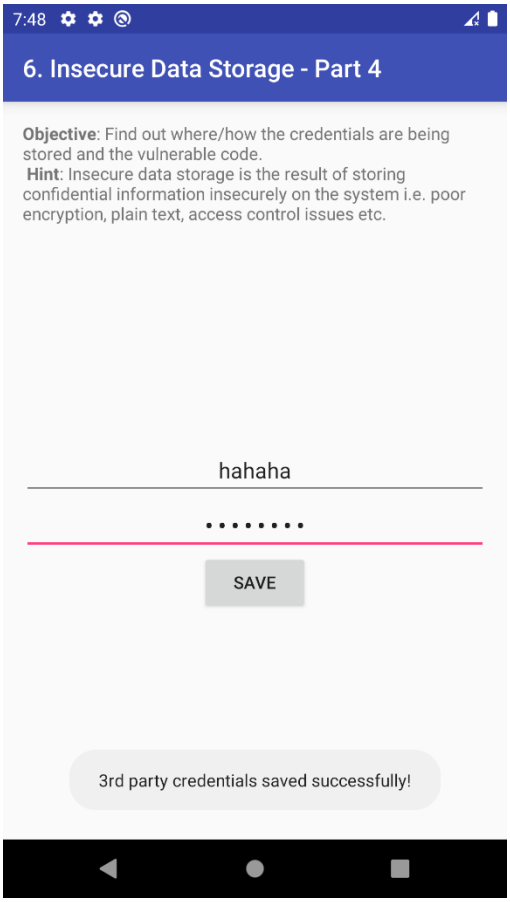
```
C:\Users\reail\Downloads\platform-tools>adb exec-out run-as jakhar.aseem.diva cat /data/data/jakhar.aseem.diva/uinfo8499528073792599641tmp
whatisthis:1234567890
```

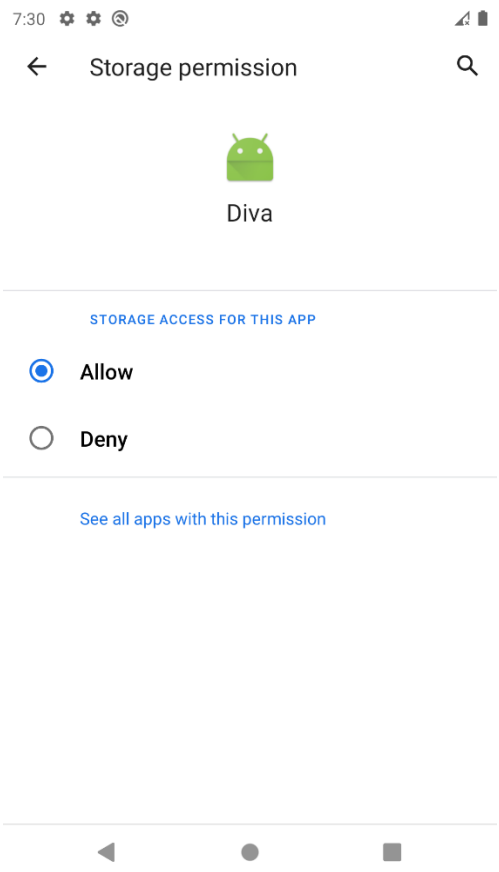
3. **Verify and Analyze Results:** We can clearly see credentials stored inside uinfo_xxxxtmp.
4. **Conclusion:** Storing credentials in a plain format without any encryption mechanisms can lead to stealing of sensitive data by perpetrators like script kiddies and those credentials can be seen by everyone because it was exposed in decompiled files in ***Jadx-gui***.

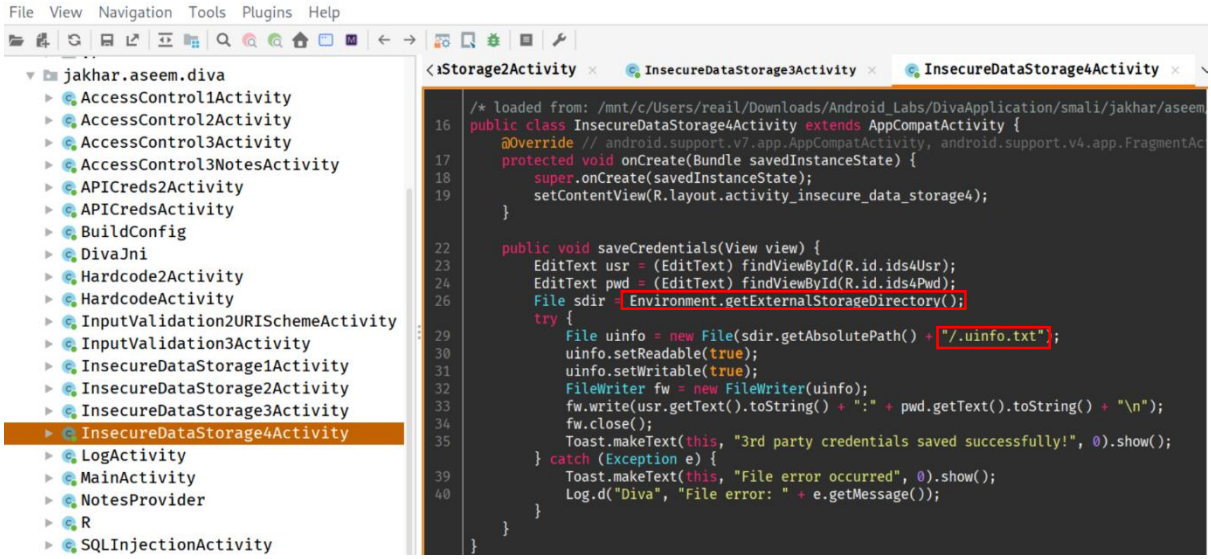
VI. Insecure Storage – IV

Objective: Identify insecure external sdcard data storage within the application.

1. **Extract Application APK:** Repeat the step 1 again for this lab too.
2. **Identify Exposed Sensitive External Storage in Jadx-gui:** Use same command with sudo like before, and search for external storage as shown below.
3. **Verify and Analyze Results:** We can clearly see that the user credentials are being stored in .uinfo.txt and because it a hidden file we need to do ***ls -la*** to see the file listed inside the folder. When we open that file we will find all credentials are stored inside.







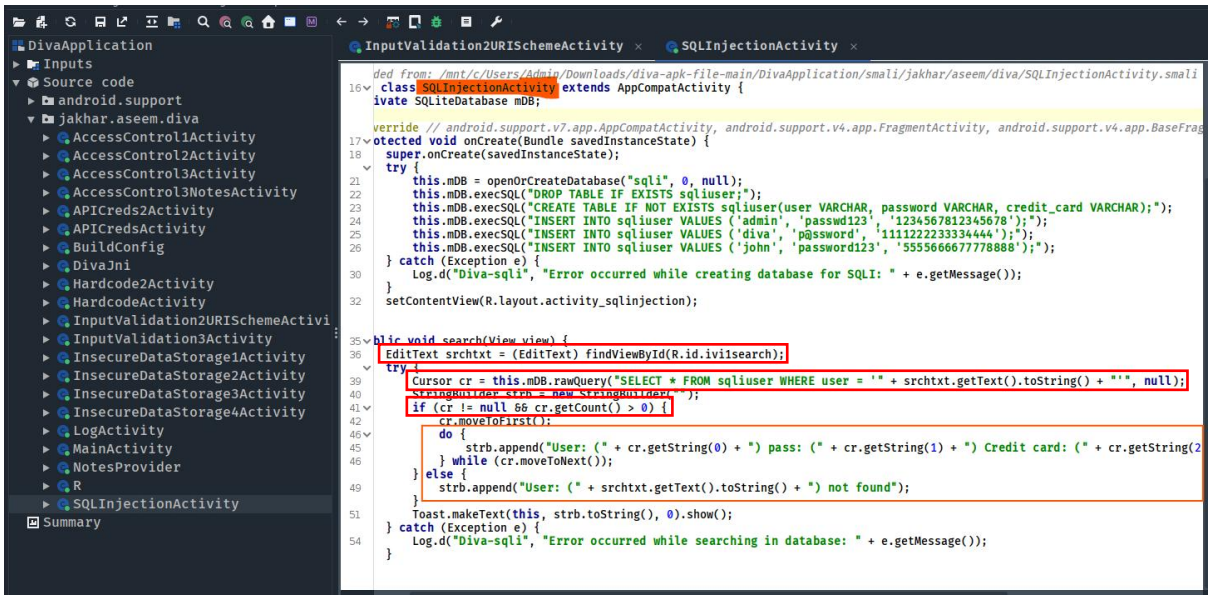

```
C:\Users\reail\Downloads\platform-tools>adb exec-out run-as jakhar.aseem.diva ls -a /mnt/sdcard
.  .uinfo.txt Android Download Music        Pictures Ringtones
..  Alarms     DCIM    Movies   Notifications Podcasts ids2

C:\Users\reail\Downloads\platform-tools>adb exec-out run-as jakhar.aseem.diva cat /mnt/sdcard/.uinfo.txt
hahaha:huhuhuhu
```

4. **Conclusion:** Storing credentials in a plain format without any encryption mechanisms can lead to stealing of sensitive data by perpetrators like script kiddies and those credentials can be seen by everyone because it was exposed in decompiled files in ***Jadx-gui***, which is similar to previous lab.

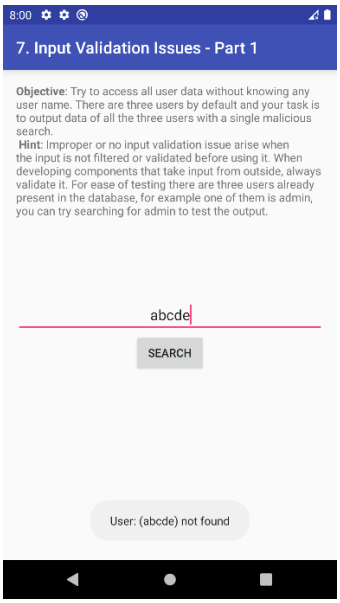
VII. Input Validation Issues – I

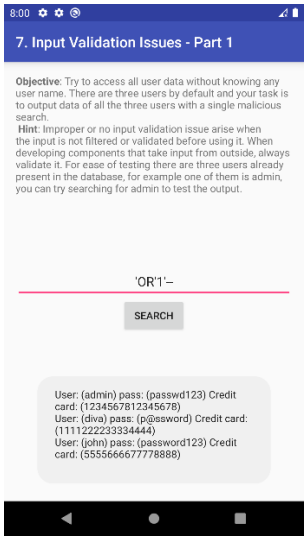
Objective: Identify insecure SQL query used for data storage within the application.

1. **Extract Application APK:** Repeat the step 1 again for this lab too.
2. **Identify Exposed SQL query in Jadx-gui:** Use same command with sudo like before, and search for query and the insecure if-statement as shown below.



3. **Verify and Analyze Results:** We can clearly see that the user credentials are being stored in without any proper validation. So, when we can try putting some strings first. Then, SQLi.



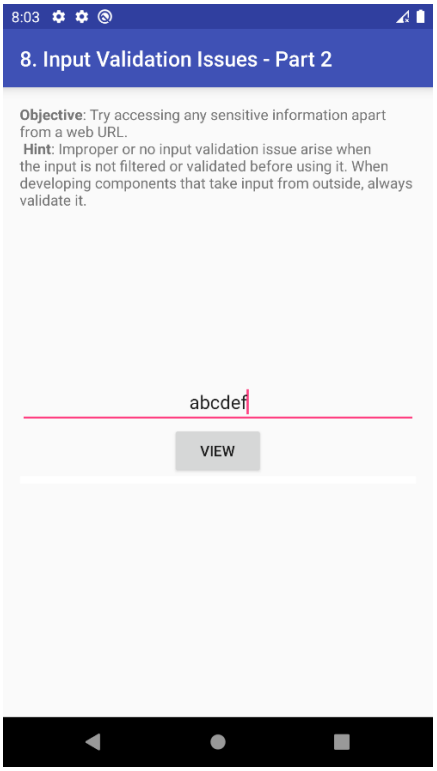


4. **Conclusion:** Storing credentials in a plain format without any encryption mechanisms can be very risky and Input should be sanitised and encoded properly to prevent attacks like SQL injection because it helps protecting sensitive data and authenticate data access.

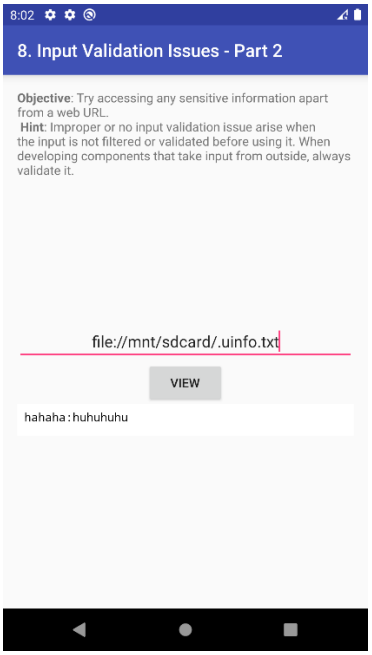
VIII. Input Validation Issues – II

Objective: Identify insecure file access within the application.

1. **Extract Application APK:** Repeat the step 1 again for this lab too.
2. **Identify the file path in Jadx-gui:** Use same command with sudo like before, and search.



3. **Verify and Analyze Results:** We can clearly see the user credentials that are being stored in .uinfo.txt, when we view that file path from previous labs.



4. **Conclusion:** Storing credentials in a plain format without any encryption mechanisms can lead to stealing of sensitive data by perpetrators like script kiddies and we should not allow users to view internal sensitive files which can cause severe damage to organisation's reputation.