Navaneetan M mr.navaneetan@gmail.com

Android VAPT – Learning Resource

Navaneetan M
mr.navaneetan@gmail.com

Navaneetan M mr.navaneetan@gmail.com

Table of Contents

Android VAPT – Learning Resource..... 1

Application Installation:..... 3

1. Insecure Logging vulnerability..... 5

2. Hardcoded Credentials Vulnerability..... 7

3. Insecure Firebase Database..... 10

4.Insecure shared preference..... 11

5. SQL injection Vulnerability..... 13

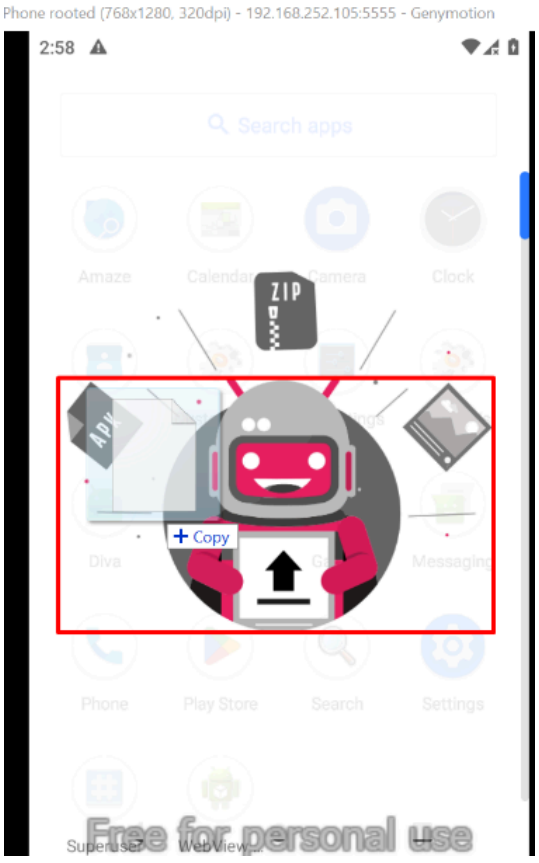
Navaneetan M mr.navaneetan@gmail.com

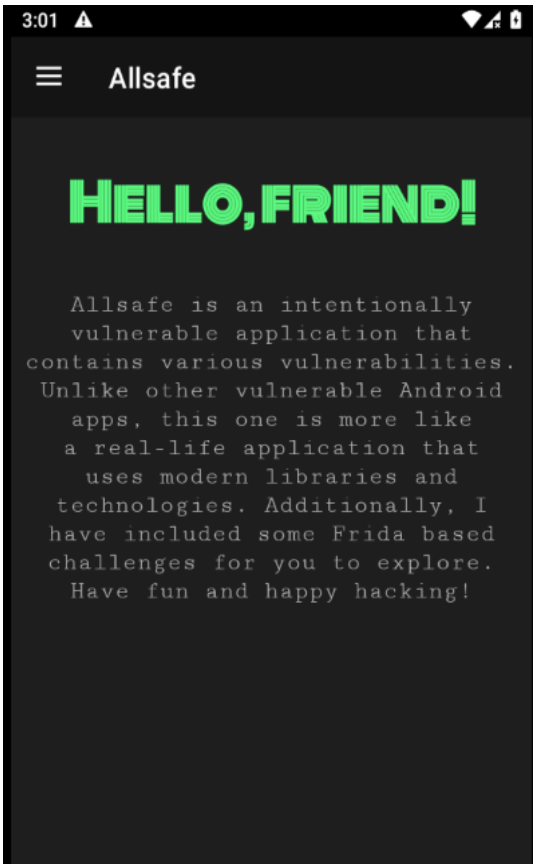
Allsafe android application walkthrough

Allsafe is an intentionally vulnerable apk application to practice common vulnerabilities in android application. In this walkthrough we will see the step-by-step explanation on how the vulnerability is found.

Application Installation:

- To install the vulnerable apk go to the website and download the apk file: https://github.com/t0thkr1s/allsafe/releases
- Drag the file into the emulator (Genymotion), now we are ready to test the application.





- Make sure adb is installed. ADB (Android Debug Bridge**)** is a versatile command-line tool used in Android penetration testing for interacting with Android devices or emulators

```
~ sudo apt install adb
[sudo] password for navaneeto:
Sorry, try again.
[sudo] password for navaneeto:
adb is already the newest version (1:34.0.5-6).
```

- Connect your emulator with the terminal.

```
~ adb connect 192.168.252.105:5555
connected to 192.168.252.105:5555
```

- Verify the connection.

```
~ adb devices
List of devices attached
192.168.252.105:5555    device
```

- Install jadx-gui, this tool decompiles the apk file and helps in SAST.

```
! ~ sudo apt install jadx
jadx is already the newest version (1.5.1-0kali1).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
```

- Now we are all set to start our testing with Allsafe.

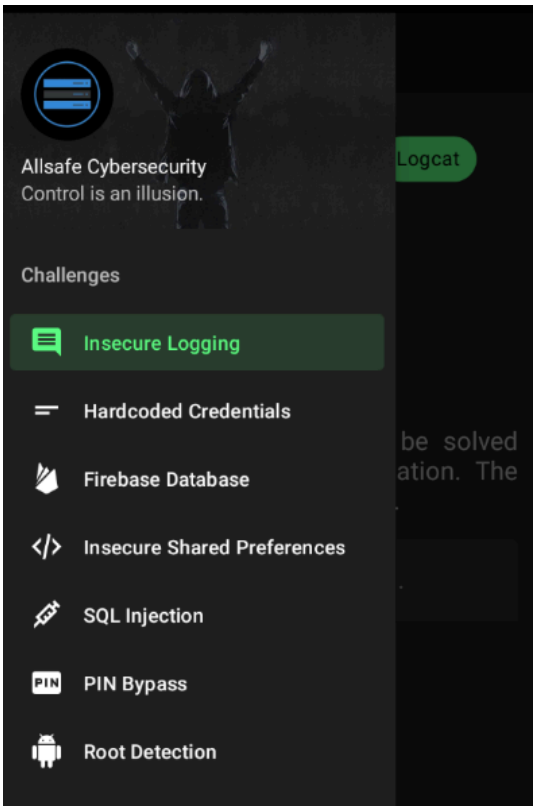
Navaneetan M mr.navaneetan@gmail.com

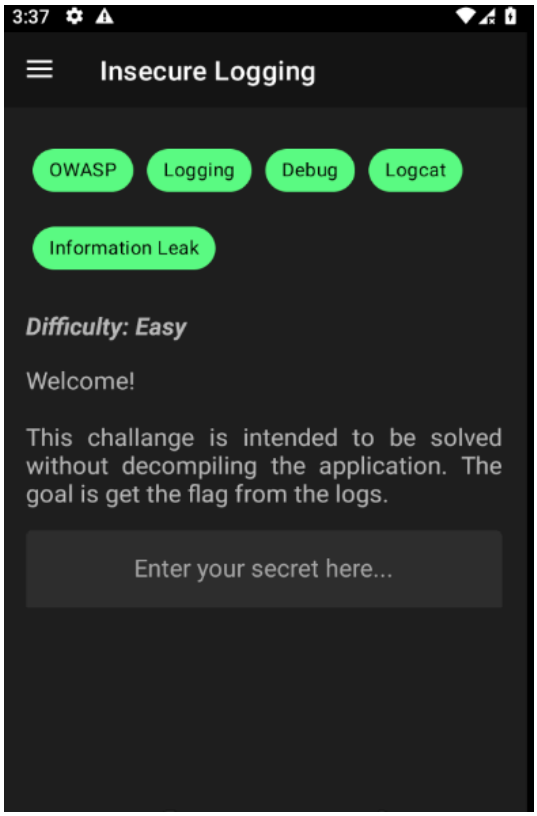
1. Insecure Logging vulnerability

Developers normally use logging to trace their code, and troubleshoot errors but, sometimes developers write sensitive data in the logs such as login credentials or authentication tokens.
To start testing for Insecure Logging in the app you need to make sure that you are connected to the device via adb then use logcat to monitor the device logs.

Steps:

- Go to the Insecure logging section

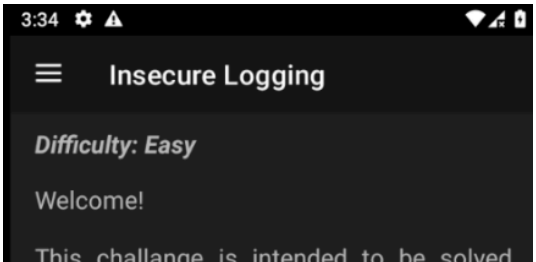




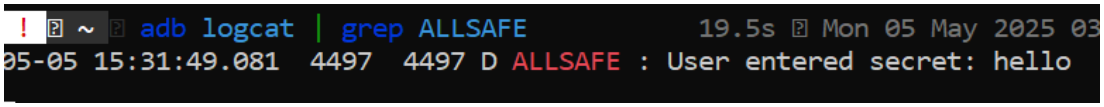
- Enter any character in the “Enter your secret here” parameter. Using the “adb logcat | grep allsafe” command we should get the logs that contain the details we entered.

  - **Note**: We use virtual keyboard to enter these details.

- Use the virtual keyboard given in the application and hit the enter button, this time we notice that the previous adb logcat command works.



Navaneetan M mr.navaneetan@gmail.com

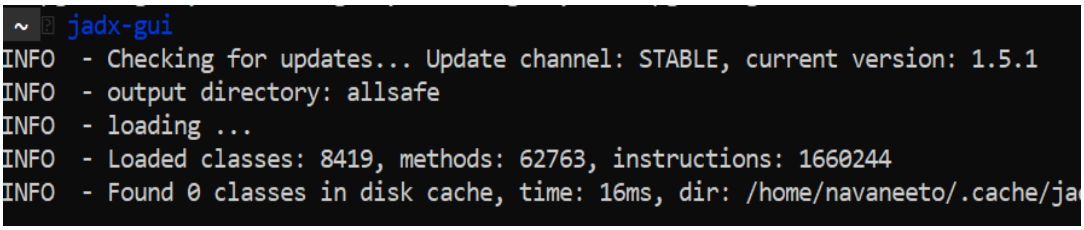


Navaneetan M mr.navaneetan@gmail.com

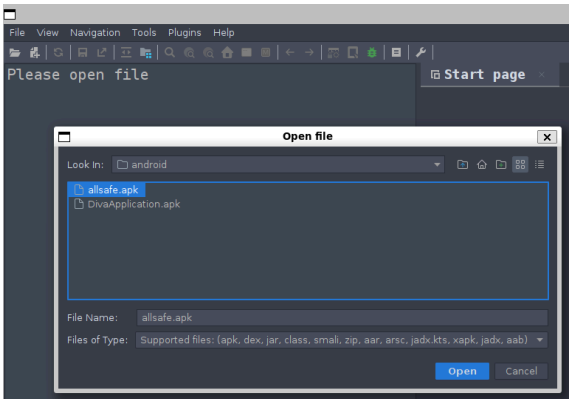
2. Hardcoded Credentials Vulnerability

In this task we will be finding two sets of hardcoded username and passwords that is stored in the source code, for this challenge we will need the jadx-gui tool.

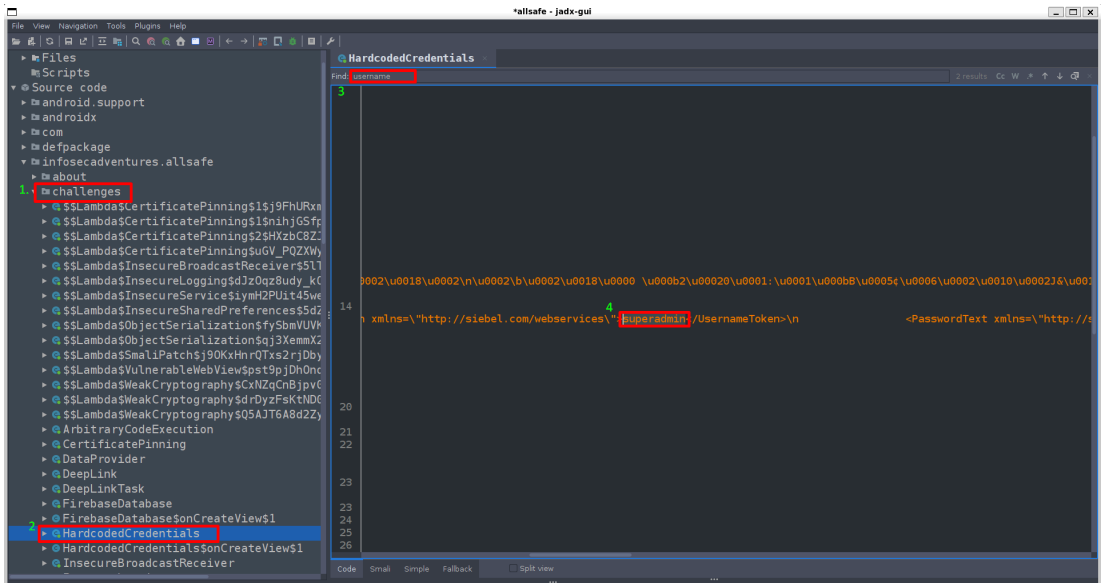
- Open the jadx-gui tool using the “jadx-gui” command, you should see a pop-up which opens a blank editor.



- In the left corner select “file”, click on “open file” and select the allsafe application that is stored in your system, if file not found copy and paste the file to the respective system which has jadx-gui.

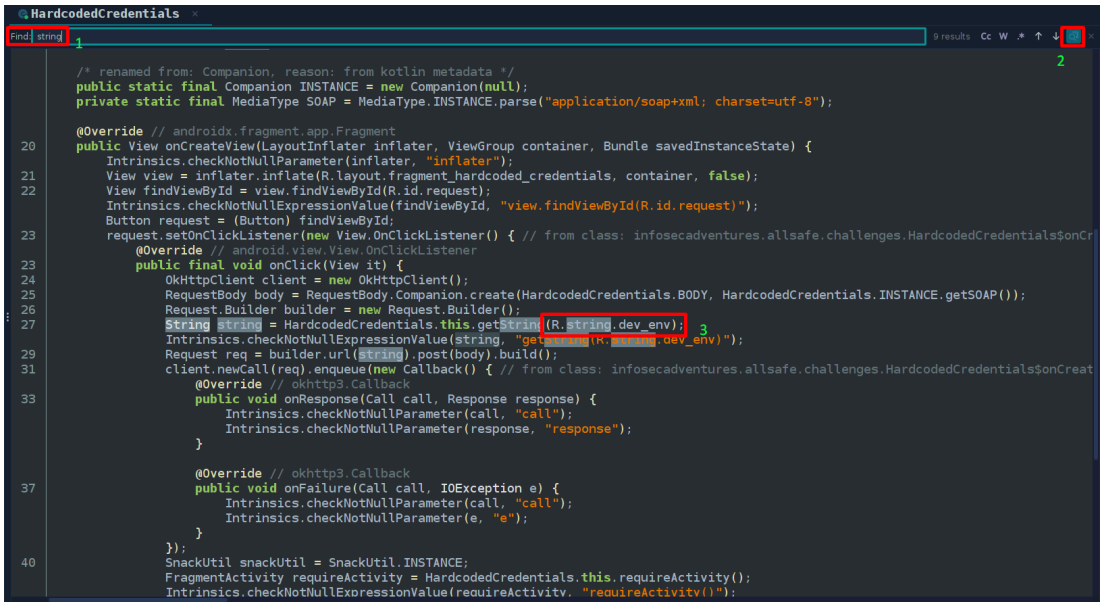


- Now to find the hardcoded credentials, go to the source code and select the challenges drop down.
- In the challenges part you will find the “HardcodedCredentials” section.
- Press “Ctrl + F” and search “Username” now in the username token section we find the username.

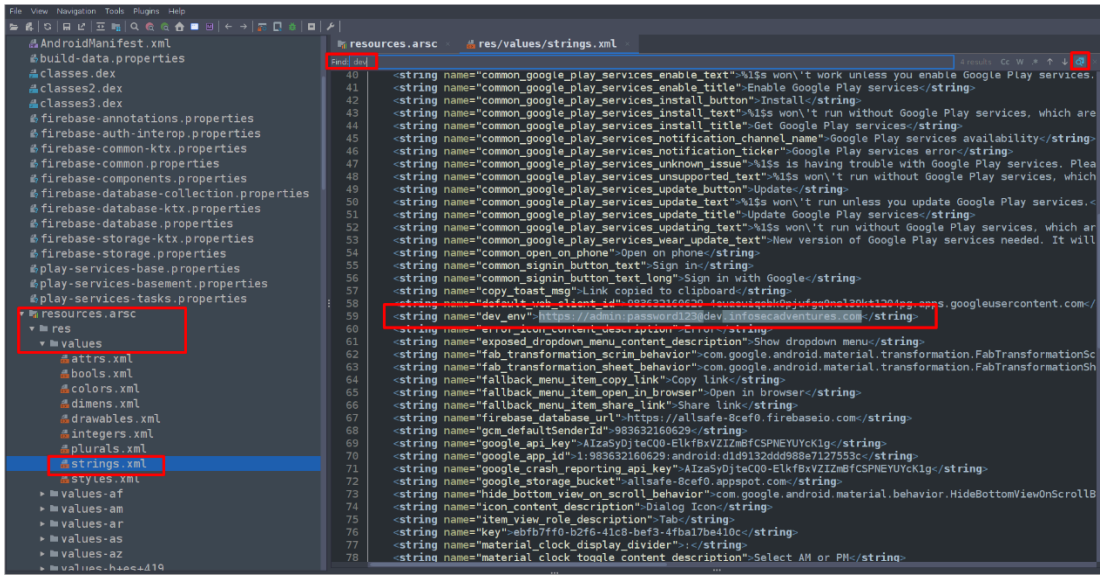


- To find the password, in the search field type “password” and you can see the password as shown below.

- First set of credentials have been found, now for the second set, we search the same file for “string”. From the below image, we notice that the file fetches a GET request that gets a “dev.env” string.



- To find the dev.env string, navigate to resources/resource.arsc/res/values/strings.xml. And search for “dev” or “env”.



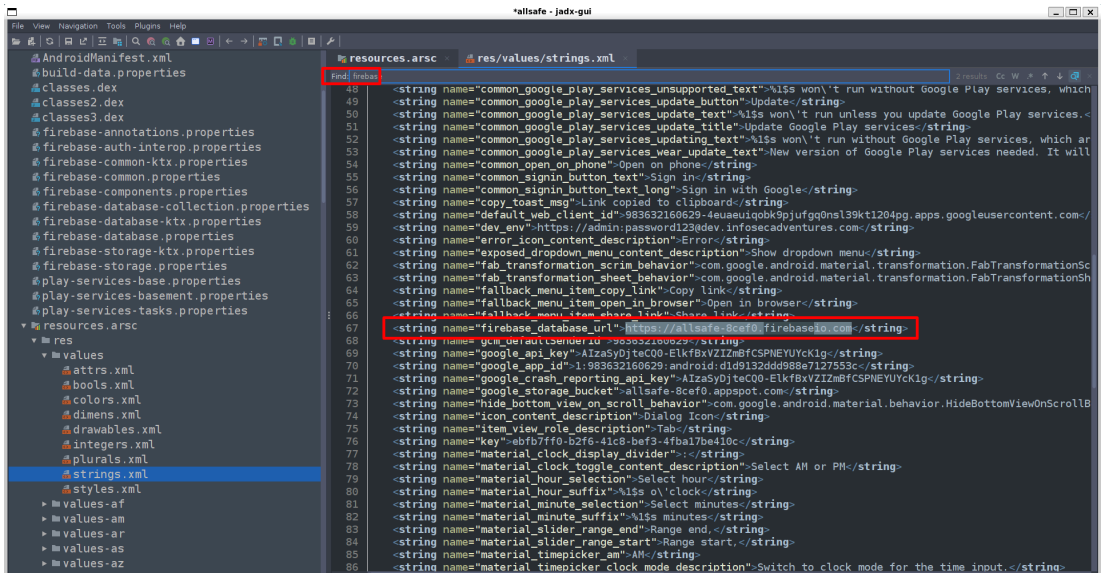
Navaneetan M mr.navaneetan@gmail.com

3. Insecure Firebase Database

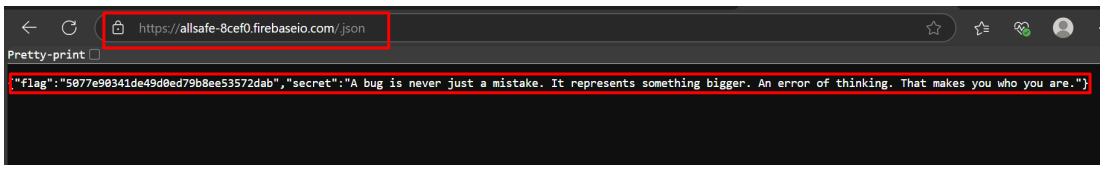
Firebase is a Realtime cloud-based NoSQL database that can be integrated with both Android and iOS apps.

To start our testing we first need to find the firebase URL.

- In jadx navigate to /Resources/resources.arsc/res/values/strings.xml and search for firebase to locate it.



- Next, from your browser try to access the /.json endpoint. If you get a Permission Denied response that means that the database is properly configured but if get a json data or even null as a response that means you at least have read permission.

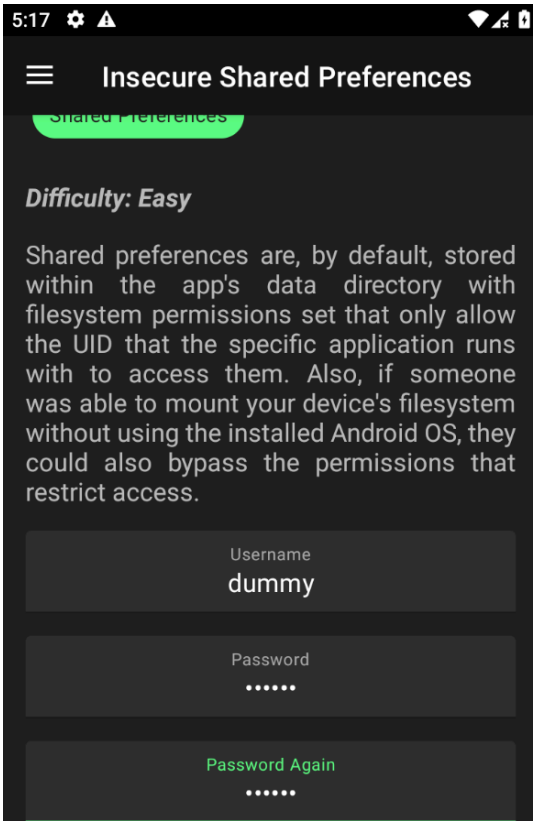


Navaneetan M mr.navaneetan@gmail.com

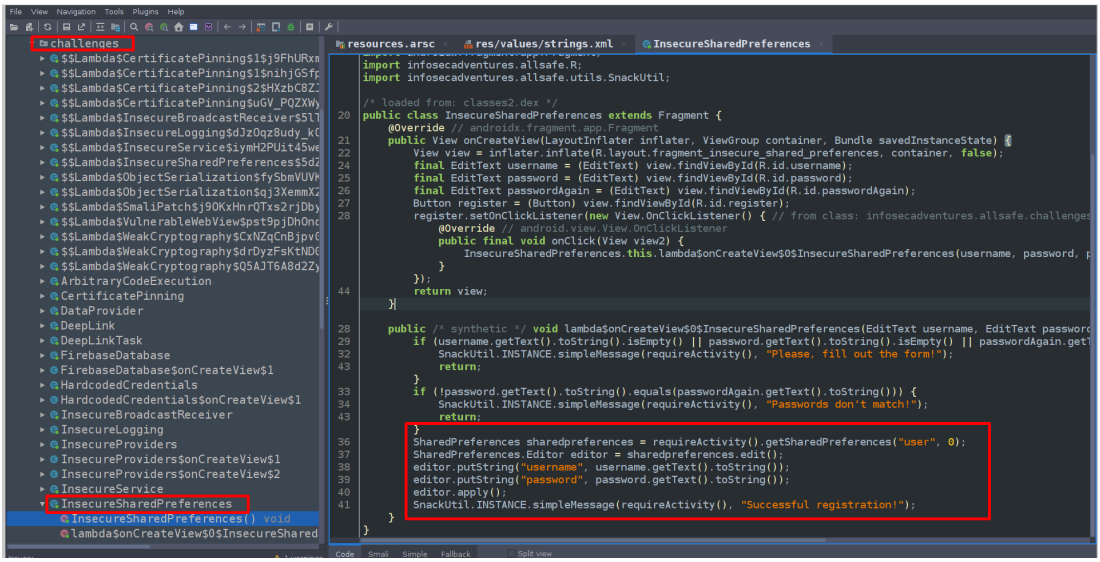
4.Insecure shared preference

Shared Preferences is the way in which one can store and retrieve small amounts of primitive data as key/value pairs to a file on the device storage such as String, int, float, Boolean that make up your preferences in an XML file inside the app on the device storage.

- In short, Shared Preference is a place where key/value pairs are stored.
- To test this vulnerability, enter a username and a password in the Insecure Shared Preferences challenge.



- If you look at the code of this challenge, in the “challenge” section as we did before, you’ll see that getSharedPreferences("user", 0) creates a shared preference object with the default creation mode MODE_PRIVATE and is named user.xml. After that, the plaintext username and password of a registered user are stored in the xml file.



- This is a very bad practice because while on non-rooted devices, other apps normally can't read that xml file, it's not the same case on a rooted android device.
- You can confirm the existence of an Insecure Data Storage vulnerability by navigating to /data/data/infosecadventures.allsafe/shared_prefs/ and reading user.xml

```
genymotion:/data/data/infosecadventures.allsafe/shared_prefs # cat user.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">Hi@123</string>
    <string name="username">dummy</string>
</map>
genymotion:/data/data/infosecadventures.allsafe/shared_prefs #
```

Navaneetan M mr.navaneetan@gmail.com

5. SQL injection Vulnerability

SQL Injection is a security flaw in web applications where attackers insert harmful SQL code through user inputs. This can allow them to access sensitive data, change database contents or even take control of the system.

- SQL injections arise due to coder's lack of knowledge in prepared statements and exception handling. Maliciously simple codes like the " and "- -" can create chaos in the database. Just think of commenting out authentication mechanism and adding your own piece of authentication and leverage it to admin privileges. That's what happens when you enter: ' or 1=1 --

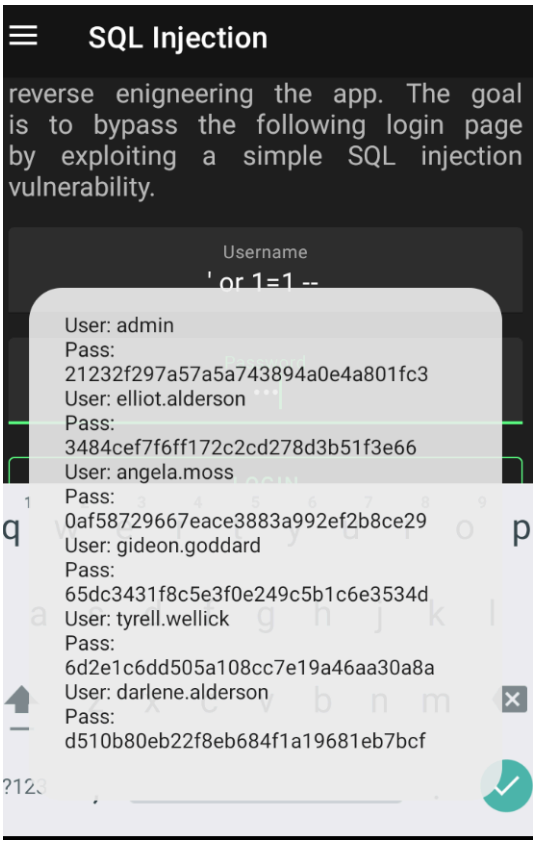
  ' - commenting the authentication query

  Or – making the query believe what you are going to enter next

  1=1 – braking the authentication, making the query believe that username and password match.

  "-- " commenting out rest of the query which fetches us the table containing username and passwords.

  So just enter "' or 1=1 -- " and you will see the username and password table.



- If you want to investigate it more, you'll notice that the username and password are both parts of a raw SQL query that lacks proper sanitization. Go to the challenges section as we went in the previous section and check for the SQL injection part.