Drozer Installation and Audit with Diva

To install Drozer open Kali terminal and use :

```
pipx install drozer
```

Next, Test if its install correctly using :

```
drozer
```

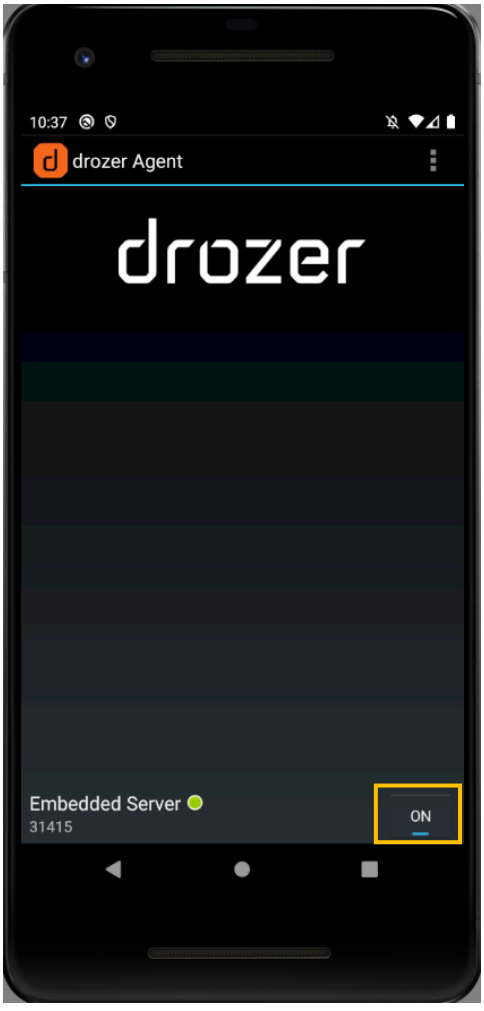
In terminal, and then download **drozer-agent.apk** from the repo release:

https://github.com/WithSecureLabs/drozer-agent/releases/

And install it using ADB into an Android Emulator running on Android Studios :

```
adb install drozer-agent.apk
```

After that there will be an APK file that will appear inside emulator, open it and start the server :

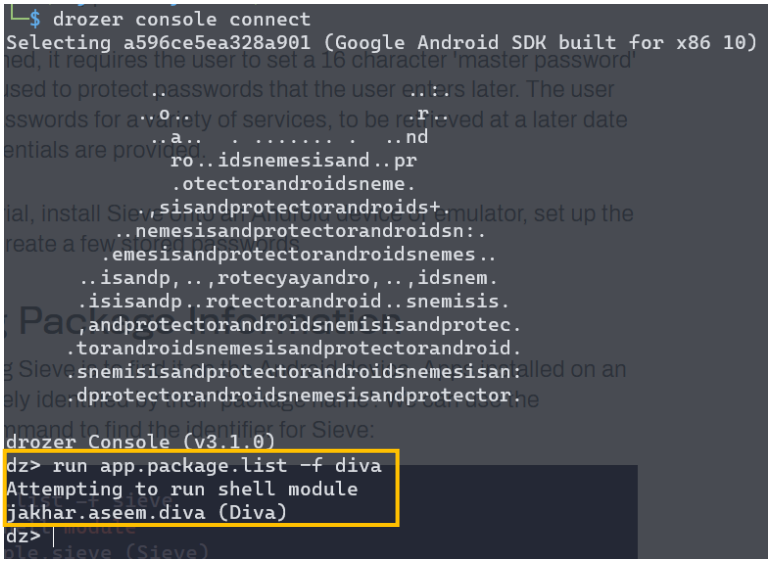


Then, go into the kali shell and use commands :

```
adb forward tcp:31415 tcp:31415
```

```
drozer console connect
```

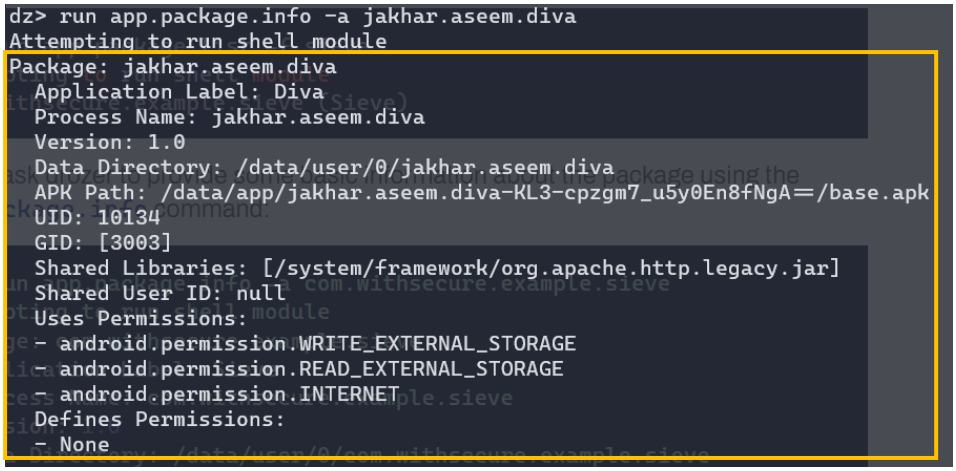
To connect to the drozer agent currently running on the device, now everything prepared for auditing.



Once connected to the android drozer console using the installed drozer agent, it will show something like what shown in the above figure. Then find the Diva APK using command :

```
run app.package.list -f diva
```

This shows an output "***jakhar.aseem.diva***" which confirms the Diva's installations.



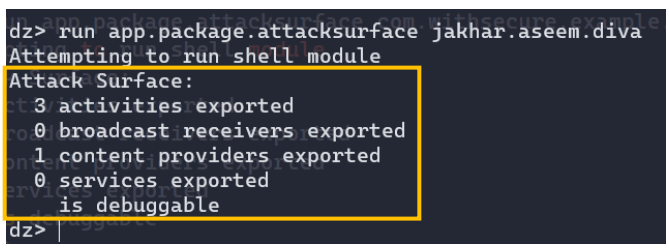
Now, run :

```
run app.package.info -a jakhar.aseem.diva
```

To see all the details (as shown in the above figure) about the package like application version, data directory where it stores app data, its installation directory, and also permissions it uses.

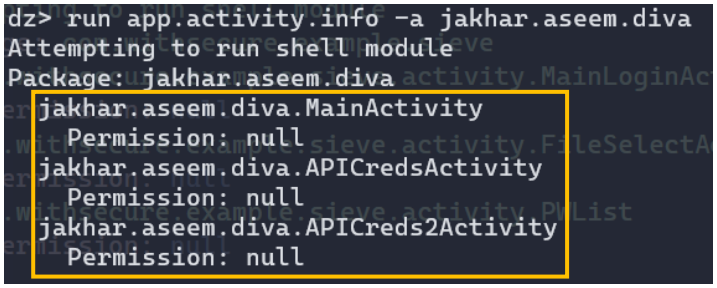
Next, identifying the attack surfaces using command :

```
run app.package.attacksurface jakhar.aseem.diva
```



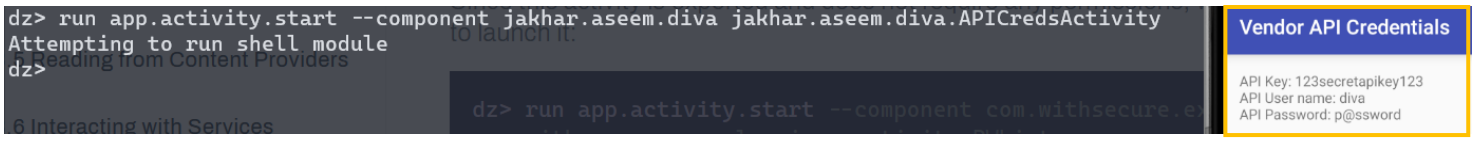
From the above figure, it has inferred that there are 3 activities and 1 content providers exported as attack surface in the application. Let's inspect on the activities for more details using command :

```
run app.activity.info -a jakhar.aseem.diva
```



In the output there are as expected three activities which has permissions as **null**, also 'MainActivity" because it is the screen displayed when the application was first launched. There are two other which are less expected and doesn't require any permissions too, so lets try launching one of those activity using drozer cli with command :

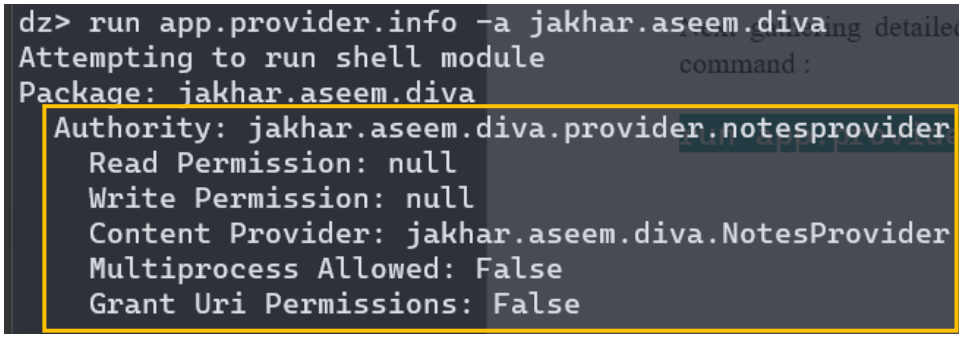
```
run app.activity.start --component jakhar.aseem.diva
jakhar.aseem.diva.APICredsActivity
```



In the Diva Android Application, there is an interface update and it shows Vendor API Credentials and the credentials stored for Vendor API, this happened because drozer utilises an appropriate **intent** in the background, and delivers it to the system through **startActivity** call, successfully bypassing the authorisation and presented a list of the API credentials.

Next gathering detailed information about the content provider exported by the application using command :

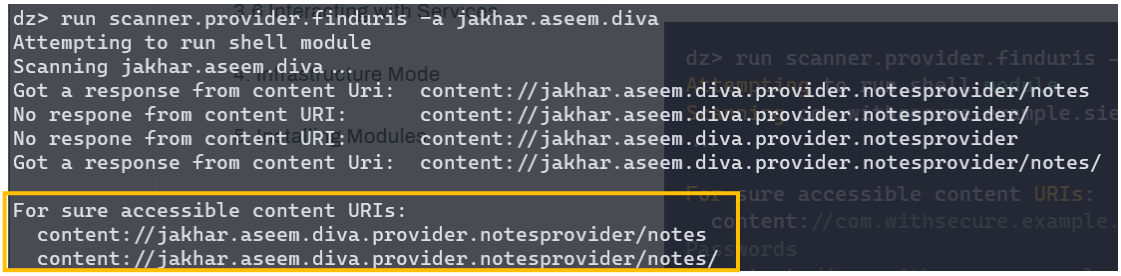
```
run app.provider.info -a jakhar.aseem.diva
```



Here, it can be inferred (from the above figure) that there is no read or write permissions required to interact with the content provider. Lets scan the provider that has various ways to guess paths, using command :

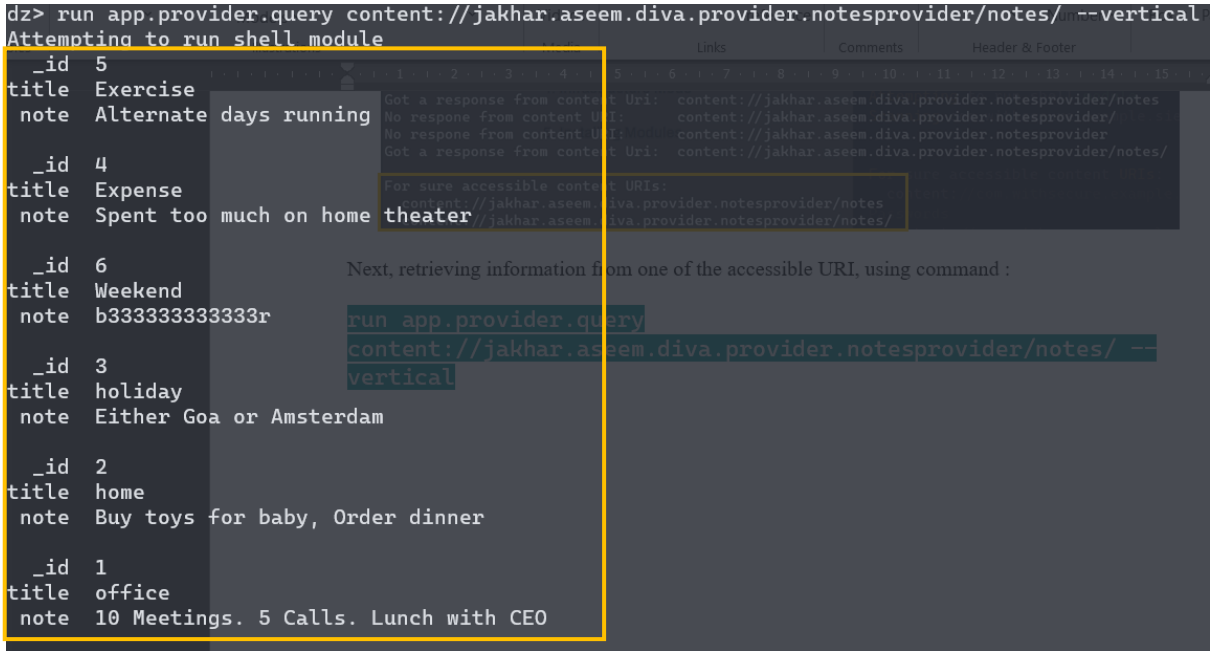
```
run scanner.provider.finduris -a jakhar.aseem.diva
```

This will show a list of accessible URIs and those URIs can be used to retrieve information or modify the data it stores.



Next, retrieving information from one of the accessible URI, using command :

```
run app.provider.query
content://jakhar.aseem.diva.provider.notesprovider/notes/ --
vertical
```



Here, the output clearly shows the data stored is in plain text inside the **notesprovider/notes/** of the content provider URI.

So, there are the steps and commands to follow in order to perform auditing on an android application like DIVA (Damn Insecure & Vulnerable Application).

Drozer also provides more commands like :

- `run app.provider.read` : To **read** files at content provider URIs
- `run app.provider.download`: To **download** files available at content provider URIs
- `run scanner.provider.injection`: To automate test **injections** on content provider URIs (Both SQLi & Directory traversal) for vulnerable URIs.
- `run app.service.info`: To **find information** of an application services and their required permissions to run.

These commands can be used to perform further auditing process.