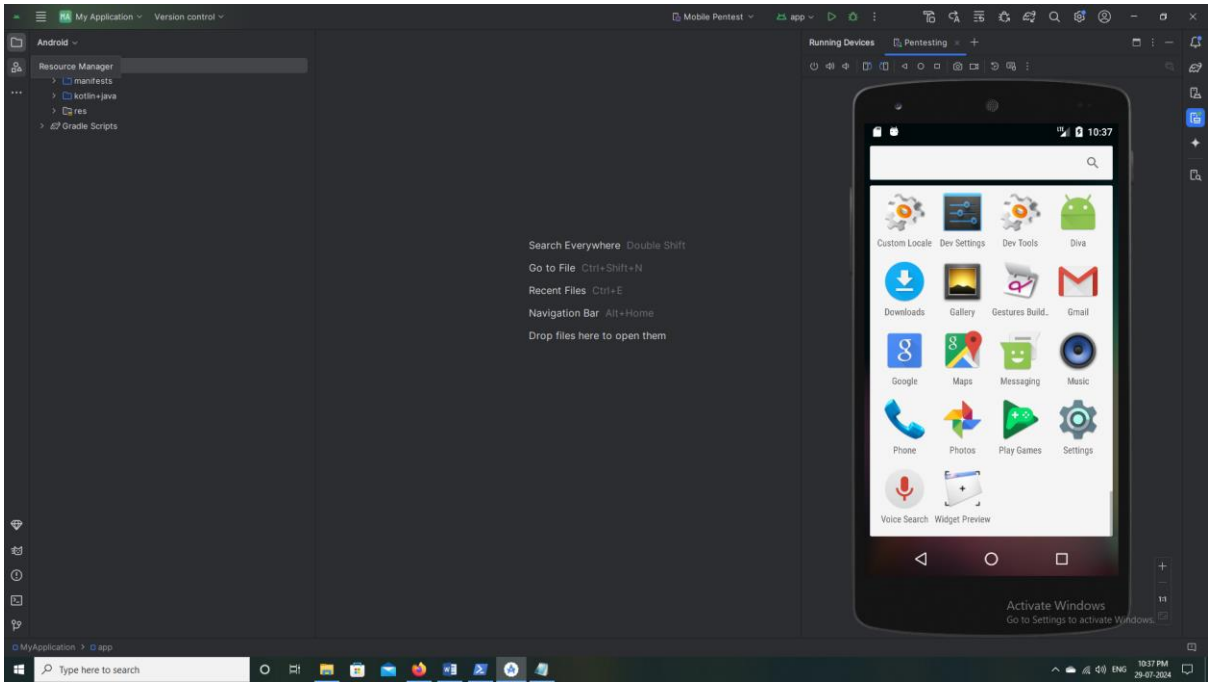
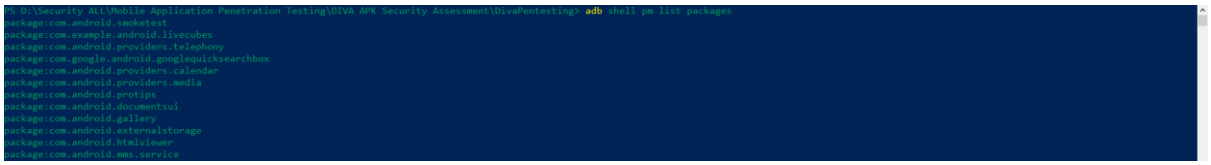
Diva Apk Mobile Pentest - Static Analysis

Introduction

The interface of Android Studio and the Emulator. Install DIVA application from playstore.

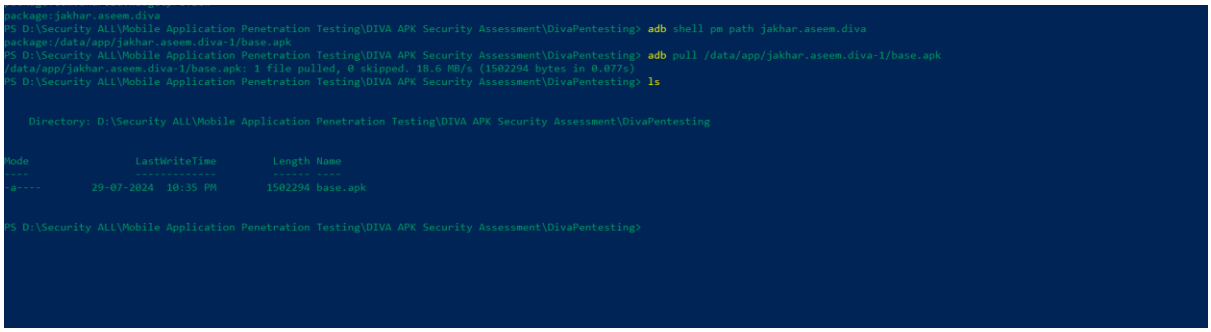


To Prints all packages - adb shell pm list packages



Print the path of the.apk of the given installed package name - adb shell pm path

To copy a file or directory and its sub-directories from the Android device - adb pull

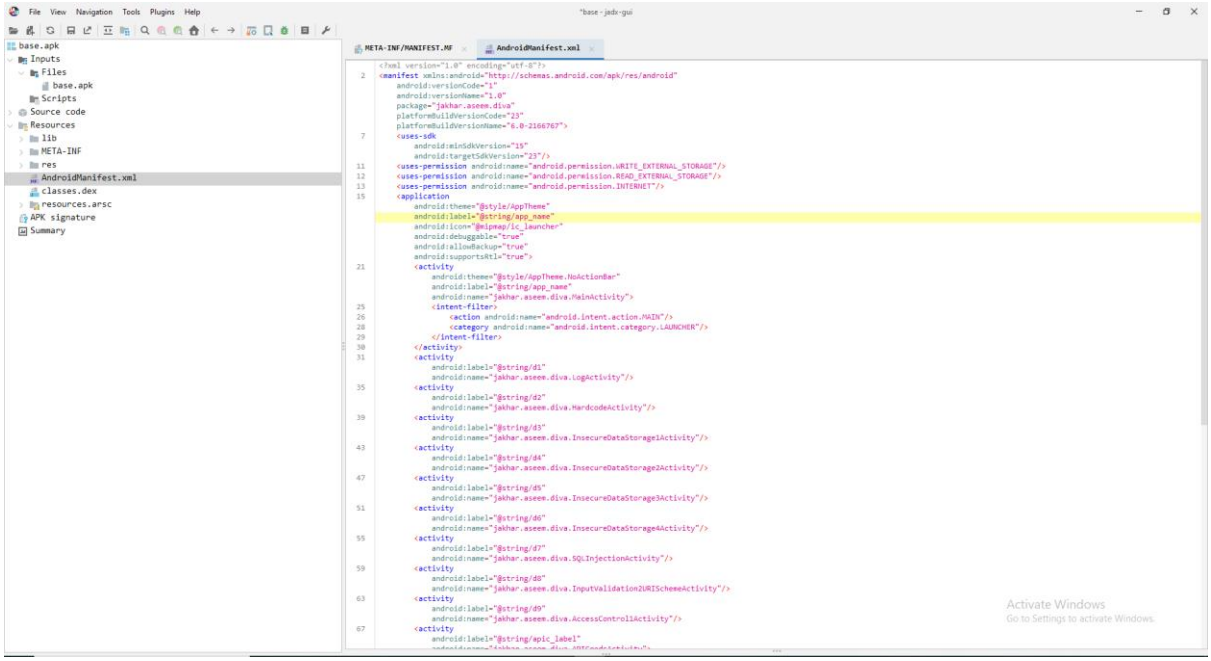


Base APK: It is a file format that Android uses to distribute and install apps. It is the original, unmodified version of an app that contains all the elements needed for installation. You can use Base APK to customize your apps, backup your data, or access apps that are not available on Google Play.

Now open the base.apk with jadx.

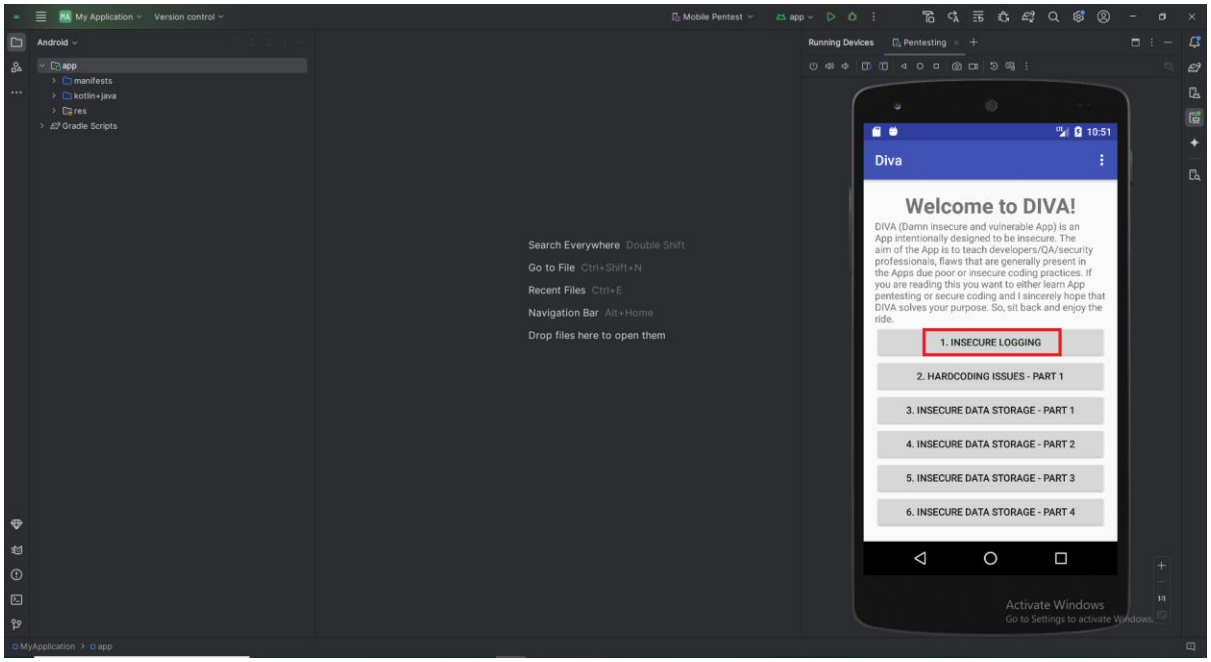
JADX: Users can analyse and comprehend the source code structure of an Android application.

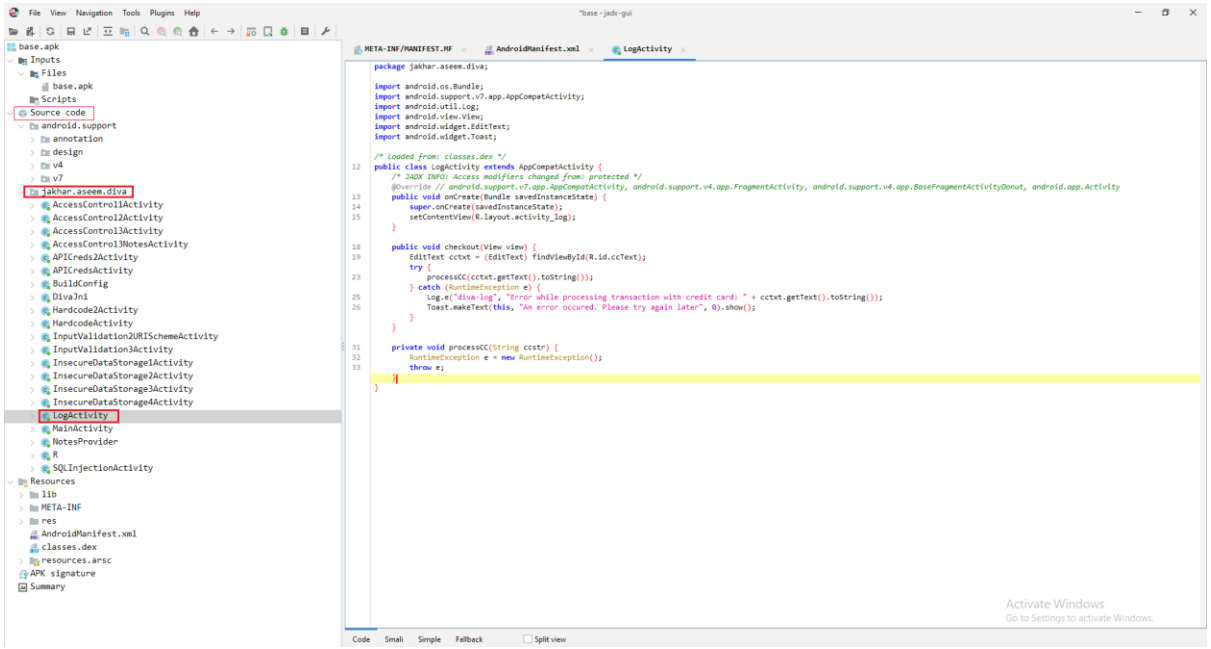
- decompile Dalvik bytecode or DEX code to Java source code from APK, dex, aar, aab and zip files.
- decode `AndroidManifest.xml` and other resources from `resources.arsc`



1. Insecure Logging

Screenshot-1: The activity **LogActivity** and its source code.



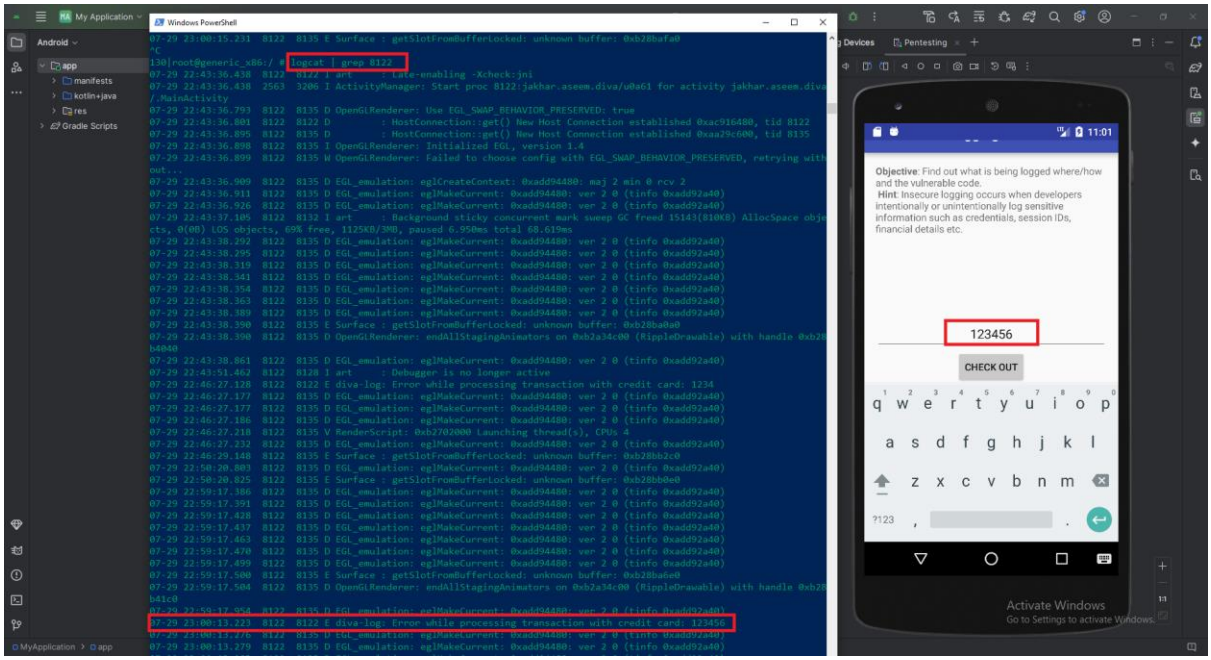


Screenshot-2: List processes - adb shell ps

```
PS D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting> adb shell ps
USER      PID   PPID  VSIZE  RSS     WCHAN              PC  NAME
root      1     0     2964   1064    SyS_epoll_ 081249c5 S /init
root      2     0     0      0       kthreadd 00000000 S kthreadd
root      3     2     0      0       smpboot_th 00000000 S ksoftirqd/0
root      5     2     0      0       worker_thr 00000000 S kworker/0:0H
root      6     2     0      0       worker_thr 00000000 S kworker/u8:0
root      7     2     0      0       smpboot_th 00000000 S migration/0
root      8     2     0      0       rcu_gp_kth 00000000 S rcu_preempt
```

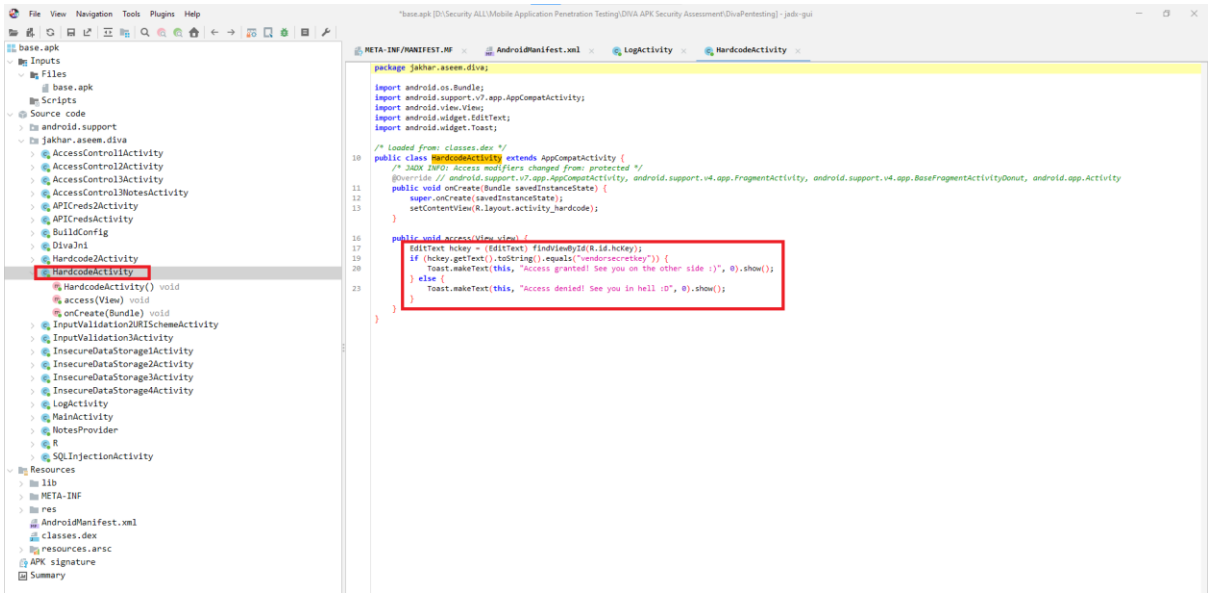
```
root      7906  2     0      0       worker_thr 00000000 S kworker/2:1
root      8034  1     21040  548     poll_sched 0808e495 S /sbin/adbd
root      8056  8034  3868   1204    hrtimer_na b7617e01 S /data/local/tmp/.studio/process-tracker
u0_a61    8122  2299  1561180 46004 SyS_epoll_ b72e1685 S jakhar.aseem.diva
root      8157  2     0      0       worker_thr 00000000 S kworker/1:0
root      8164  8034  3980   1108            0 b754b7d6 R ps
```

Screenshot-3: The Logcat window in Android Studio helps you debug your app by displaying logs from your device in real time.

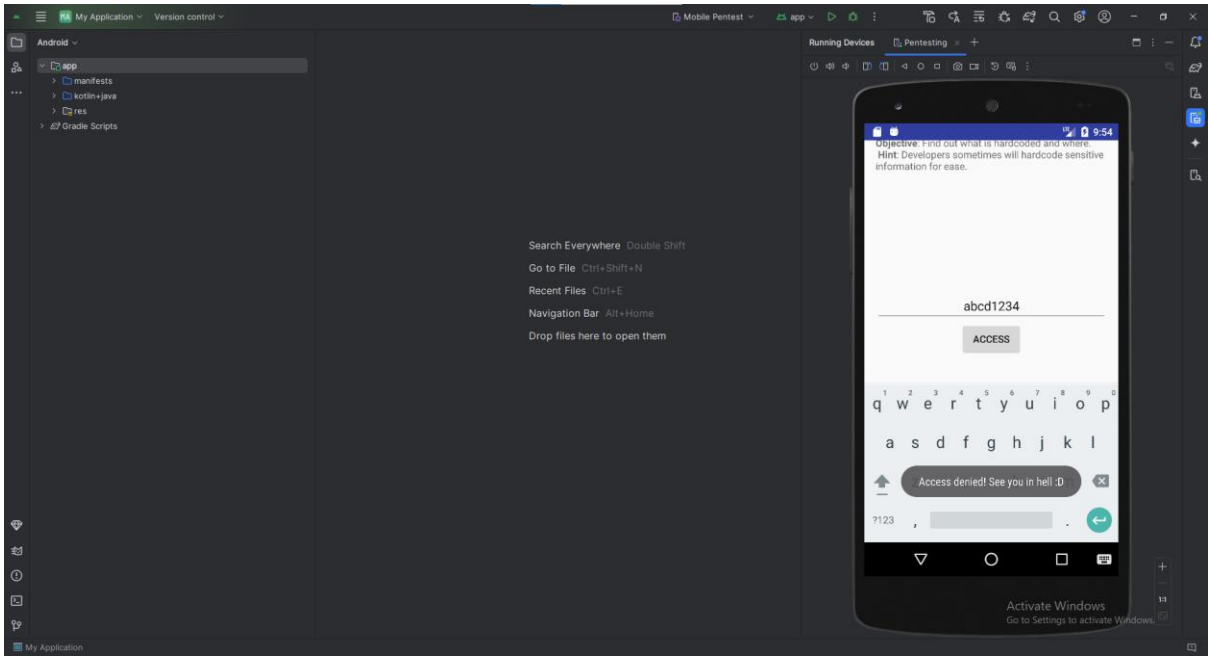


2. Hardcoding Issue_Part-1

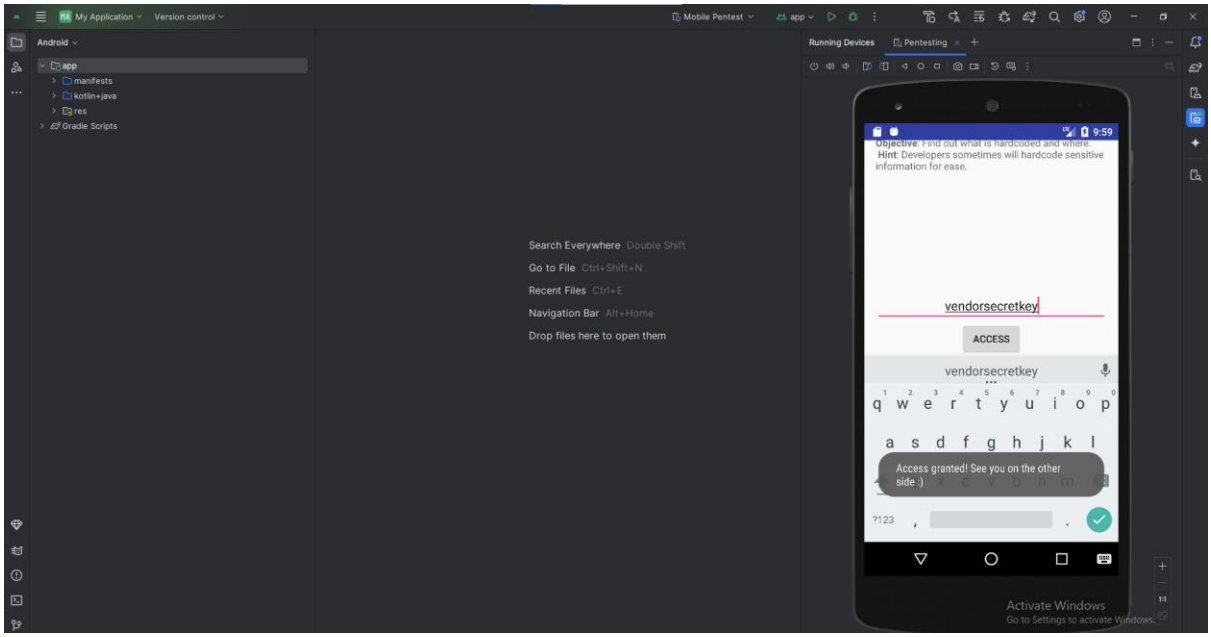
Screenshot-1: Sensitive information are hardcoded in source code of the .apk file.



Screenshot-2: Type wrong vendor code and get the error "Access denied! See you in hell :D"

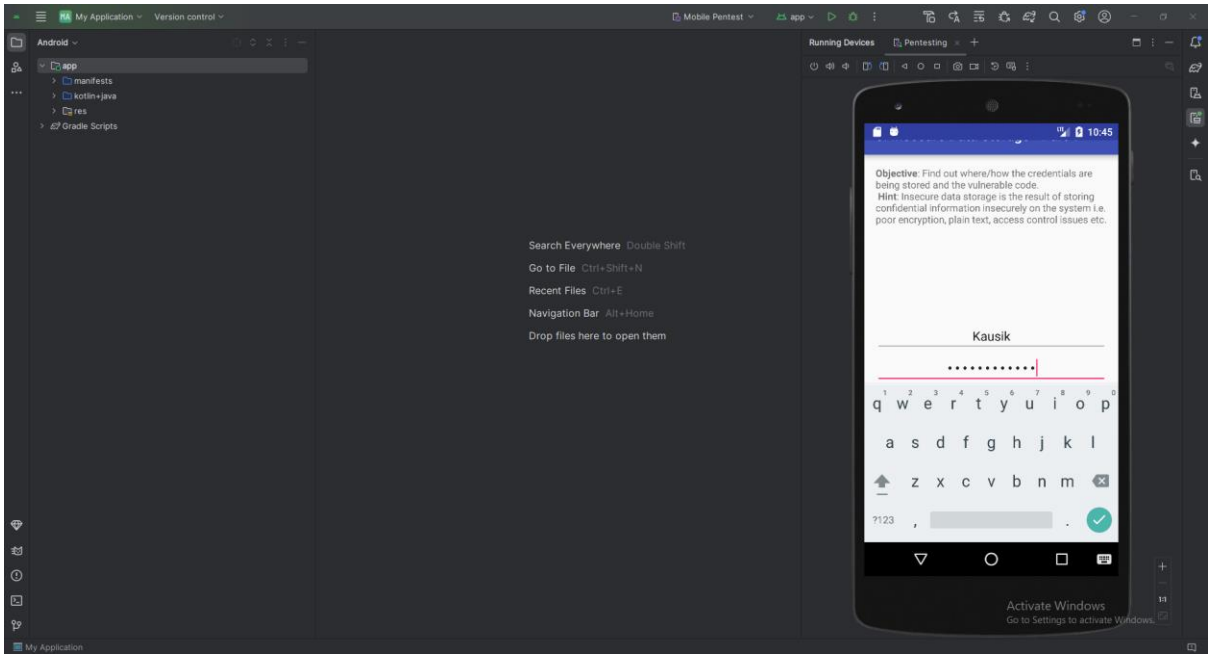


Screenshot-3: Type correct vendor code which is hardcoded in source code and get the error "Access granted! See you on the other side :)"

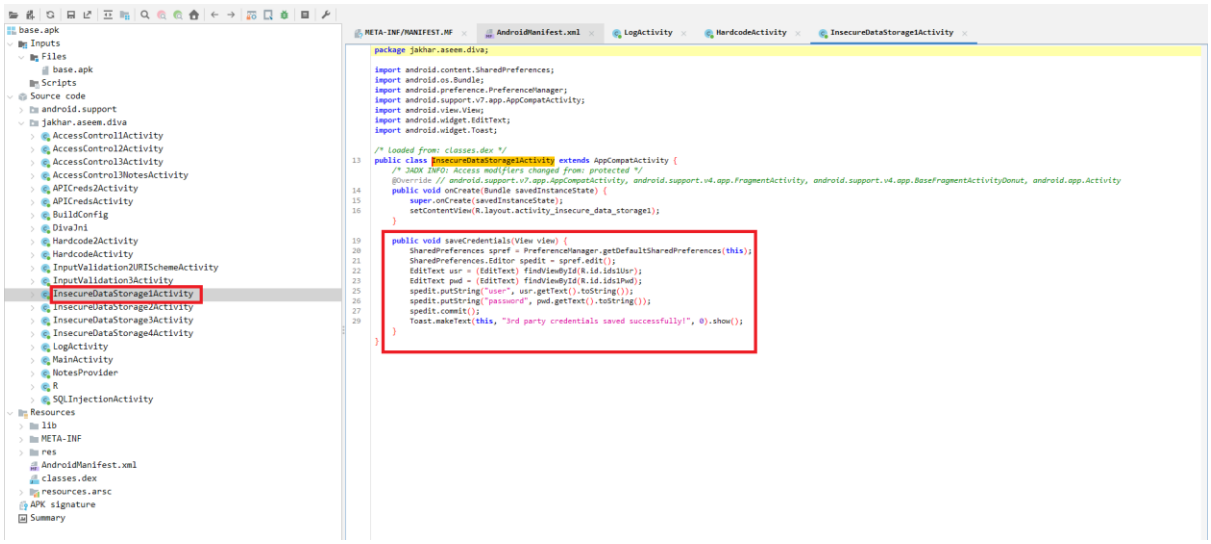


3. Insecure Data Storage_Part-1

Screenshot-1: Enter any random User ID and Password.



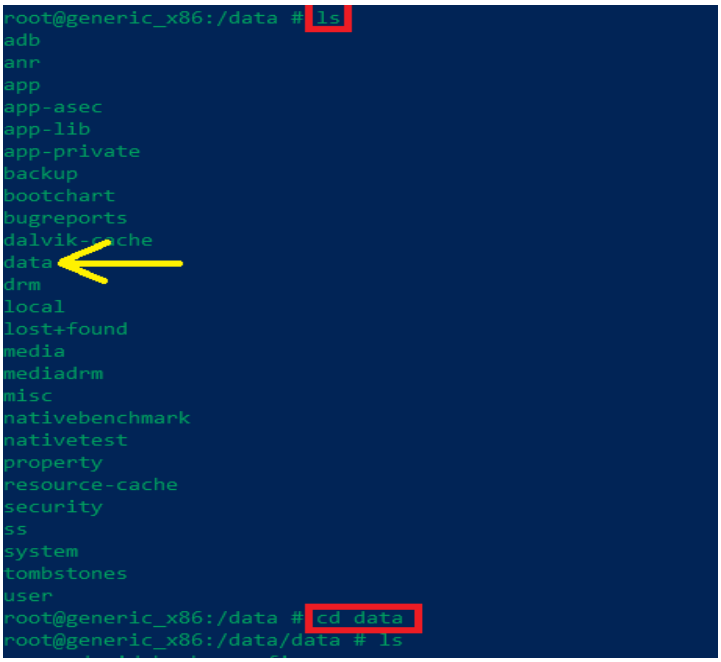
Screenshot-2: Respective activity & its source code. It could observe that credentials are storing inside of Shared Preferences file.

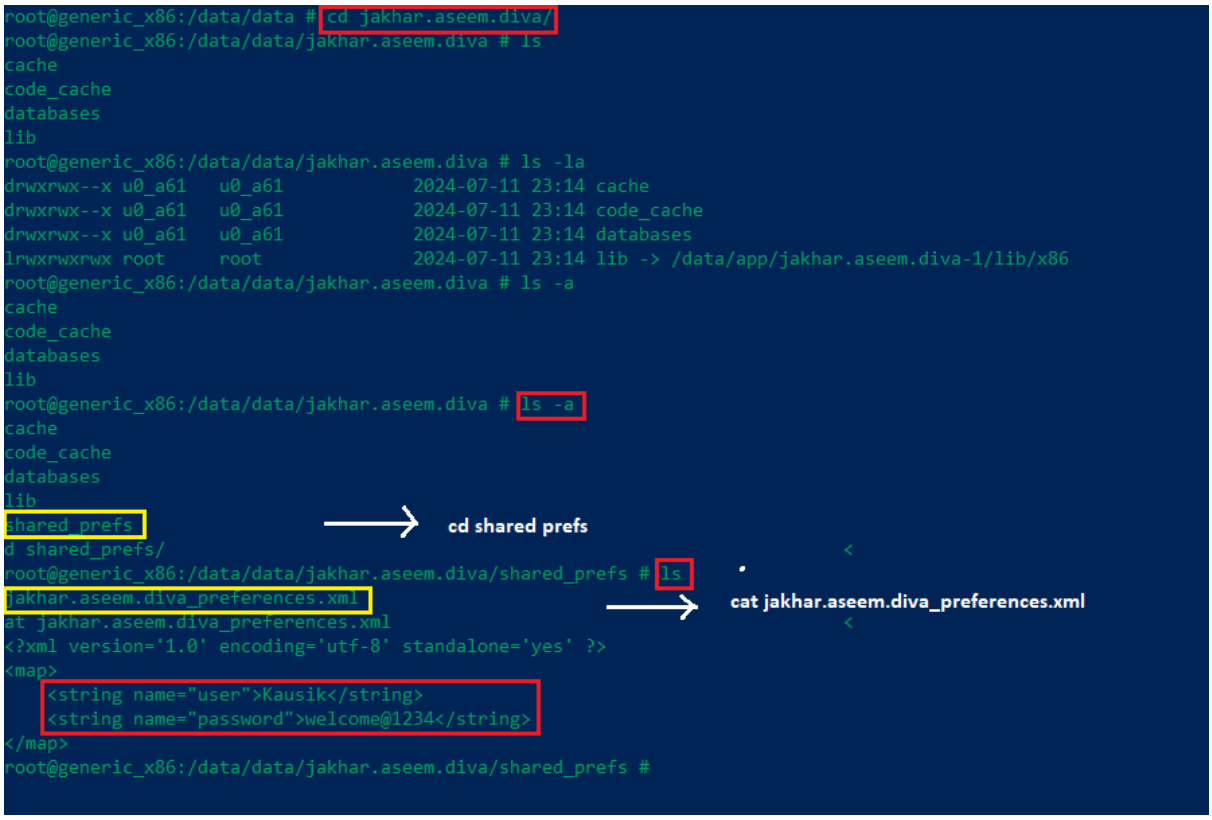


Screenshot-3:

Go to root using adb shell and go to the path **/data/data/jakhar.aseem.diva/shared_prefs #**
Now**,** we can find the file **jakhar.aseem.diva_preferences.xml** and also can see the entered User ID and Password as the content of the file.

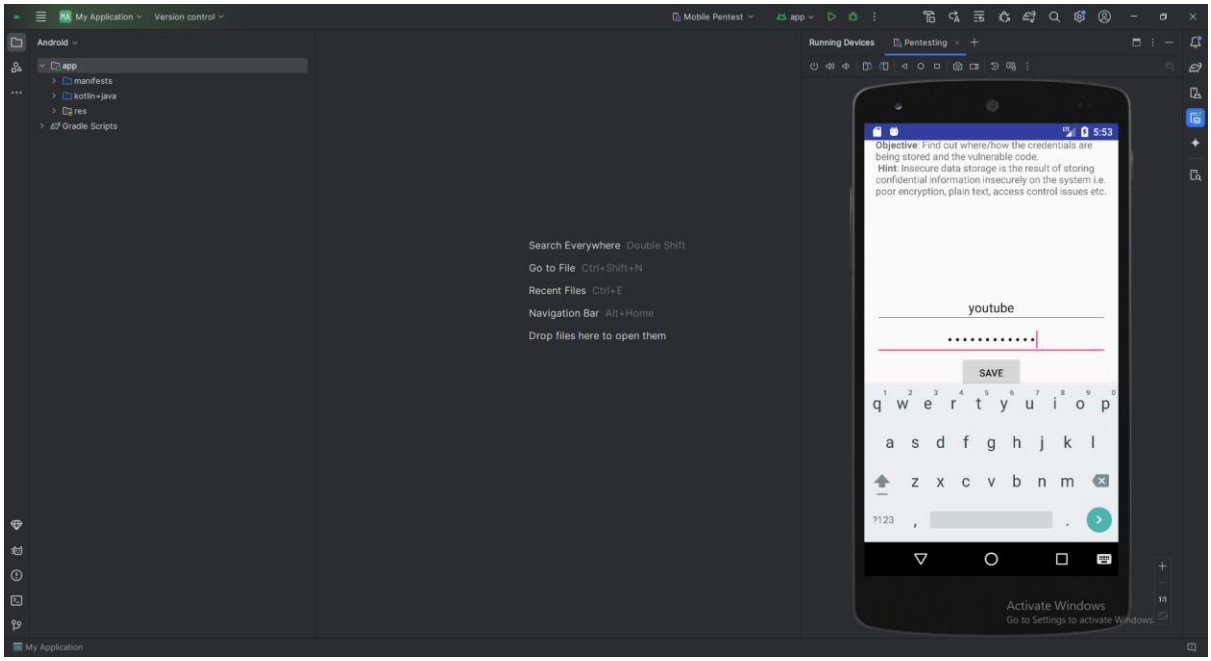
```
root@generic_x86:/data # exit
PS D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting> adb shell
root@generic_x86:/ # ls
acct
cache
charger
config
d
data
default.prop
dev
etc
file_contexts
fstab.goldfish
fstab.ranchu
fstab.ranchu-encrypt
init
init.environ.rc
init.goldfish.rc
init.ranchu-common.rc
init.ranchu-encrypt.rc
init.ranchu.rc
init.rc
init.trace.rc
init.usb.rc
init.zygote32.rc
mnt
oem
proc
property_contexts
root
sbin
sdcard
seapp_contexts
selinux_version
sepolicy
service_contexts
storage
sys
system
ueventd.goldfish.rc
ueventd.ranchu.rc
ueventd.rc
vendor
root@generic_x86:/ # cd data
root@generic_x86:/data # ls
```


root@generic_x86:/data # ls
adb
anr
app
app-asec
app-lib
app-private
backup
bootchart
bugreports
dalvik-cache
data
drm
local
lost+found
media
mediadrm
misc
nativebenchmark
nativetest
property
resource-cache
security
ss
system
tombstones
user
root@generic_x86:/data # cd data
root@generic_x86:/data/data # ls

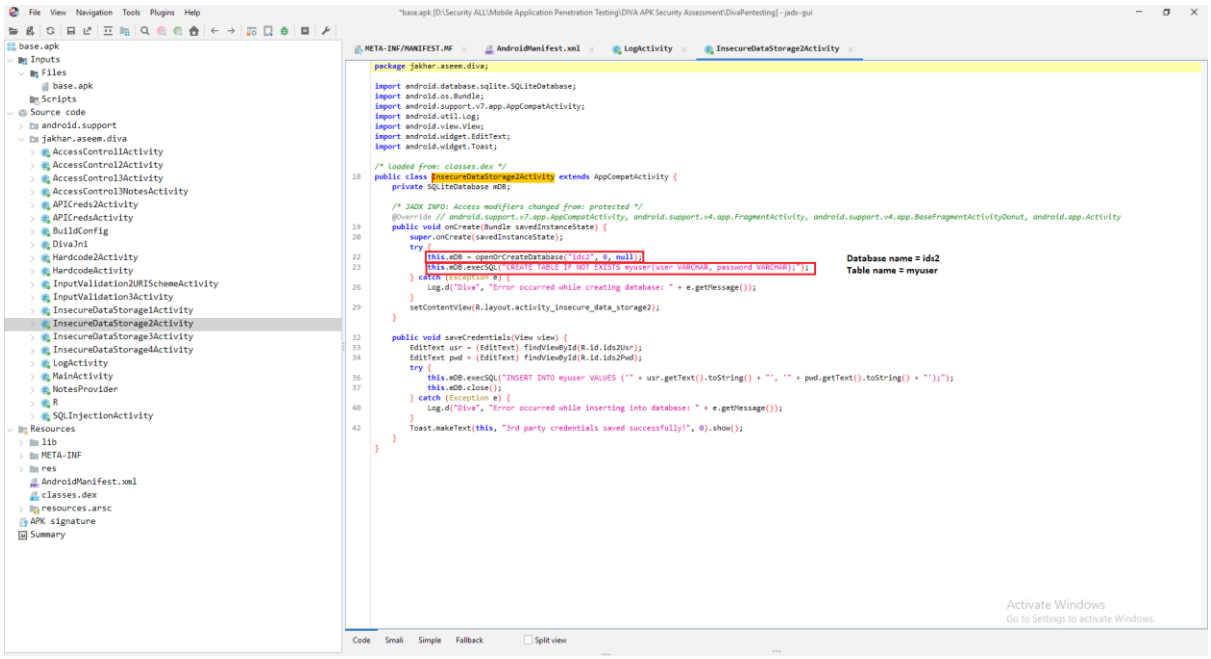

root@generic_x86:/data/data # cd jakhar.aseem.diva/
root@generic_x86:/data/data/jakhar.aseem.diva # ls
cache
code_cache
databases
lib
root@generic_x86:/data/data/jakhar.aseem.diva # ls -la
drwxrwx--x u0_a61 u0_a61 2024-07-11 23:14 cache
drwxrwx--x u0_a61 u0_a61 2024-07-11 23:14 code_cache
drwxrwx--x u0_a61 u0_a61 2024-07-11 23:14 databases
lrwxrwxrwx root root 2024-07-11 23:14 lib -> /data/app/jakhar.aseem.diva-1/lib/x86
root@generic_x86:/data/data/jakhar.aseem.diva # ls -a
cache
code_cache
databases
lib
root@generic_x86:/data/data/jakhar.aseem.diva # ls -a
cache
code_cache
databases
lib
shared_prefs
cd shared prefs
d shared_prefs/
root@generic_x86:/data/data/jakhar.aseem.diva/shared_prefs # ls
jakhar.aseem.diva_preferences.xml
cat jakhar.aseem.diva_preferences.xml
at jakhar.aseem.diva_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="user">Kausik</string>
<string name="password">welcome@1234</string>
</map>
root@generic_x86:/data/data/jakhar.aseem.diva/shared_prefs #

4. Insecure Data Storage_Part-2

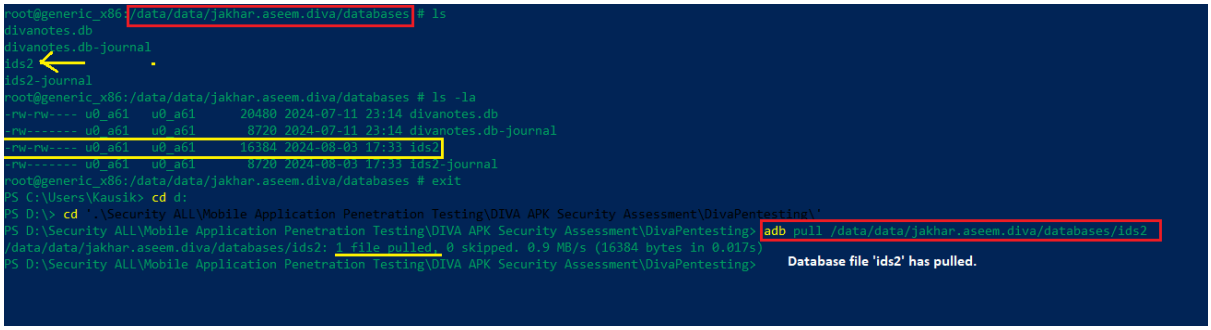
Screenshot-1: Enter any random User ID and Password.



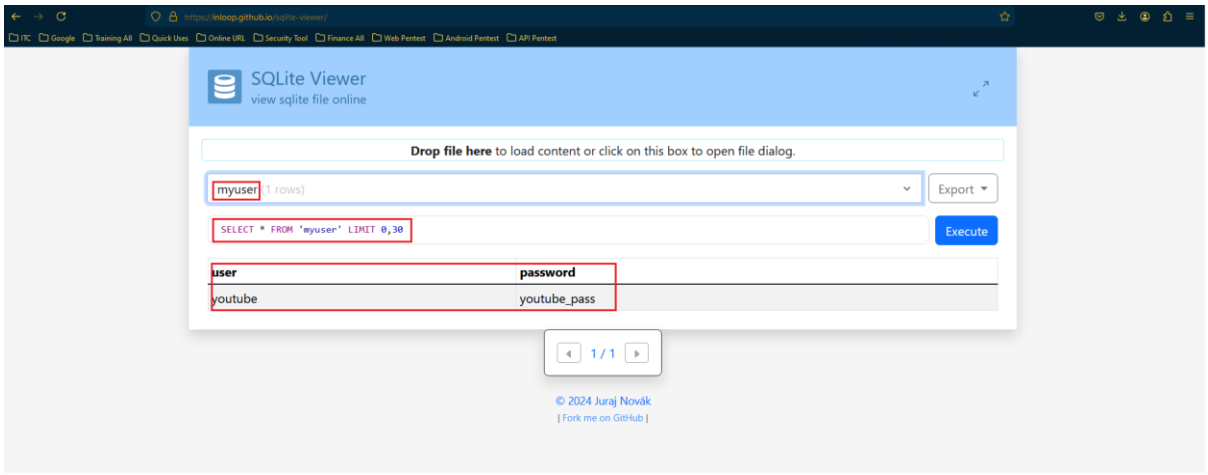
Screenshot-2: Respective activity & its source code. We can find the name of database and table name.



Screenshot-3:
Go to root using adb shell and go to the path **/data/data/jakhar.aseem.diva/databases #**
Now**,** we can see the database file & pull the file **ids2.**

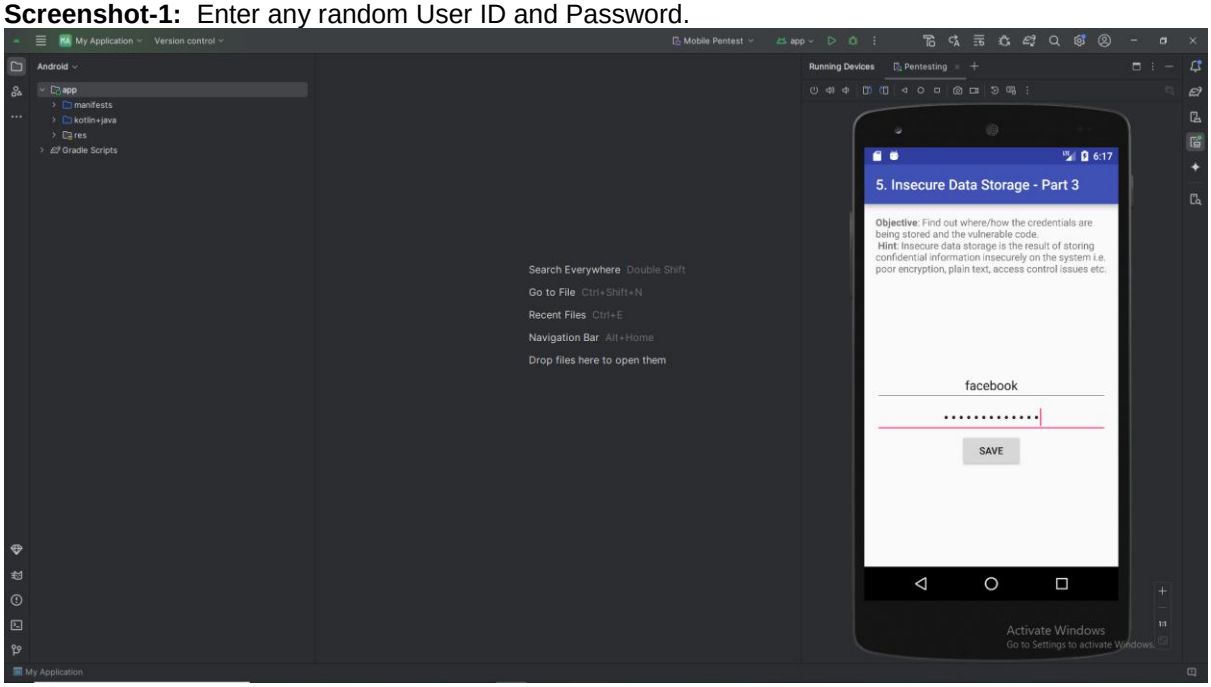


Screenshot-4: Open the database file **ids2** using online tool sqllite3 viewer and select the respective table **myuser**. You can find entered User ID and Password.

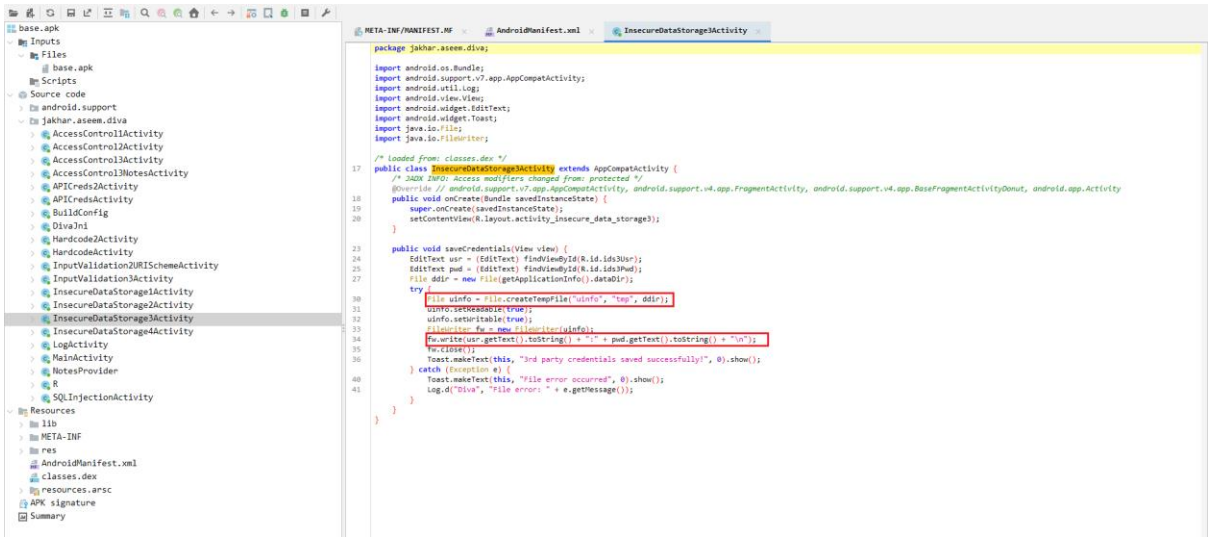


5. Insecure Data Storage_Part-3

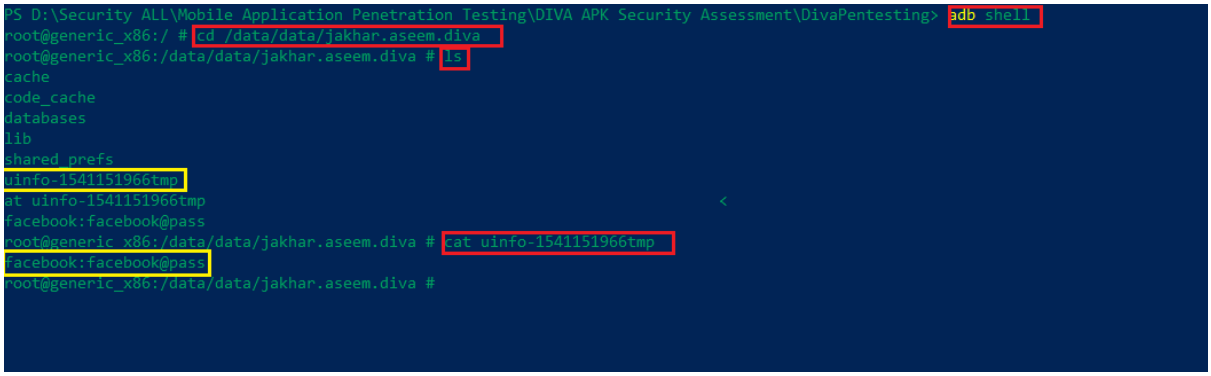
Screenshot-1: Enter any random User ID and Password.



Screenshot-2: Respective activity & its source code. We can see that entered credentials are storing in a file name as uinfo tmp in username:password format.

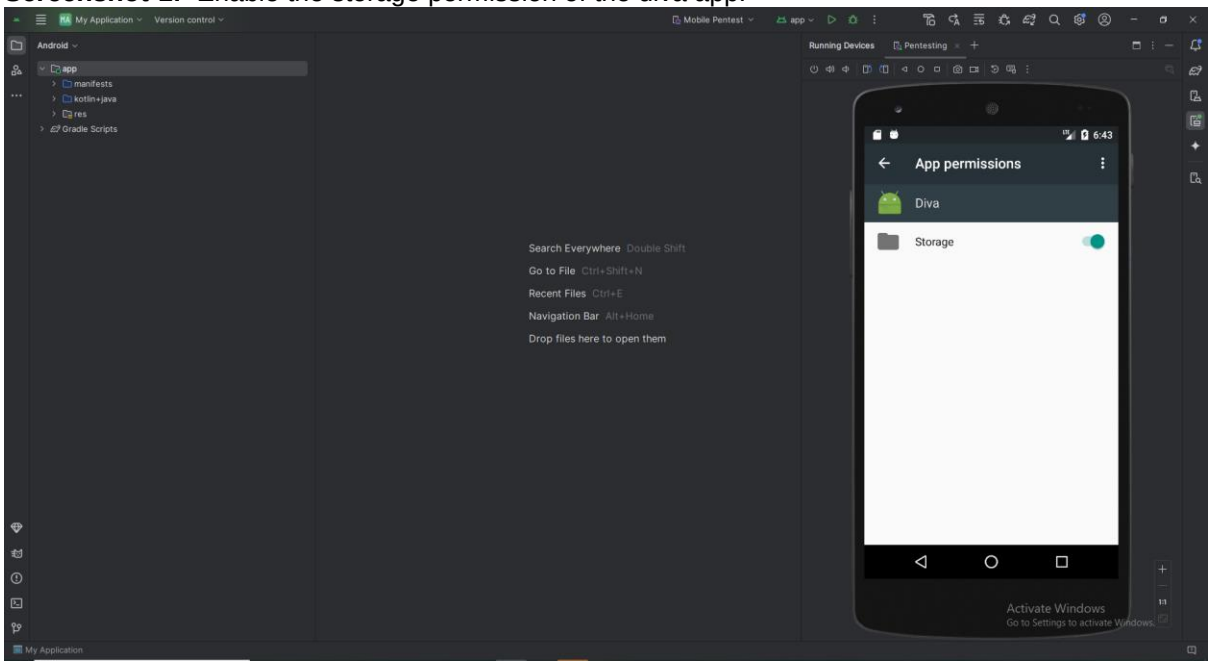


Screenshot-3:
Go to root using adb shell and go to the path **/data/data/jakhar.aseem.diva/**
Now**,** we can find the file **uinfo-1541151966tmp** and can see the credentials as content of the file.

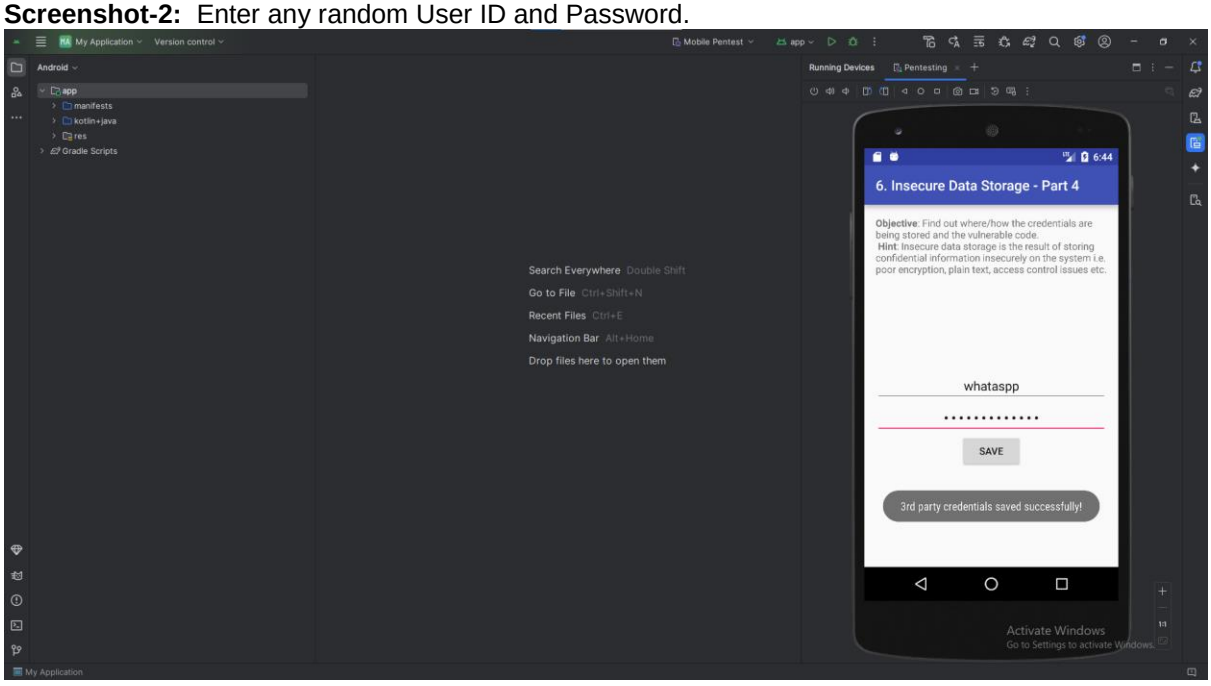


6. Insecure Data Storage_Part-4

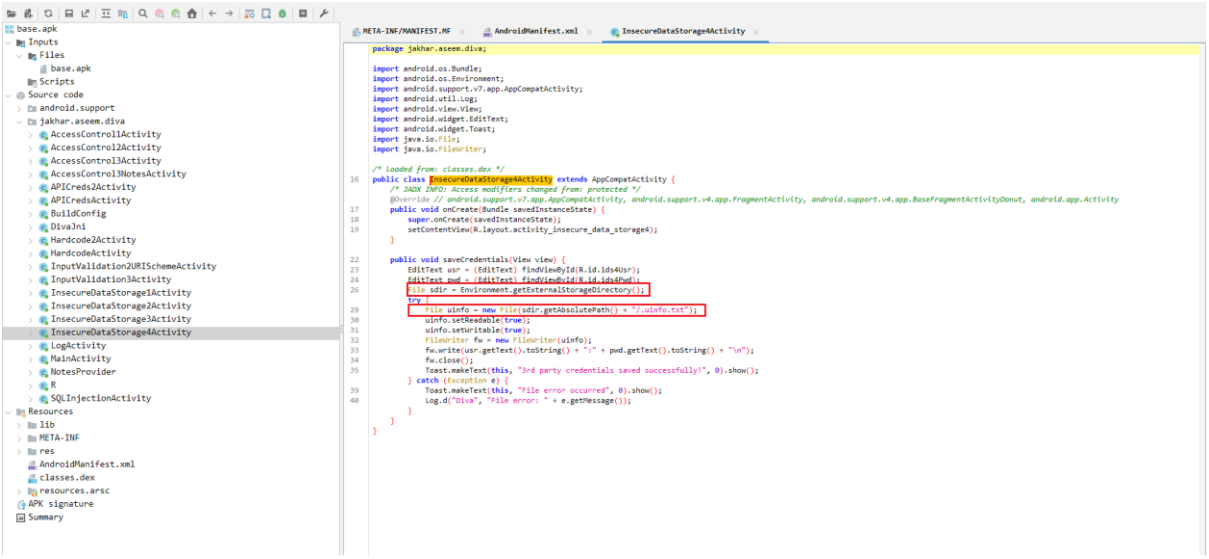
Screenshot-1: Enable the storage permission of the diva app.



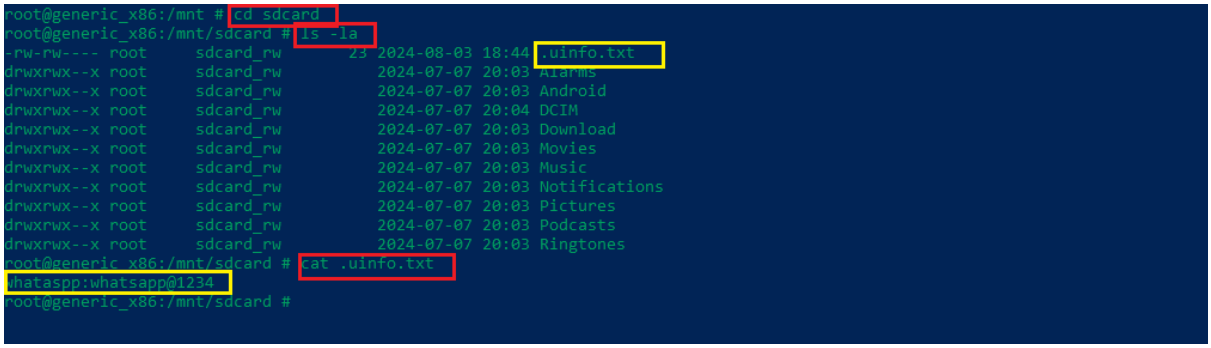
Screenshot-2: Enter any random User ID and Password.



Screenshot-3: Respective activity & its source code. We can see that entered credentials are storing in a file name as.uinfo.txt inside of sdcard external storage.

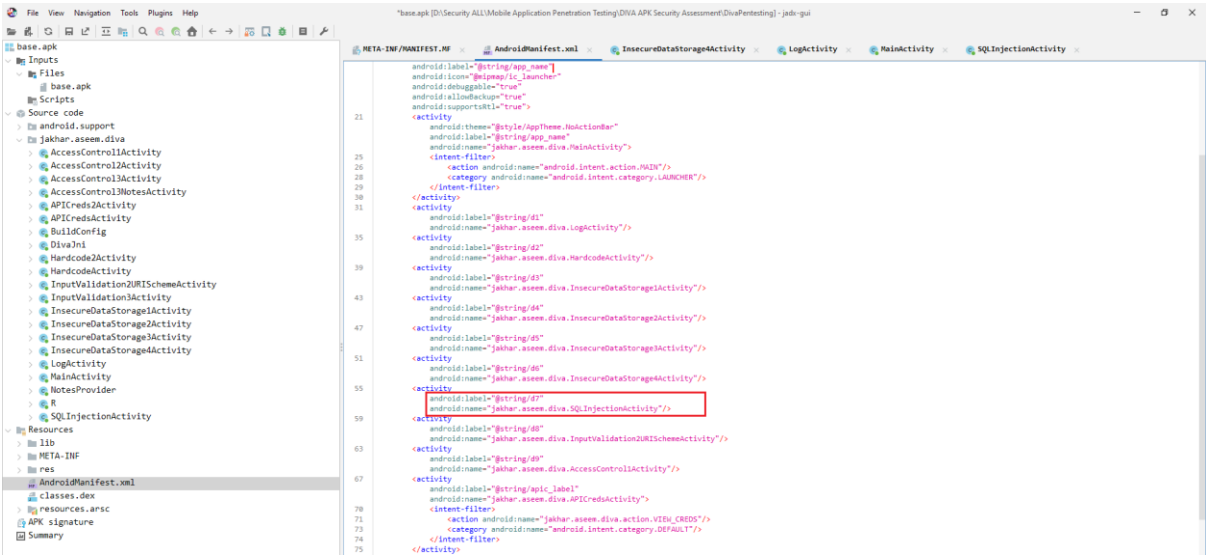


Screenshot-3:
Go to root using adb shell and go to the path **/mnt/sdcard**
Now**,** we can find the hidden file .uinfo.txt and can see the credentials as content of the file.

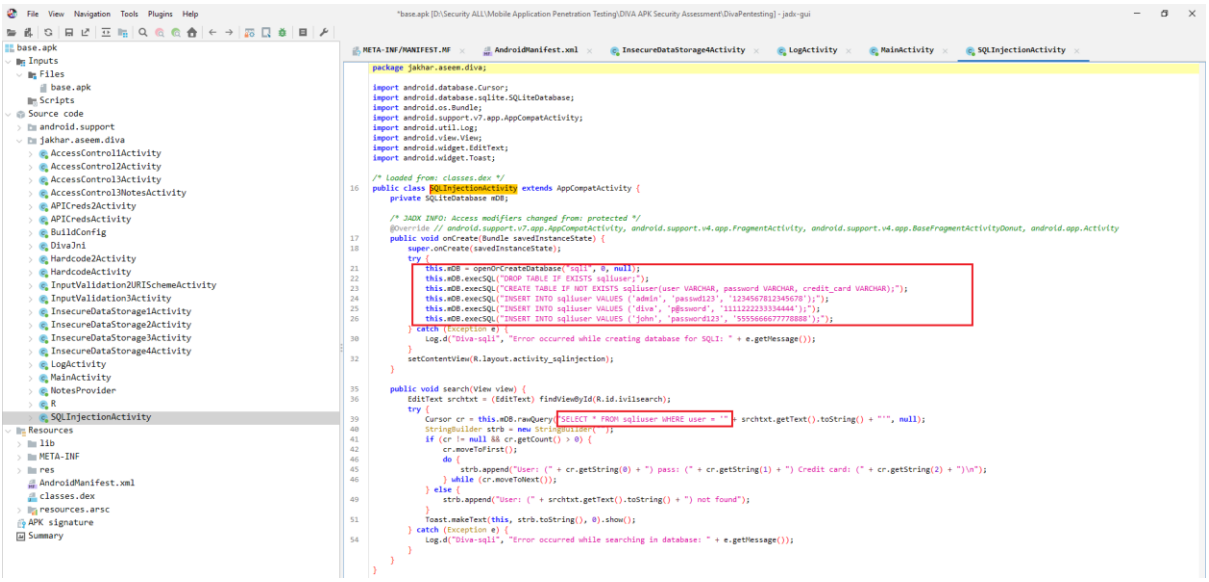


7. Input Validation Issues_Part-1

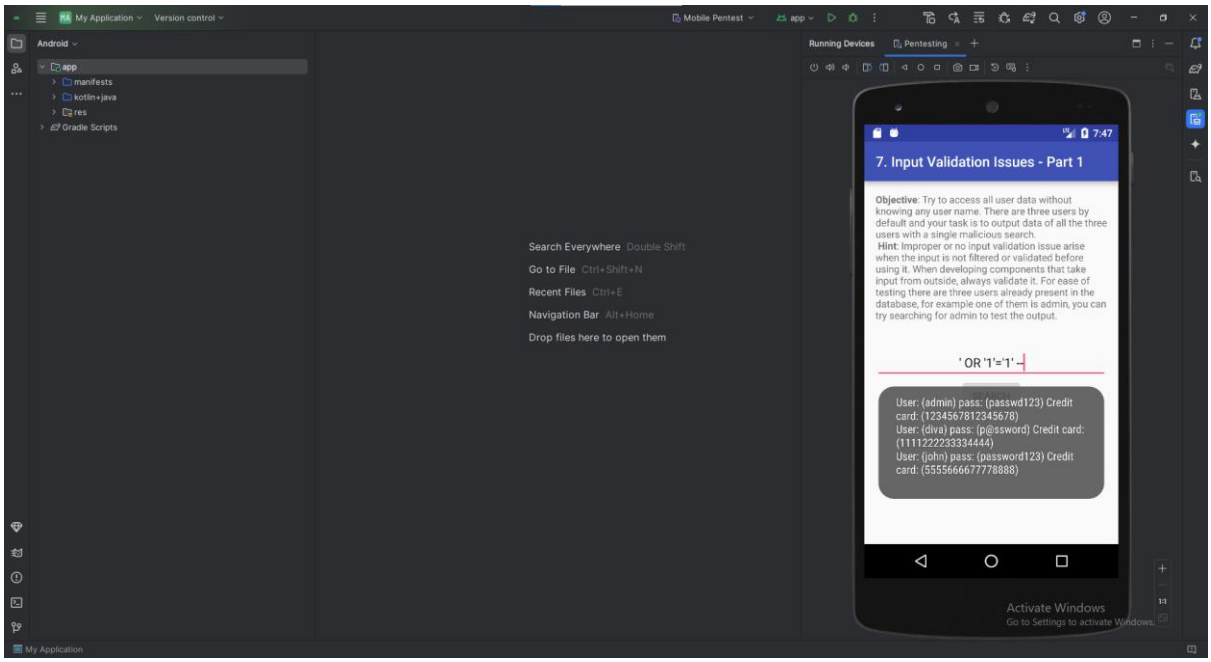
Screenshot-1: Check the AndroidManifest.xml file to find out the exact activity.



Screenshot-2: Respective activity & its source code. We can see that user information details are stored in SQL database.

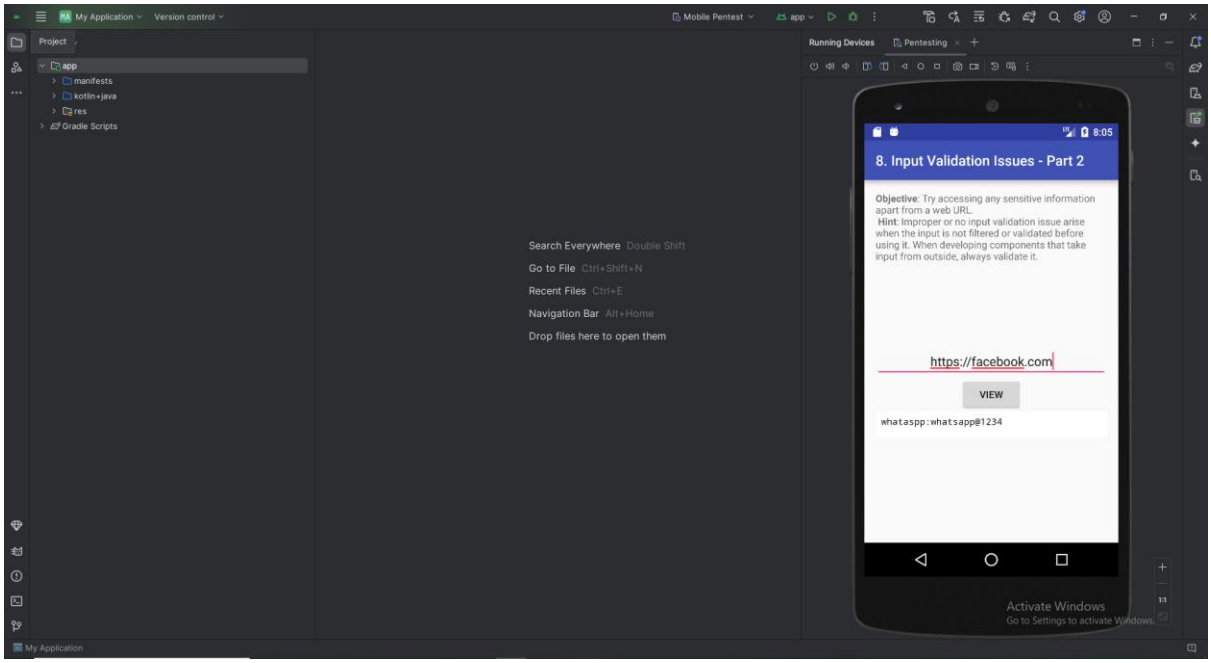


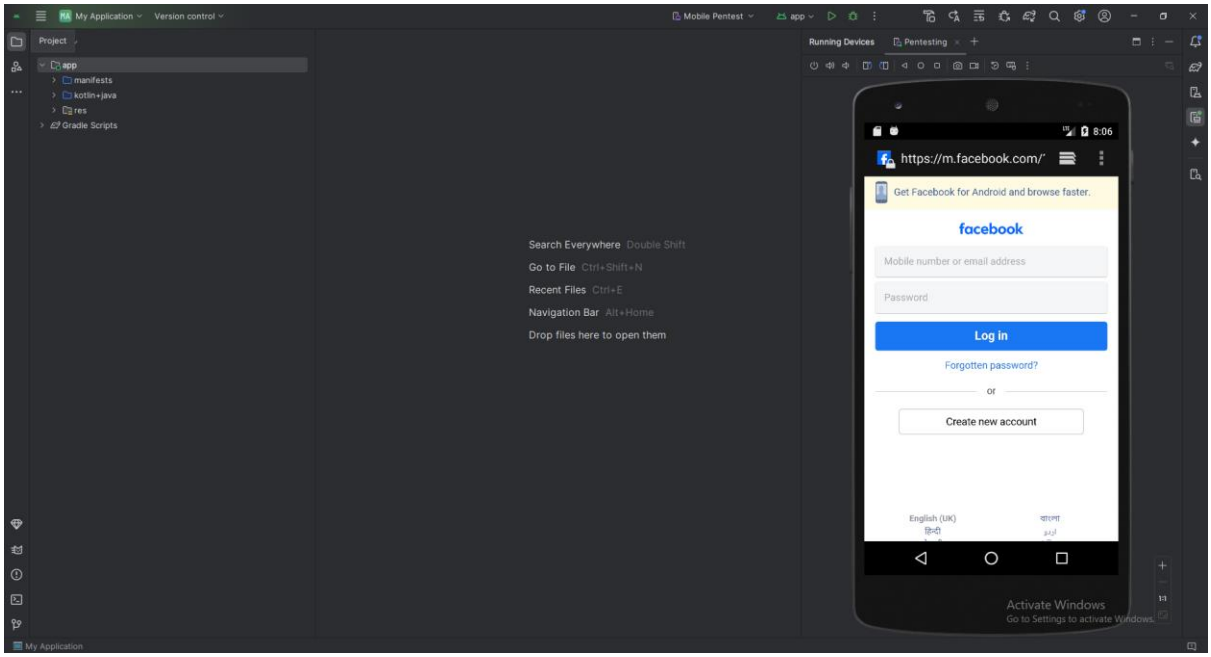
Screenshot-3: We can perform SQL Injection with single malicious payload ' OR '1'='1' – and can see all user's details.



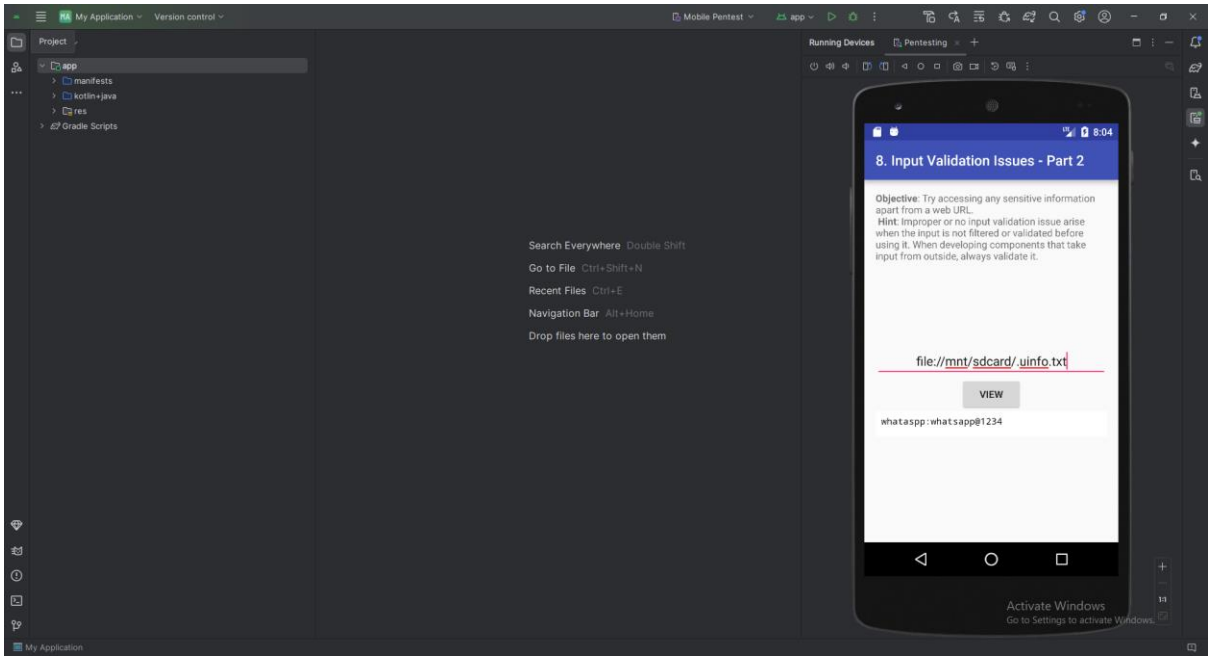
8. Input Validation Issues_Part-2

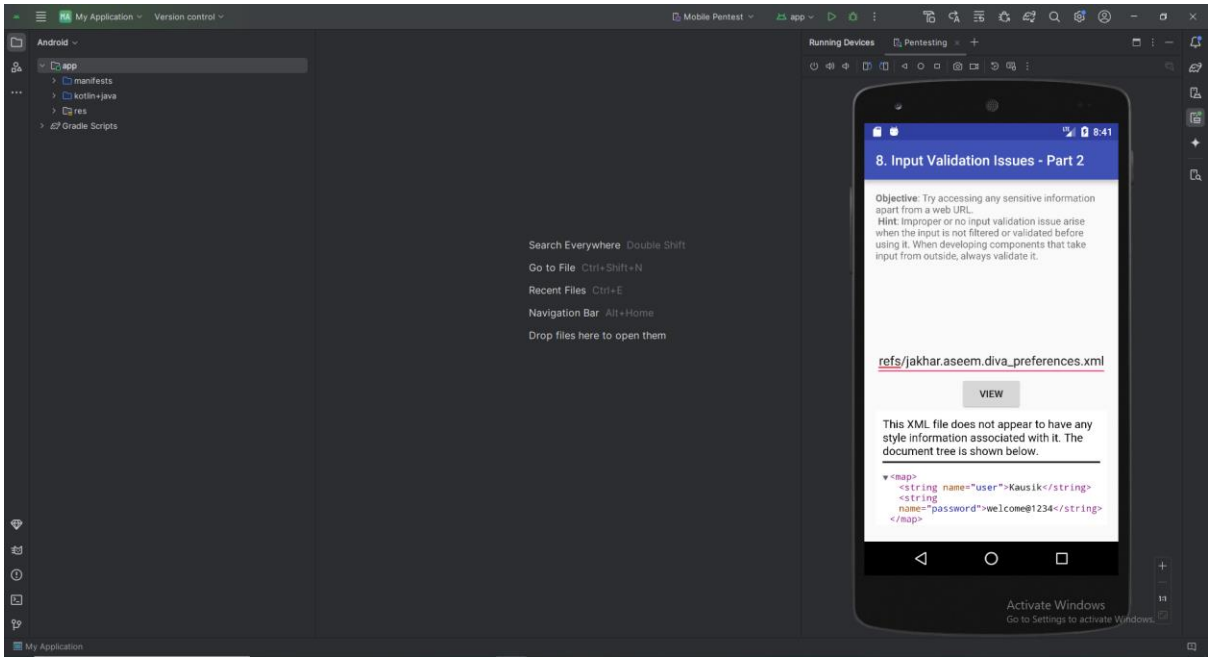
Screenshot-1: Type any URL and see that webpage is working as normal.





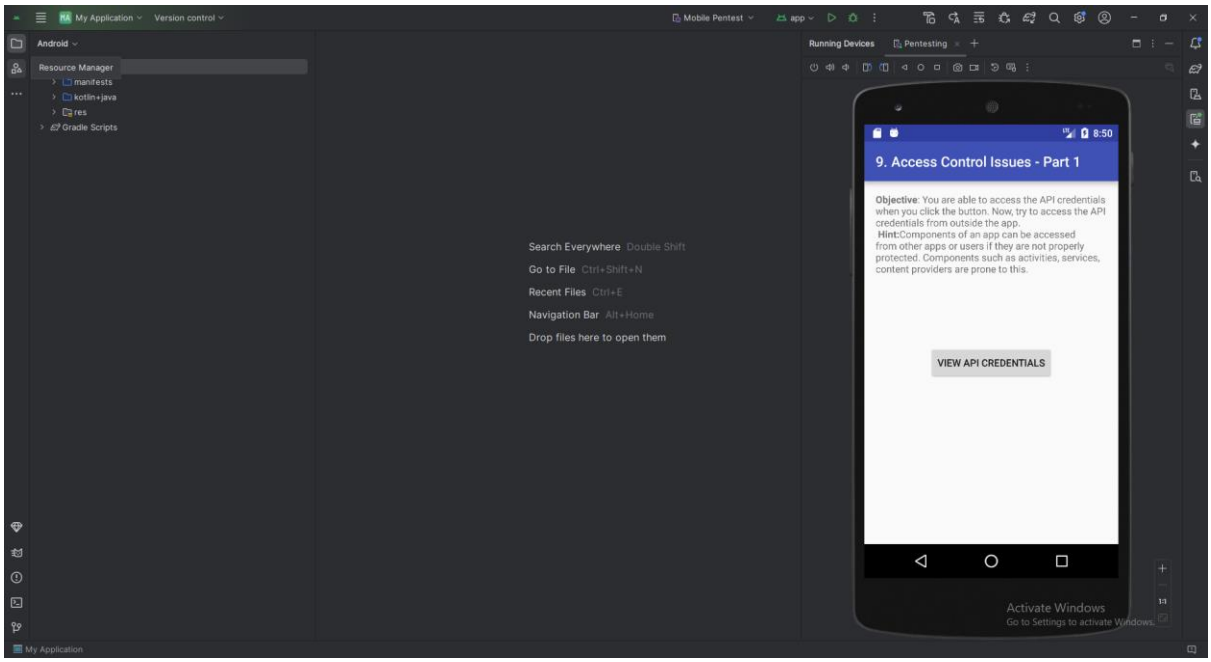
Screenshot-2: Enter any file path file://mnt/sdcard/.uinfo.txt and see the content. This is called Local File Inclusion (LFI).



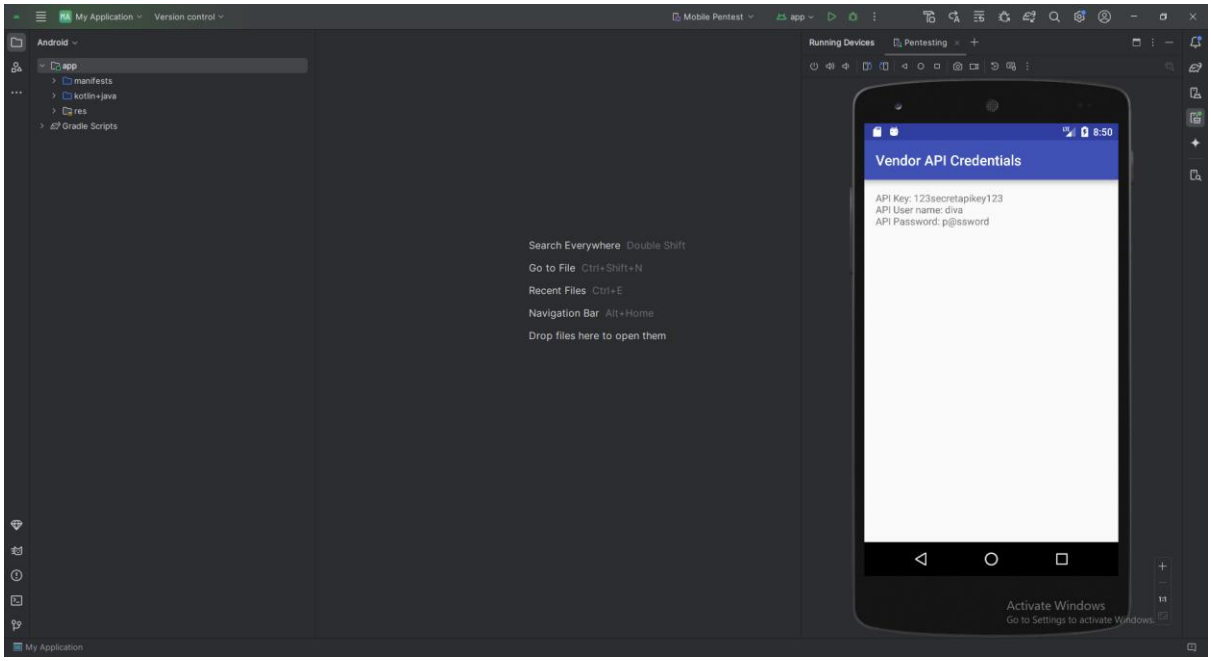


9. Access Control Issues_Part-1

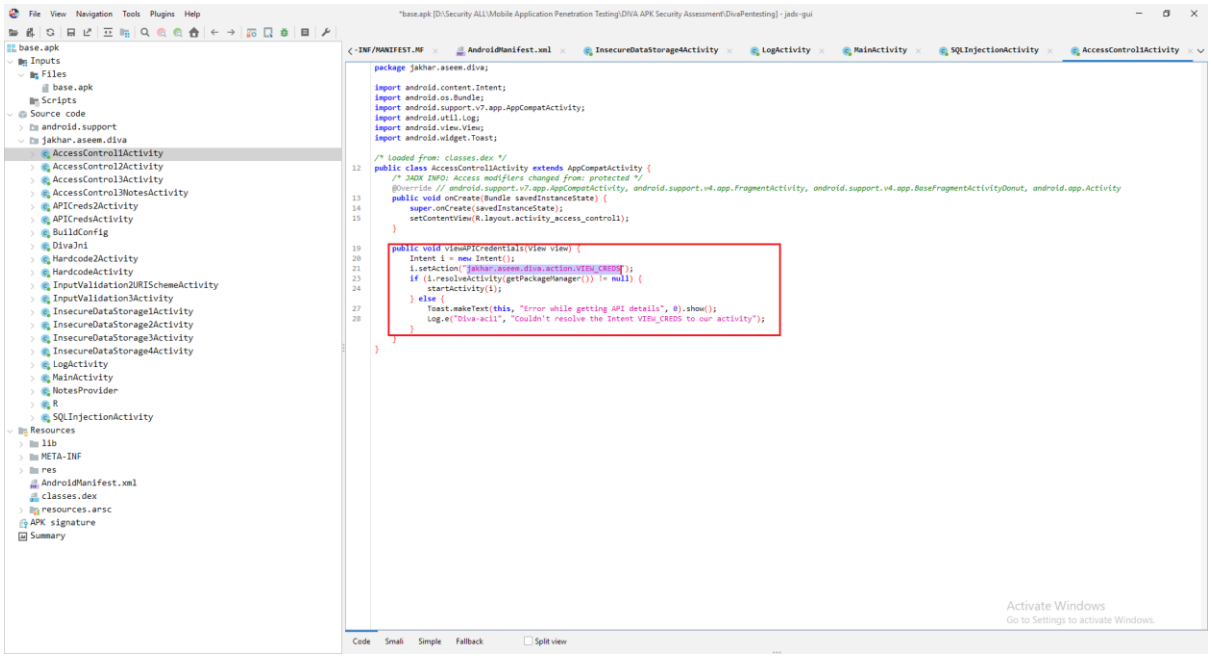
Screenshot-1: Until we are clicking the button, API credentials will not be shown.



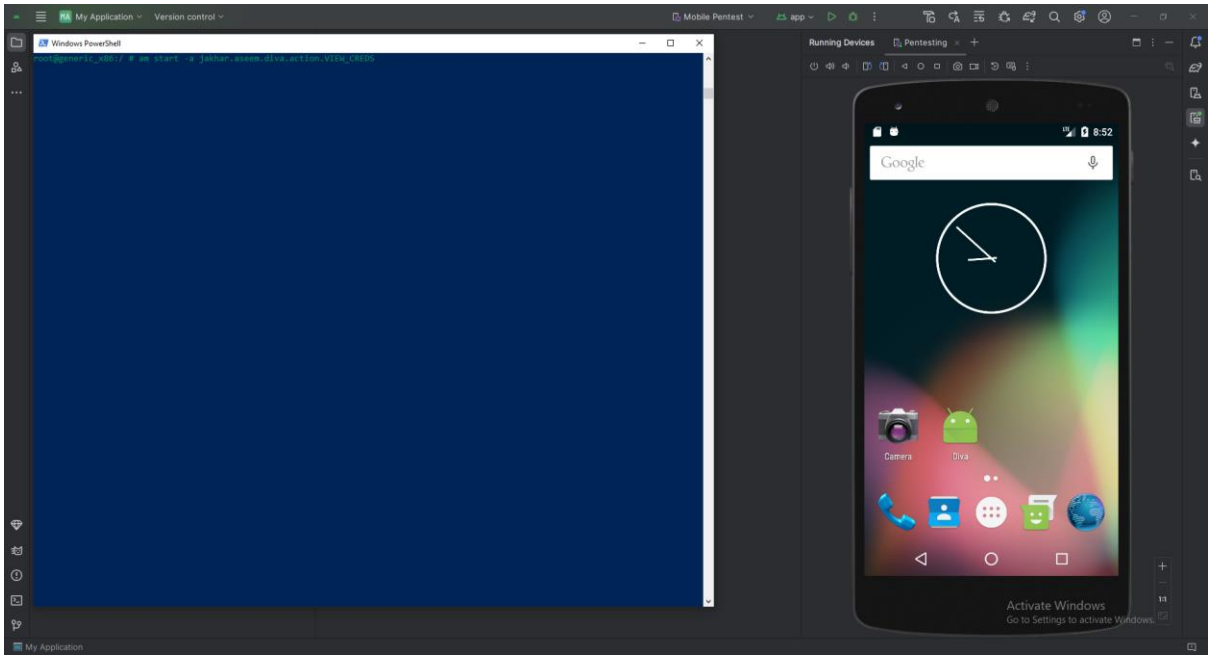
Screenshot-2: Once we are clicking the button, API credentials will be shown.



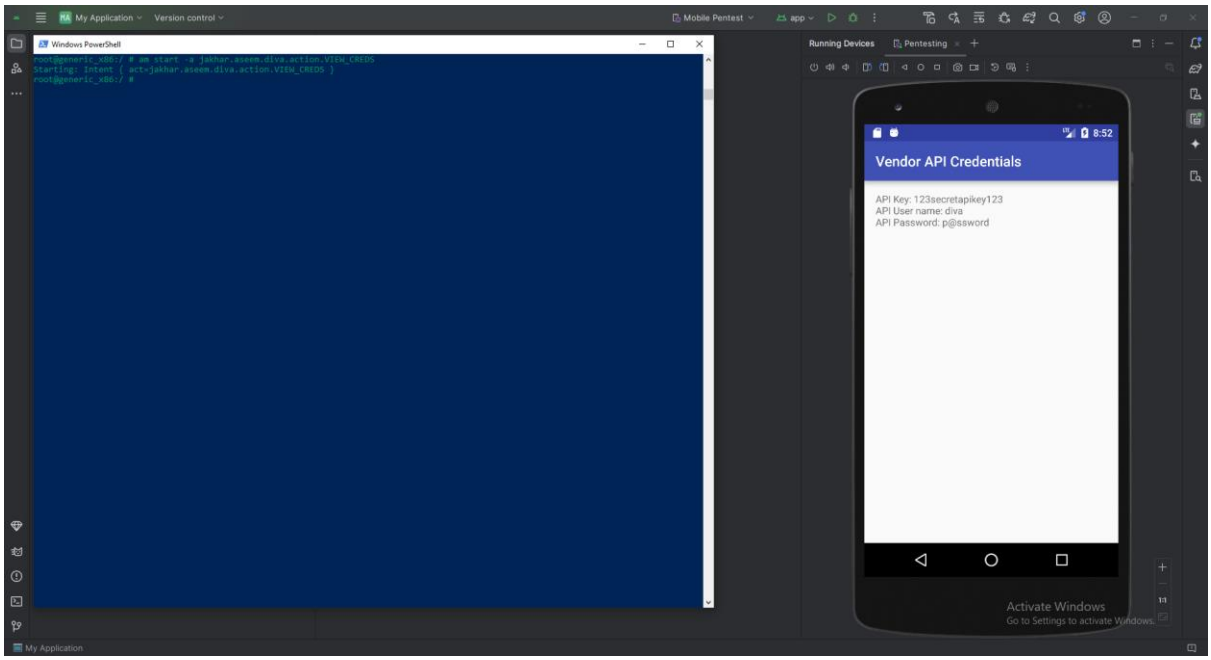
Screenshot-3: Respective activity & its source code. We can see the activity name by starting which we can see the API credentials on the devices without open the app explicitly.



Screenshot-4: Now the DIVA app is closed.

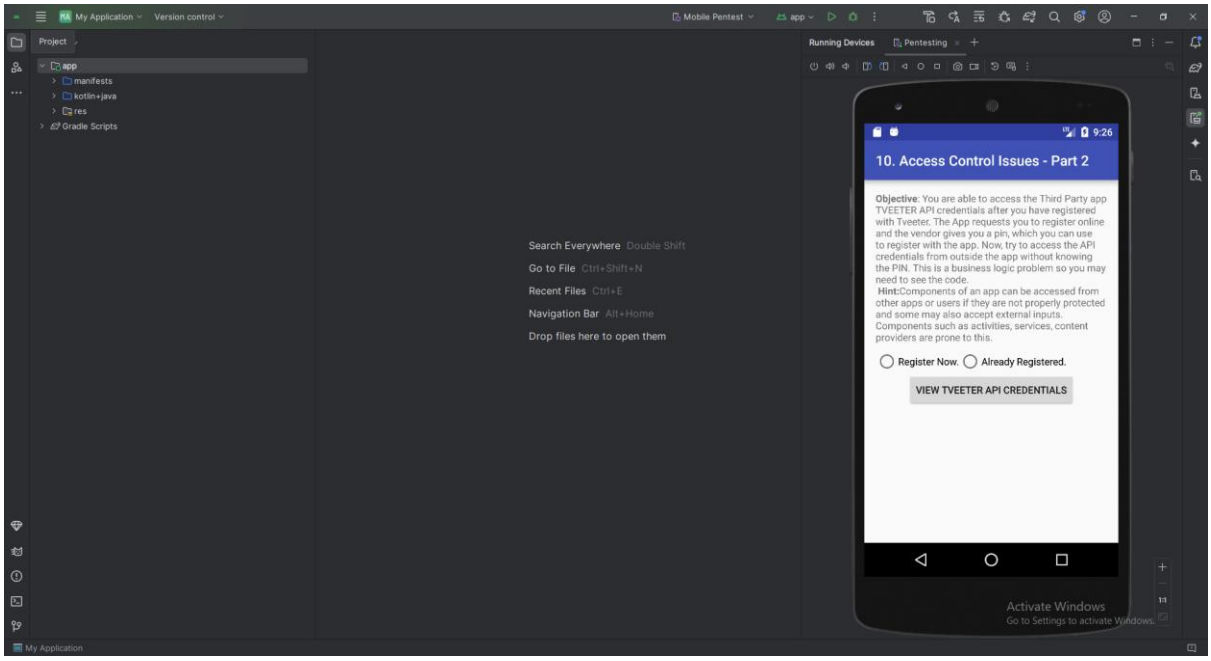


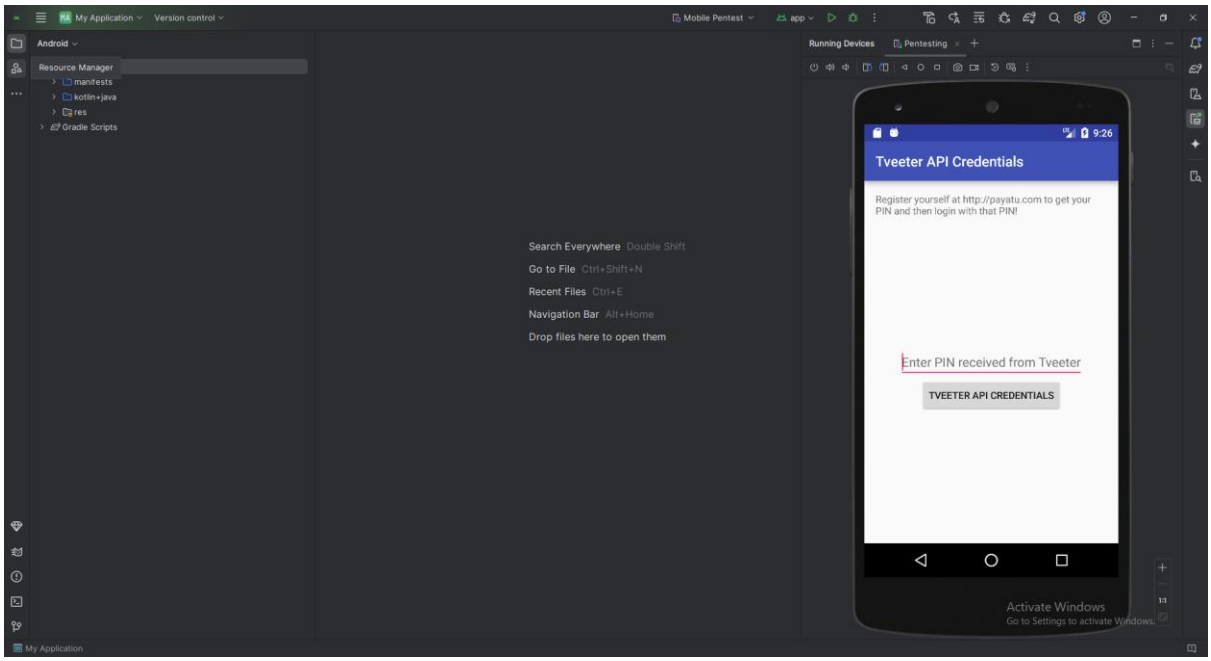
Screenshot-5: Now from adb shell try to start the activity by using the command `am start – a < jakhar.aseem.diva.action.VIEW_CREDS >` and after entering hit API Credentials are exposing on the emulator without validating any authorization.



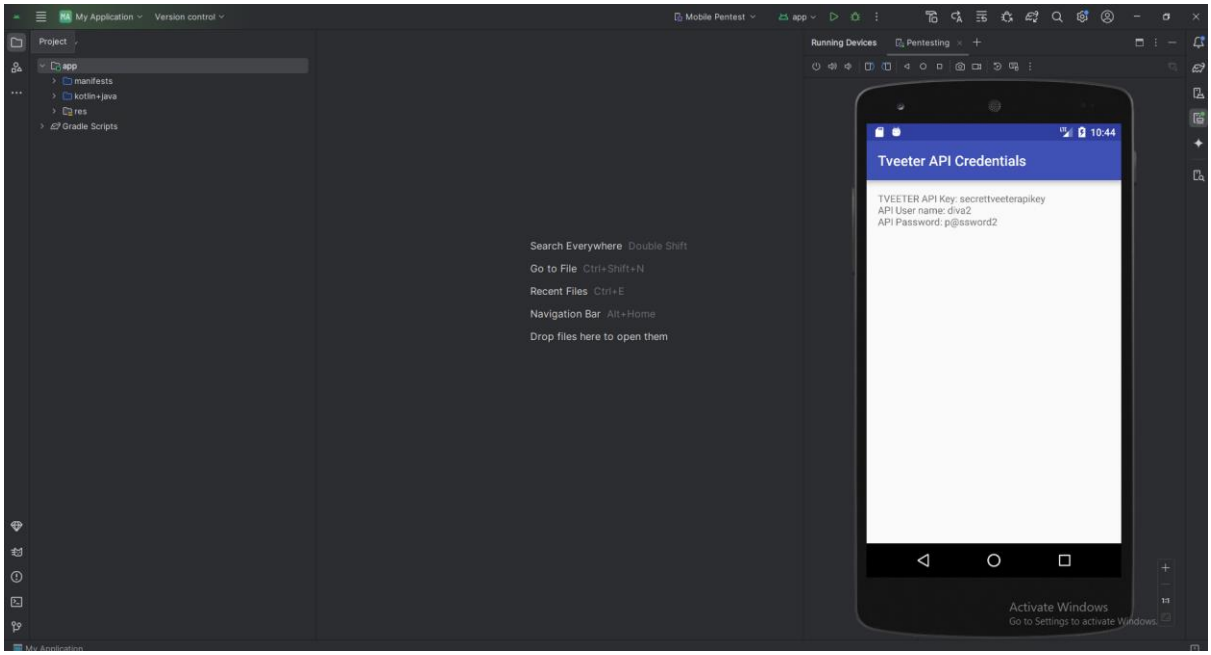
10. Access Control Issues_Part-2

Screenshot-1: The 1st Activity is `jakhar.aseem.diva.APICreds2Activity` with chk pin verification step.

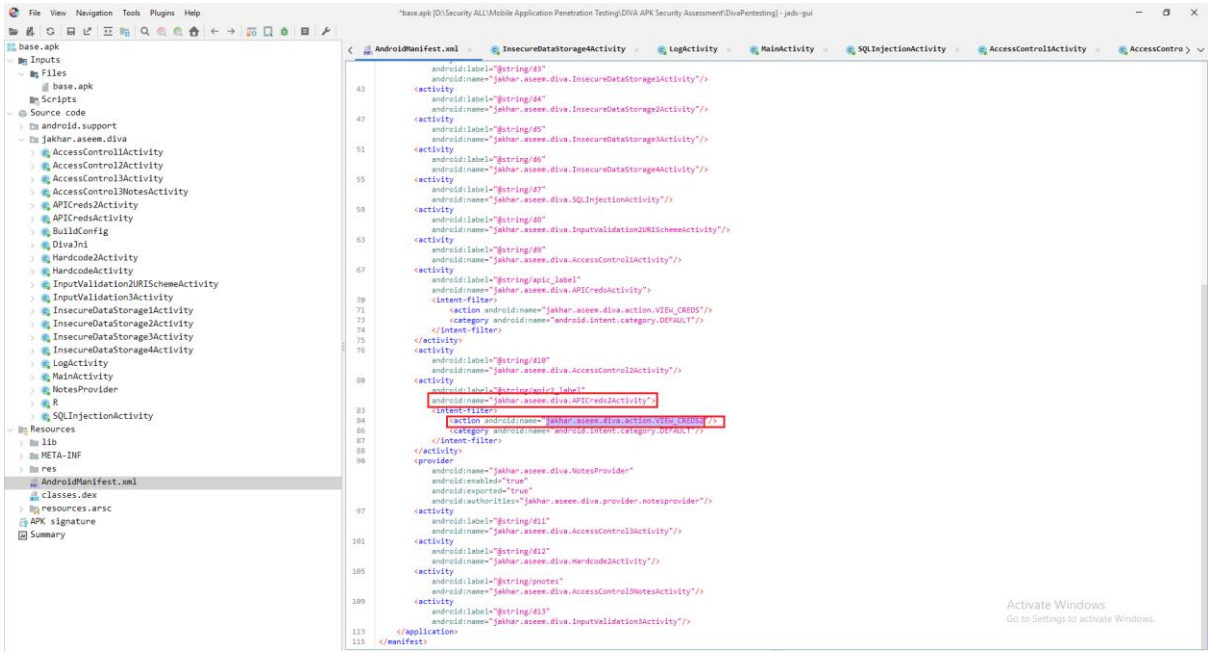




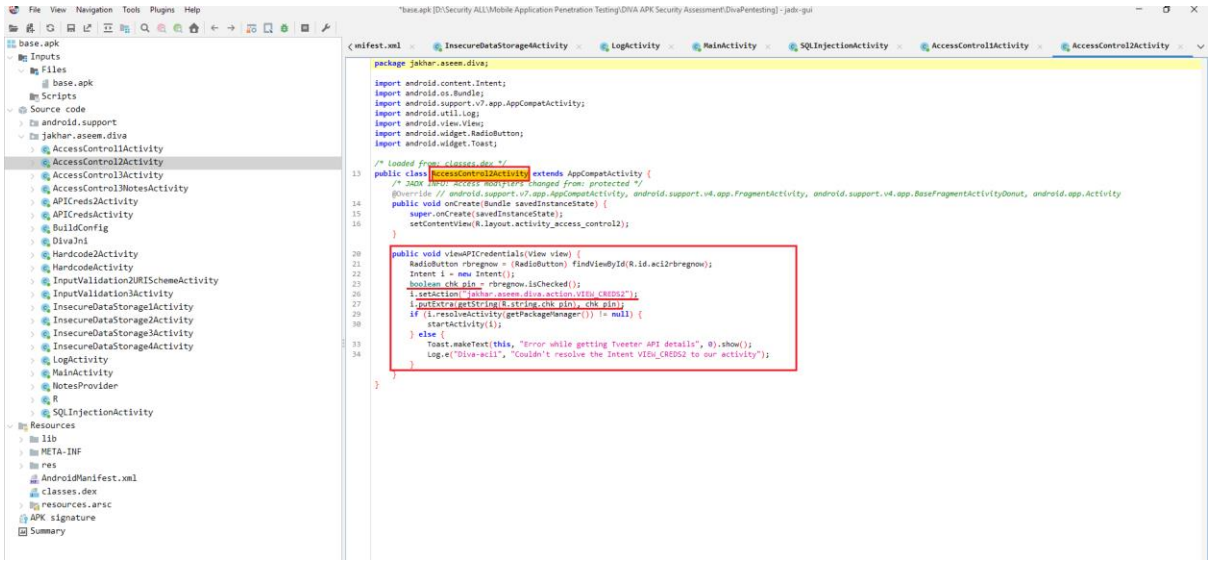
Screenshot-2: The 2nd Activity is `jakhar.aseem.diva.action.VIEW_CREDS2` which will show API Credentials.

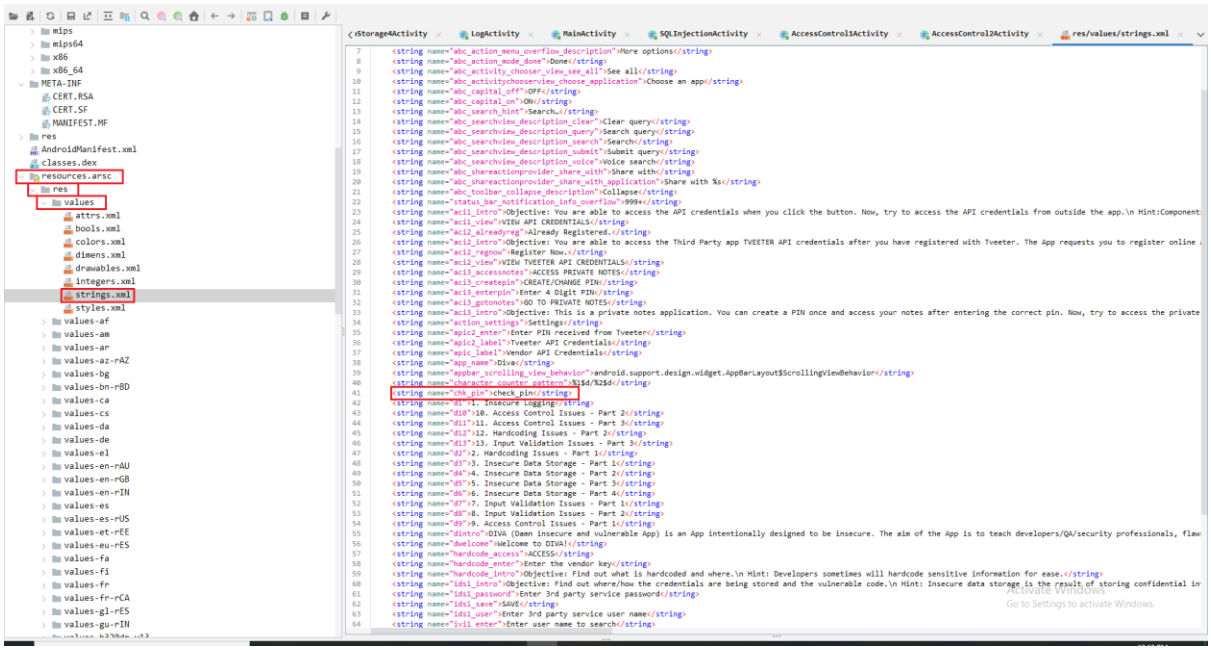


Screenshot-3: 1st and 2nd activity name details from AndroidManifest.xml file.

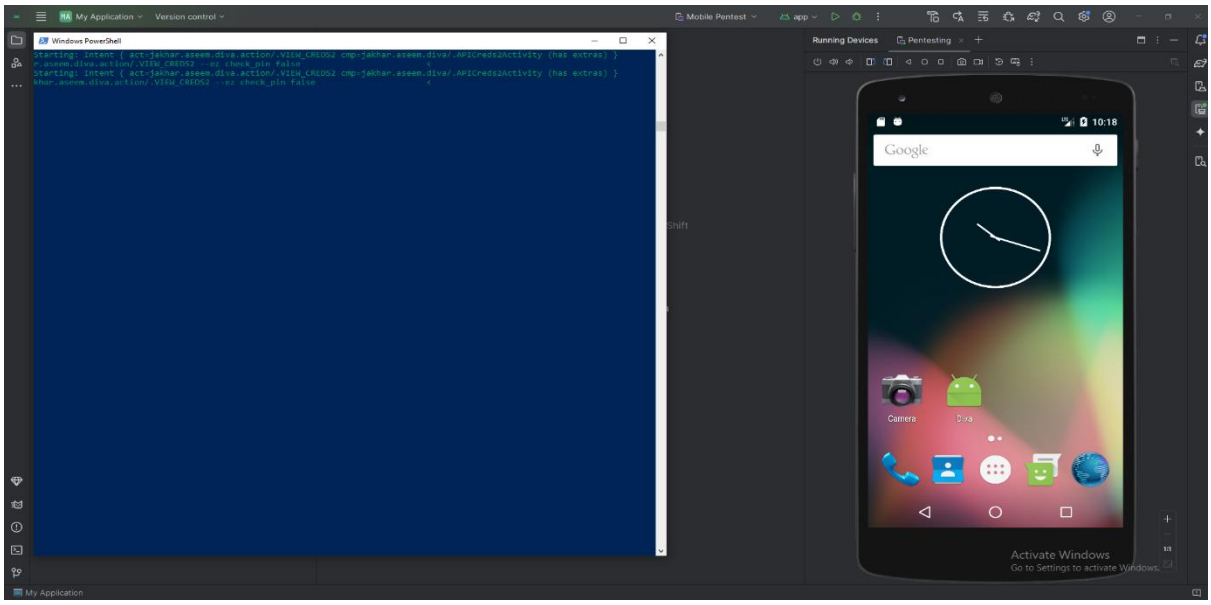


Screenshot-4: Respective activity & its source code. We can see that before start 2nd Activity (VIEW CREDS2) one chk pin validation point is there.
Also, find the chk pin string value (check_pin) from strings.xml file.



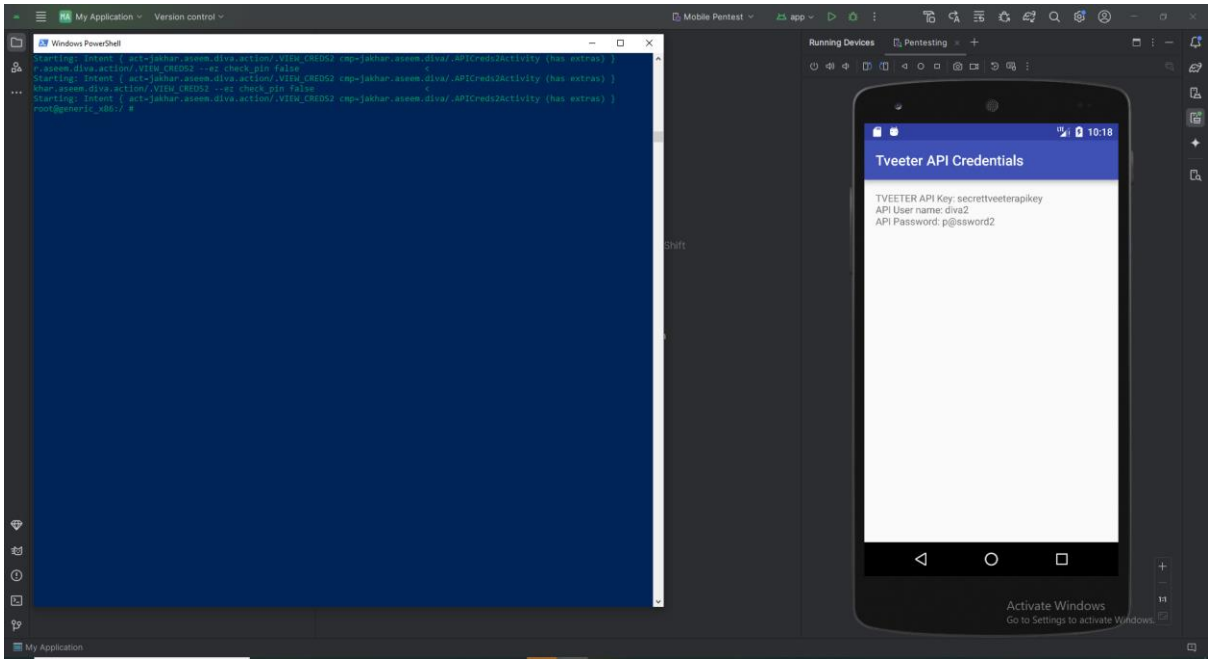


Screenshot-5: Now the DIVA app is closed.



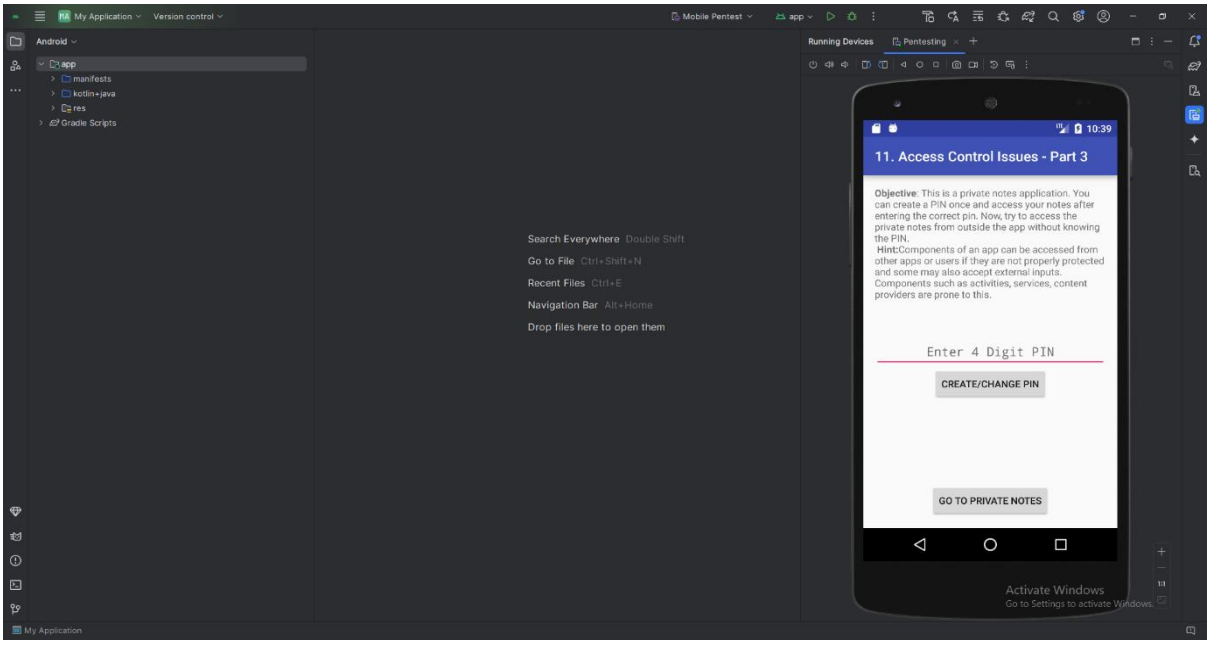
Screenshot-6: Now from adb shell try to execute the command
`am start -n jakhar.aseem.diva/.APICreds2Activity -a jakhar.aseem.diva.action/.VIEW_CREDS2 --ez check_pin false`
and after entering hit API Credentials are exposing on the emulator without validating any authorization.

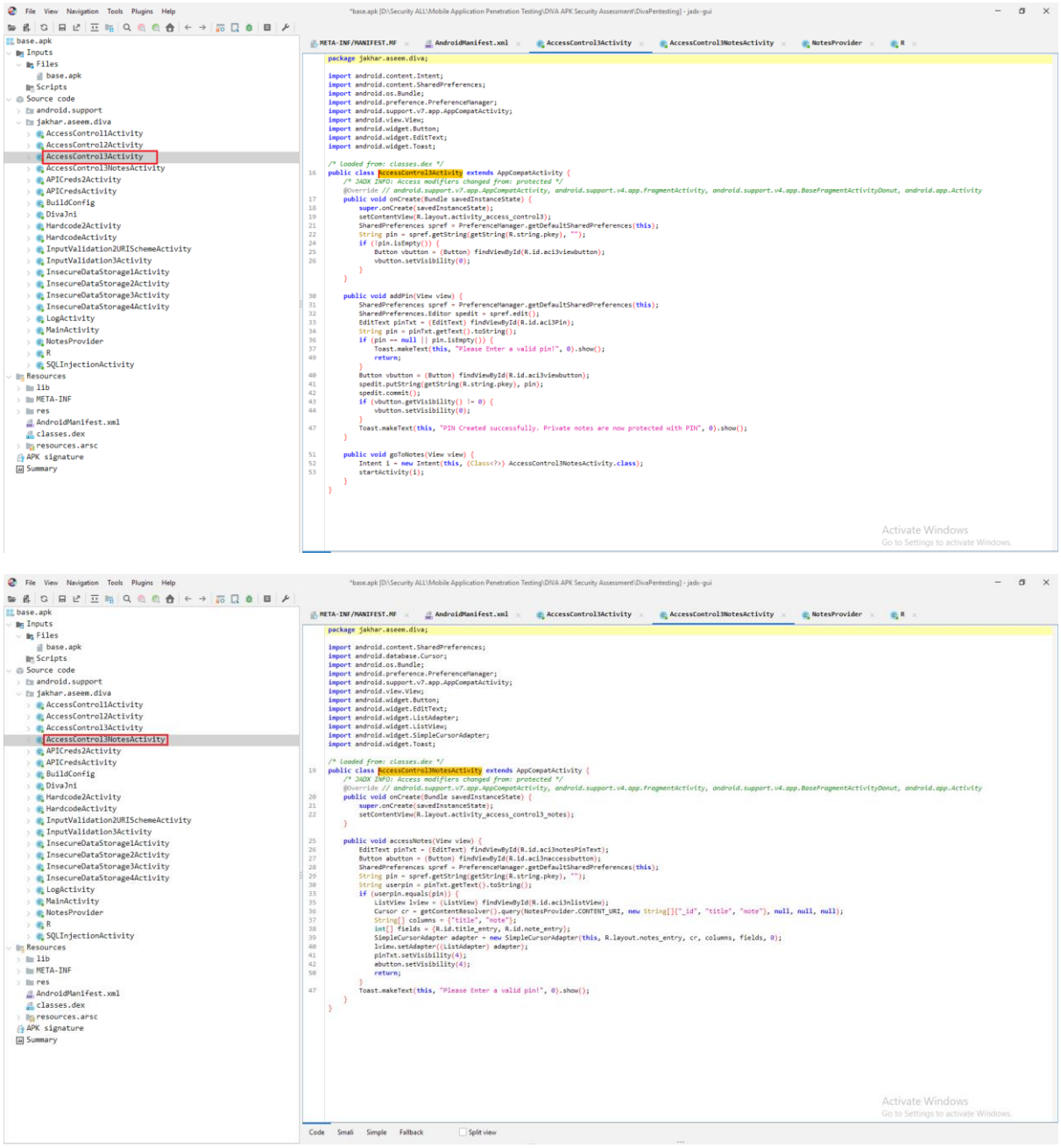
`am` = Activity Manager
`-n` = Do not check whether I am verified or not.
`-a` = Start an activity
`--ez check_pin false` = Value of the extra variable (check_pin) make as false as we don't know the pin.



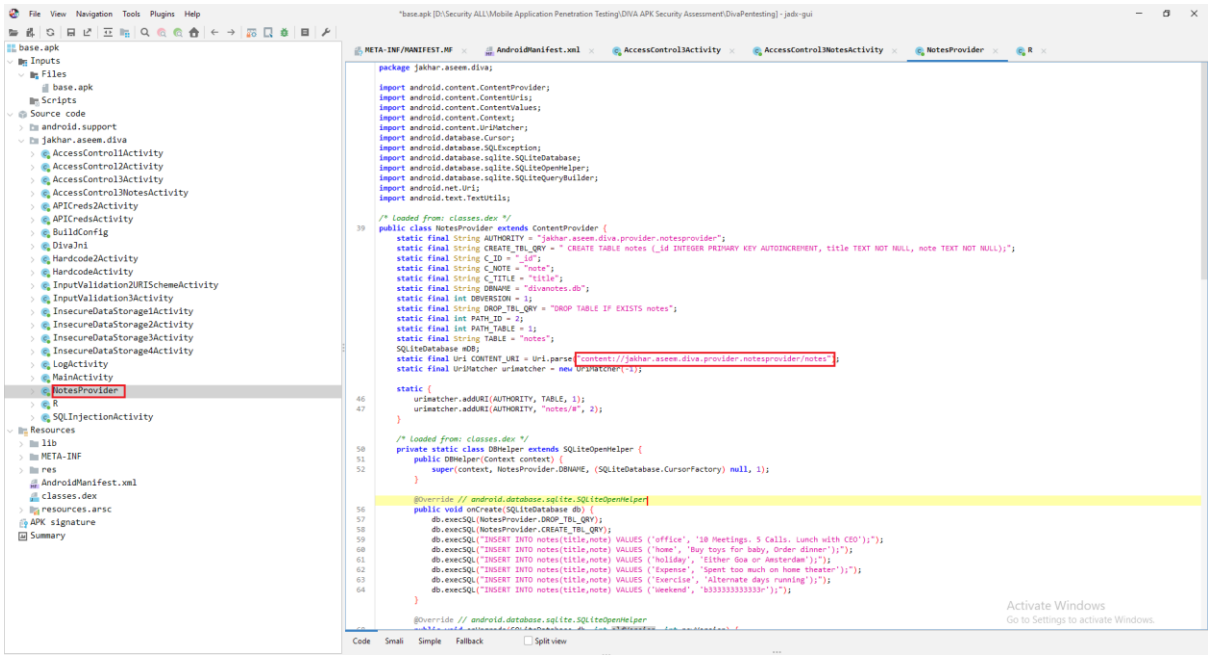
11. Access Control Issues_Part-3

Screenshot-1: The 1st Activity is **AccessControl3Activity** which is responsible for creation of the PIN. The 2nd Activity is **AccessControl3NotesActivity** which is responsible to store the content. **NotesProvider** give us content.



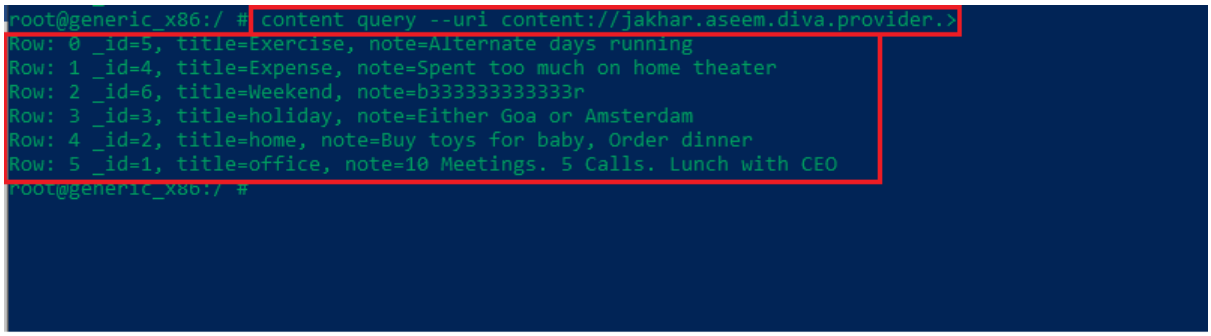


Screenshot-2: `NotesProvider` give us content. Now copy the Content URI from here.



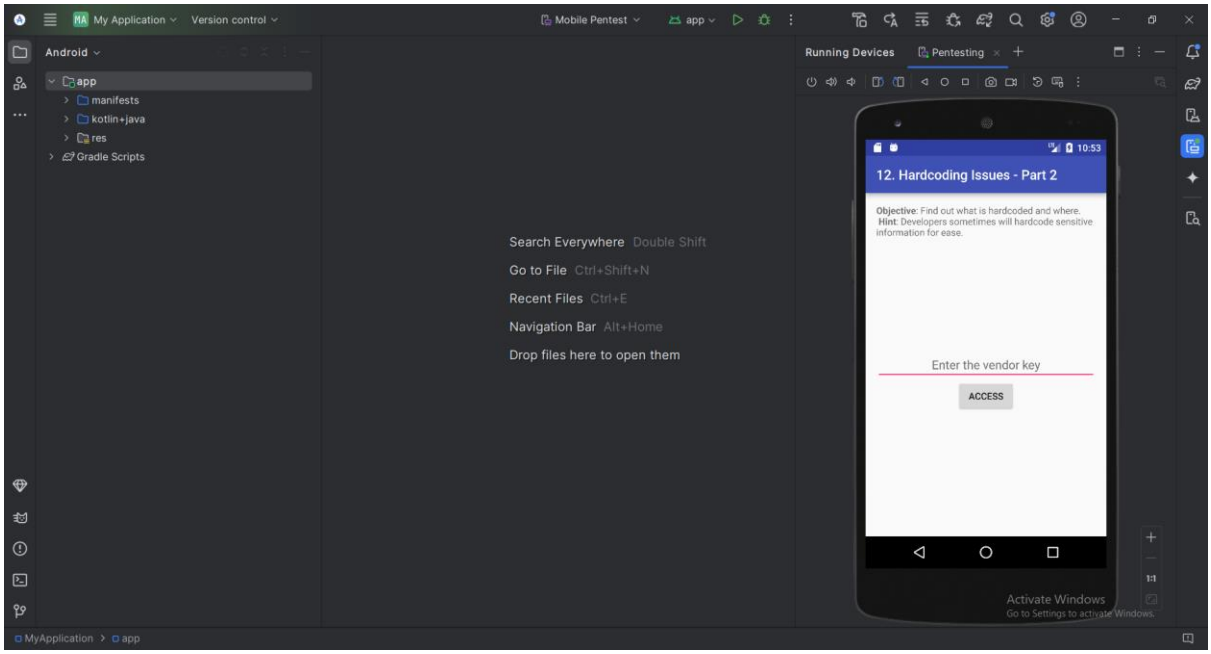
Screenshot-3: Now from adb shell try to execute the below command and fetch all details.

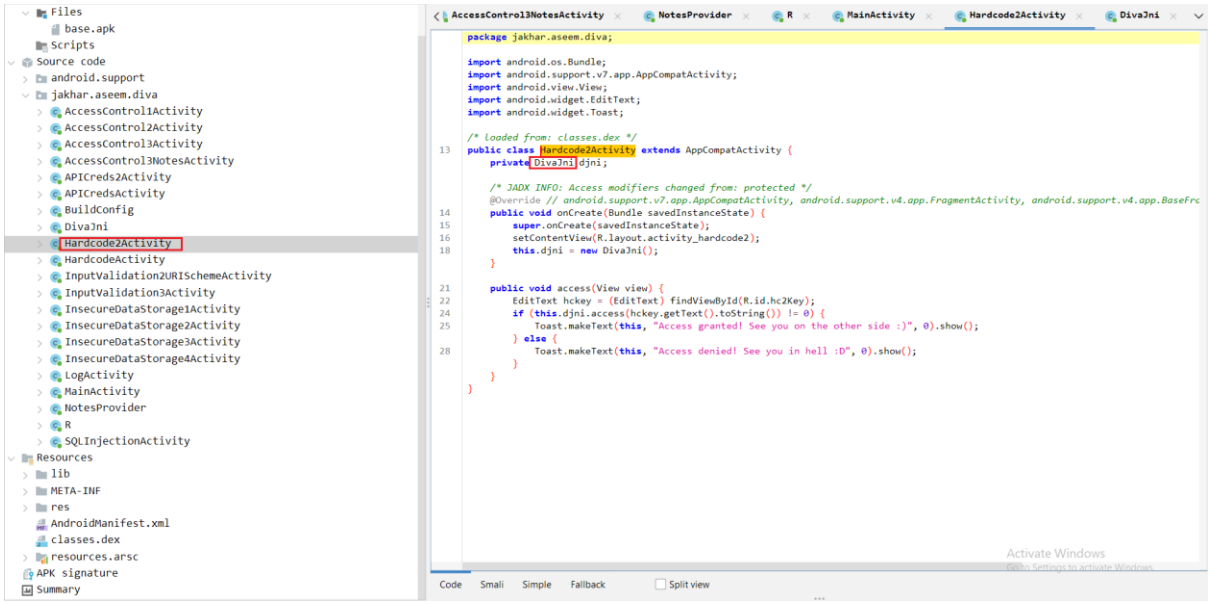
content query --uri `content://jakhar.aseem.diva.provider.notesprovider/notes`



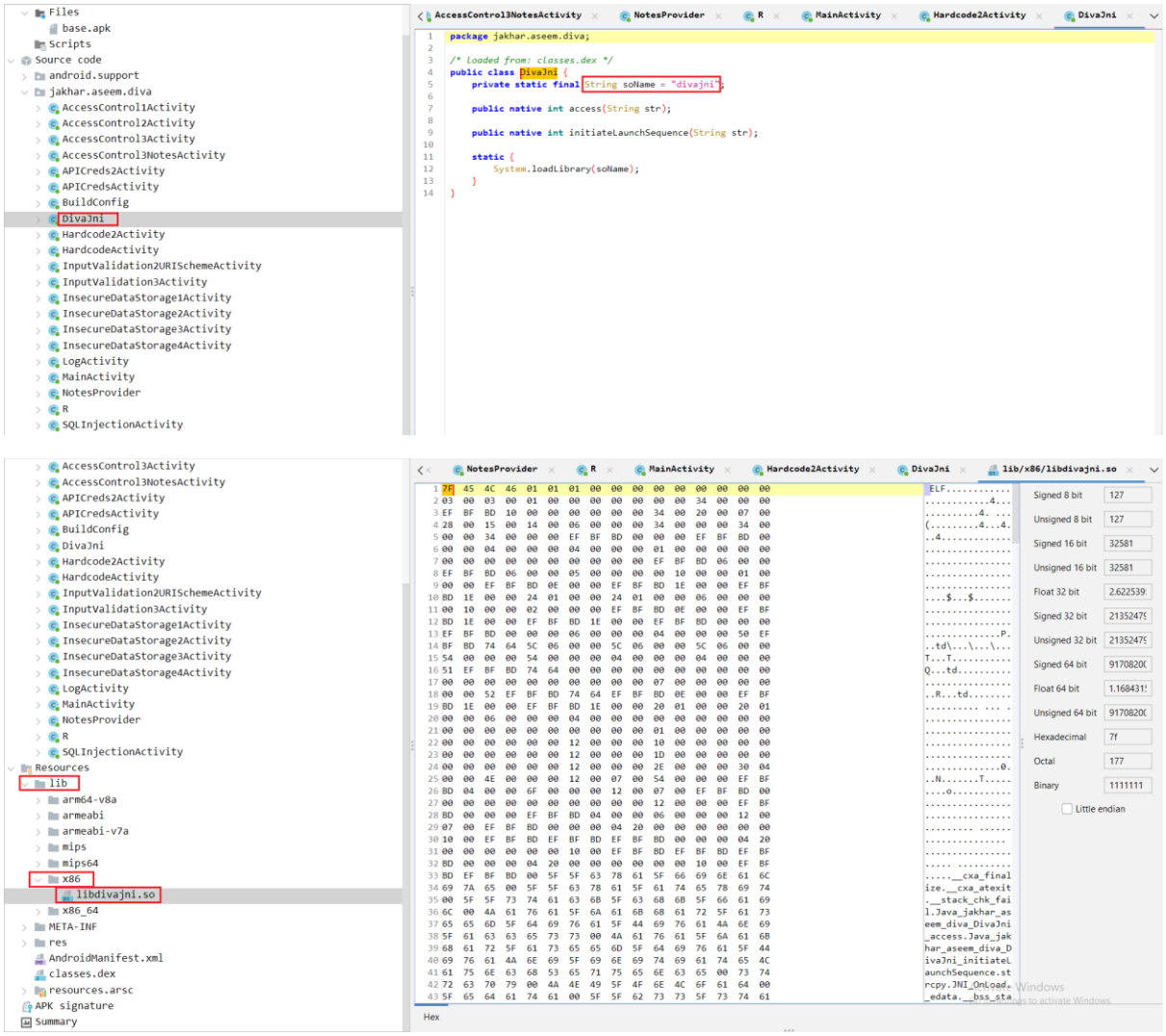
12. Hardcoding Issues_Part-2

Screenshot-1: The 1st activity **Hardcode2Activity** is responsible for entering the vendor key. We are getting 2nd activity name **'DivaJni'** from the source code of the **Hardcode2Activity.**





Screenshot-2: The source code of the 2nd activity **`DivaJni`** `provide the string value soName = “divajni”. We navigate the file` **`libdivajni.so`** `and found as compiled content.`



Screenshot-3: Decompile the **base.apk** file by using below command and found **base** directory.

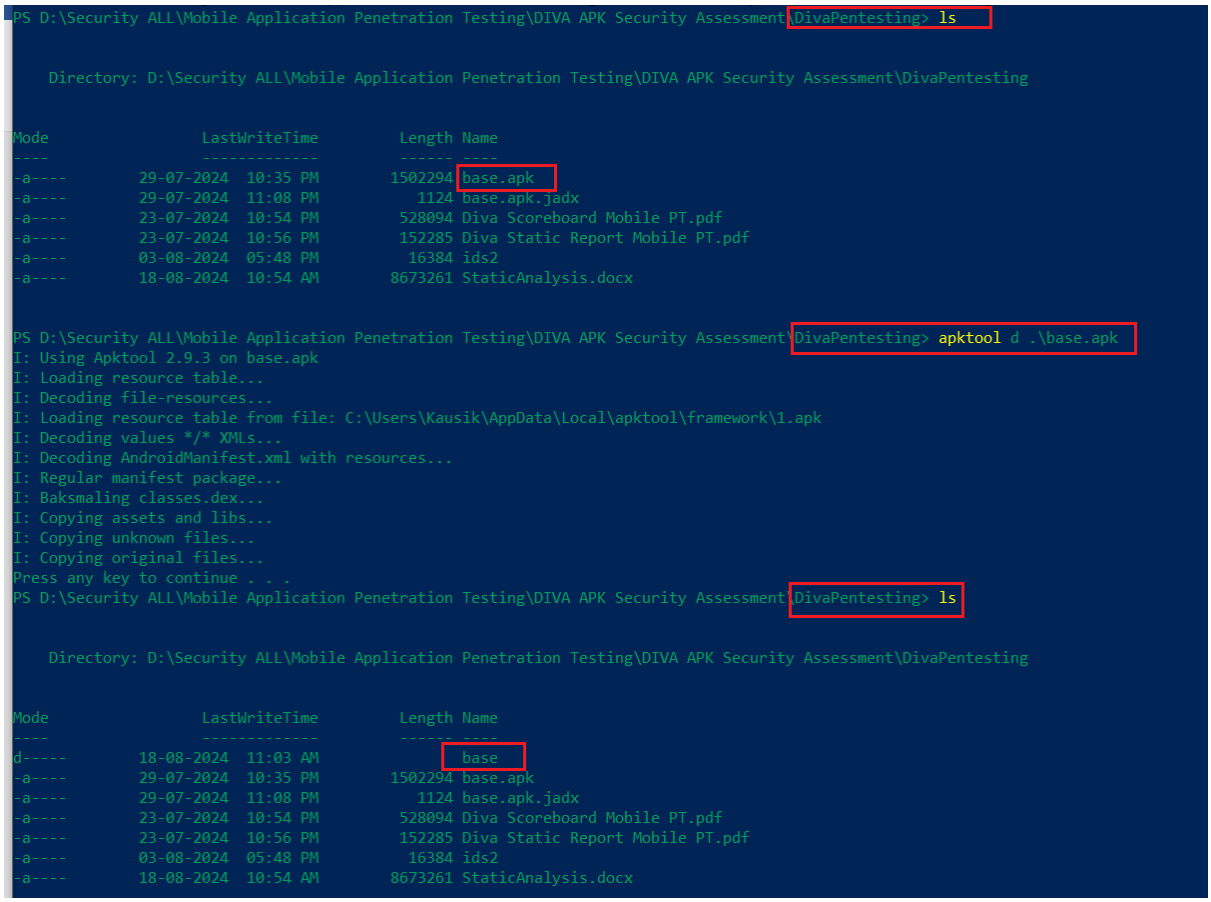
apktool d .\base.apk

Now navigate the below path:

base\lib\x86\libdivajni.so

Now see the all vendor key value from **libdivajni.so** by using bellow command**:**

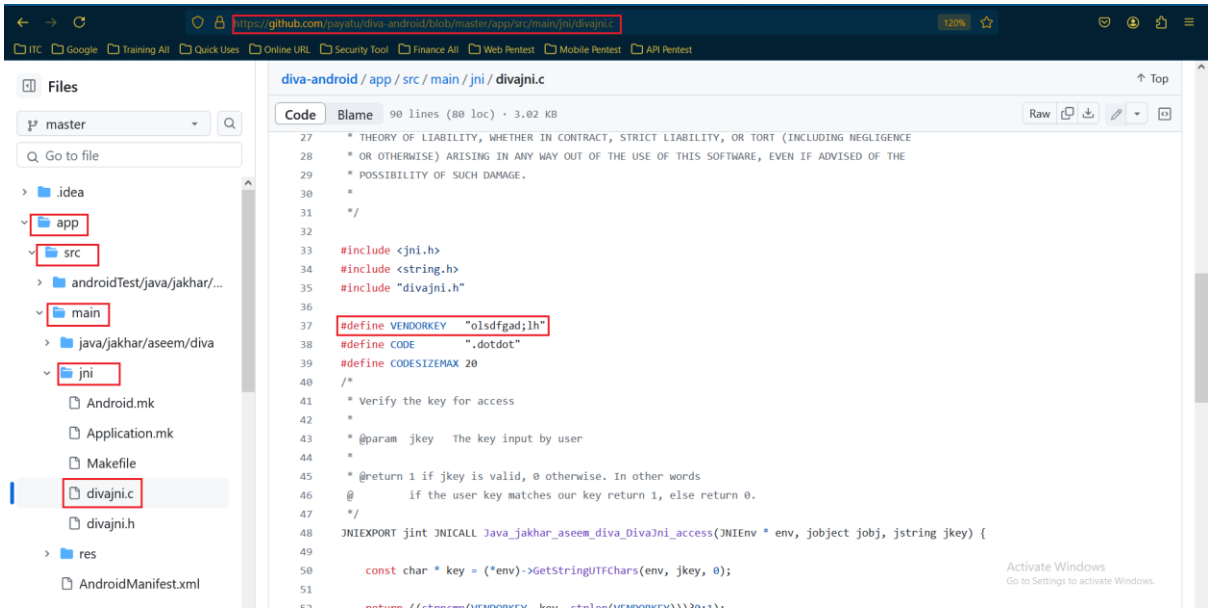
strings .\libdivajni.so

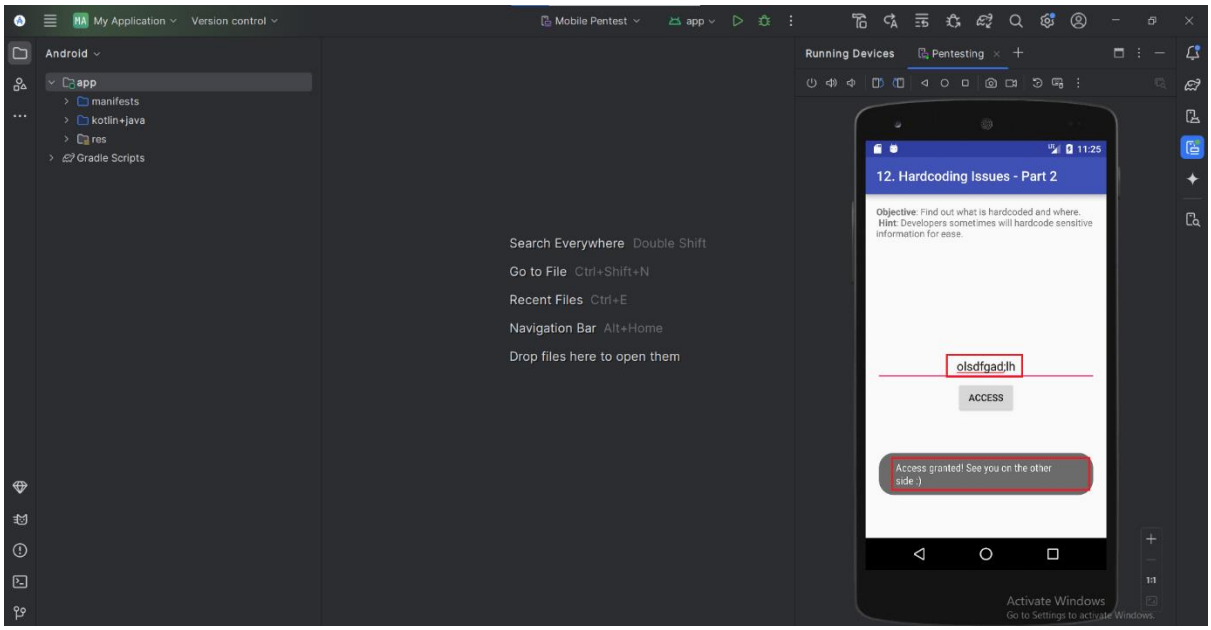


```
PS D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting> cd .\base
PS D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting\base> ls


    Directory: D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting\base


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        18-08-2024  11:03 AM                lib
d-----        18-08-2024  11:03 AM                original
d-----        18-08-2024  11:03 AM                res
d-----        18-08-2024  11:03 AM                smali
-a----        18-08-2024  11:03 AM           3390 AndroidManifest.xml
-a----        18-08-2024  11:03 AM            400 apktool.yml


PS D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting\base> cd .\lib
PS D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting\base\lib> ls


    Directory: D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting\base\lib


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        18-08-2024  11:03 AM                arm64-v8a
d-----        18-08-2024  11:03 AM                armeabi
d-----        18-08-2024  11:03 AM                armeabi-v7a
d-----        18-08-2024  11:03 AM                mips
d-----        18-08-2024  11:03 AM                mips64
d-----        18-08-2024  11:03 AM                x86
d-----        18-08-2024  11:03 AM                x86_64


PS D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting\base\lib> cd x86
PS D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting\base\lib\x86> ls


    Directory: D:\Security ALL\Mobile Application Penetration Testing\DIVA APK Security Assessment\DivaPentesting\base\lib\x86


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        18-08-2024  11:03 AM           5164 libdivajni.so
```

```
PS D:\divaPentesting\base\lib\x86> strings .\libdivajni.so
__cxa_finalize
__cxa_atexit
__stack_chk_fail
Java_jakhar_aseem_diva_DivaJni_access
Java_jakhar_aseem_diva_DivaJni_initiateLaunchSequence
strcpy
JNI_OnLoad
_edata
__bss_start
_end
libstdc++.so
libm.so
libc.so
libdl.so
libdivajni.so
d$0[
olsdfgad;lh
.dotdot
;*2$"
GCC: (GNU) 4.8
gold 1.11
.shstrtab
.dynsym
.dynstr
.hash
.rel.dyn
.rel.plt
.text
```

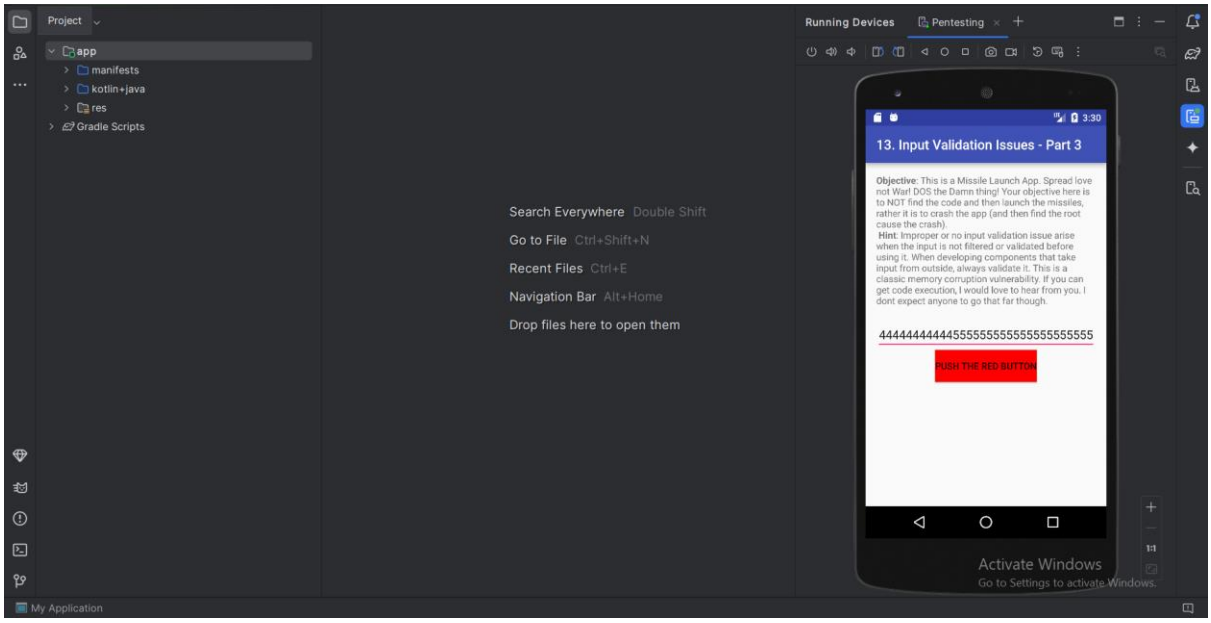
Screenshot-4: Either brute force all Vendor Key value or recon the same and then use it.





13. Input Validation Issues_Part-3

Screenshot-1: Here it could observe that improper input validation as there have no limit of input value. We are entering a large no of input and hit the red button.



Screenshot-2: After entering the large no of input the DIVA app has stopped.