CMC TELECOM

8th Floor, CMC Tower, 19 Street, Tan Thuan EPZ, District 7, Ho Chi Minh City

Tel: +842871090100| Fax:   
+84 28 3925 9755| [cmctelecom.vn](https://cmctelecom.vn/)

**SECURITY ASSESSMENT REPORT**

**ASSESSMENT SUBJECT: Insecure Bank V2**

*Ho Chi Minh City, 2023*

**CMC TELECOM**

|  |  |
| --- | --- |
| **Version** | 1.0 |
| **Target** | Insecurebankv2.apk |
| **Date** | 06/04/2023 |
| **Document Type** | Report |
| **Prepared By** | Tran Truong Giang |

**MỤC LỤC**

[**MỤC LỤC** 3](#_Toc131704209)

[**I.** **Overview** 4](#_Toc131704210)

[**1.** **Synopsis** 4](#_Toc131704211)

[**2.** **Method of implementation** 4](#_Toc131704212)

[**3.** **Classification of Vulnerabilities** 5](#_Toc131704213)

[**4. Scope of Work** 6](#_Toc131704214)

[**5. Summary of Testing Process** 7](#_Toc131704215)

[**II.** **Details of Implementation** 9](#_Toc131704216)

[**1.** **Application Information** 9](#_Toc131704217)

[**2.** **Summarized findings and Vulnerability Graph** 10](#_Toc131704218)

[**3.** **Vulnerability List** 11](#_Toc131704219)

[**1.** **Vulnerability details** 12](#_Toc131704220)

[**4.1** **Developer Login** 12](#_Toc131704221)

[**4.2 Weak Cryptography in data storage** 13](#_Toc131704222)

[**4.3  Insecure Logging** 16](#_Toc131704223)

[**4.4  Application Backup Enabled** 18](#_Toc131704224)

[**4.5  Bypassing Login Screen using Exported Activity** 21](#_Toc131704225)

[**4.6  Hidden Create User Button for Admins** 23](#_Toc131704226)

[**4.7  Root Detection Bypass** 28](#_Toc131704227)

[**4.8 Debug Mode Enabled** 33](#_Toc131704228)

[**4.9 Flawed Broadcast Receivers** 34](#_Toc131704229)

[**4.10 Insecure Content Provider Access** 37](#_Toc131704230)

[**4.11 Insecure WebView Implementation** 39](#_Toc131704231)

[**4.12 Parameter Manipulation** 42](#_Toc131704232)

[**4.13 Username Enumeration** 44](#_Toc131704233)

[**4.14 Insecure HTTP Connections** 48](#_Toc131704234)

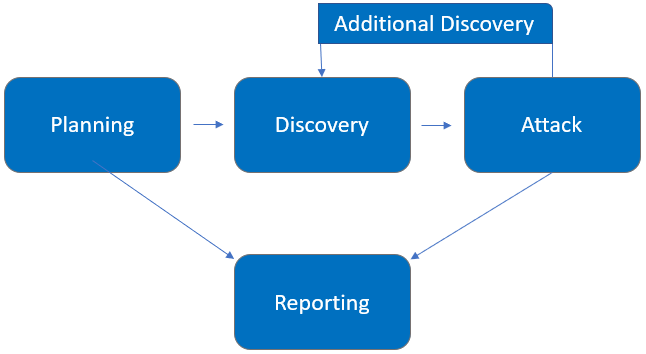
# **Overview**

## **Synopsis**

From March 07, 2023 – March 10, 2023 CMCCS and REDACTED had collaborated to conduct the penetration test for the app Insecurebankv2.apk. All tests follow the OWASP standards.

**The assessment procedure includes:**

* Planning: Identify the subject and scope for assessment
* Discovery: Test, Scan, Search and Identify intel relevant to the test subject including Versions, Vulnerabilities, Weakness, Sensitive data, etc
* Attack: CMC Personnel will perform attacks and exploits on discovered vulnerabilities.
* Reporting: Document the vulnerabilities along with the method of exploit recognised on the test subject, and recommendation for remedy.



## **Method of implementation**

Gray Box Pen-testing: In this method, the internal structure of the application is known partially (usually internal account or test account)

With Grey Box Pentesting, CMC will be provided with an internal account along with necessary information of the system to conduct the test.

Diagram

Description automatically generated

## **Classification of Vulnerabilities**

|  |  |  |
| --- | --- | --- |
| **CLASSIFICATION OF VULNERABILITY** | | |
| **Level** | **CVSS V3 Scoring** | **Description** |
| **Critical** | 9 – 10 | Vulnerabilities that allow hackers to attack from the outside in with the highest privilege, exposing sensitive or full data, impacting severely the information integrity (data is modified or completely erased) as well as its availability (all services are shut down) |
| **High** | 7 – 8,9 | Vulnerabilities allowing attackers to impact the system within a certain scope such as taking over user authority to access a device without authentication, exposing large amount of data (but have low level of sensitivity), data is subjected to modification and its integrity is affected, leading to the system being stalled of interrupted. However, the impact is not too severe to the reputation of the organisation and only affects a group of users |
| **Medium** | 4 – 6,9 | Vulnerabilities at this level is usually used as a predecessor for future attacks and exploits to potentially affect the system at a higher level. These types of vulnerabilities can cause nuisance for users but usually do not affect the availability of the service directly |
| **Low** | 0,1 – 3,9 | Vulnerabilities that leak data at a low level where said data are not valuable for exploits and does not affect the integrity of the information as well as the activities of the system. The fix is often feasible and easy with little to no cost. Organisations' reputation is not affected |

## **4. Scope of Work**

|  |  |  |
| --- | --- | --- |
| Method of Implementation | Test Object | Information provided |
| Gray Box  Pen-testing | App Insecurebankv2.apk | Redacted’s testing accounts |

## **5. Summary of Testing Process**

After discussing with REDACTED on ensuring the continual availability of the app Insecurebankv2.apk. CMC proposed conducting the penetration test from 27/03/2023 – 08/04/2023. The detail of work is as follow:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Contents of work performed** | | **Condition** |
| 1 | Collection  Of  Information | Determine the types of data connections that the app uses 3G, WiFi connection, NFC connection, Bluetooth. | **PASS** |
| The permissions that the app requires when installing. | **PASS** |
| Collect information about unfamiliar domains or IP connections in the application. | **PASS** |
| Collect information about the SDK if built into the app. | **PASS** |
| 2 | Static  Analysis | Evaluate the authentication mechanism. | **FAIL** |
| Check the anti-root, anti-vm, cert-pinning mechanisms (if any) of the application. | **FAIL** |
| Check the app’s permission configurations. | **PASS** |
| Check the configuration in the Manifest (Activity Hijacking) file. | **FAIL** |
| Check session management mechanisms and insecure cookie storage. | **PASS** |
| Check for sensitive information in logs, code, in directories or in sqlite. | **FAIL** |
| Check information about libraries, dependencies, and open source from 3rd parties. | **FAIL** |
| Data transport cascade assessment. | **PASS** |
| Evaluate the possibility of decompiling source code and tampering with applications. | **FAIL** |
| 3 | Dynamic  Analysis | Evaluate Web App issues related to the application: XSS, Command Injection, CRSF, SQL Injection, Cookies … | **FAIL** |
| Evaluation of the application’s encryption machanisms. | **FAIL** |
| Analyze files created during application installation. | **PASS** |
| Memory analysis. | **PASS** |
| Evaluation of authentication mechanisms. | **FAIL** |
| Evaluating the authorization mechanism. | **FAIL** |
| Evaluation of session management mechanisms. | **PASS** |
| Data transfer layer assessment. | **PASS** |
| Evaluate server-side attacks from the application. | **PASS** |

# **Details of Implementation**

## **Application Information**

|  |  |
| --- | --- |
| Platform | Android |
| Package Name | com.android.insecurebankv2 |
| Version | Android 11 |
| Min SDK | 15 |
| Target SDK | 22 |
| MD5 | 5ee4829065640f9c936ac861d1650ffc |
| SHA1 | 80b53f80a3c9e6bfd98311f5b26ccddcd1bf0a98 |
| SHA256 | b18af2a0e44d7634bbcdf93664d9c78a2695e050393fcfbb5e8b91f902d194a4 |

## **Summarized findings and Vulnerability Graph**

|  |  |
| --- | --- |
| **Classification** | **Quantity** |
| **Target** | **Insecurebankv2.apk** |
| **Total vulnerabilities found** | **14** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CRITICAL/ HIGH / MEDIUM / LOW** | **01** | **03** | **10** | **00** |

## **Vulnerability List**

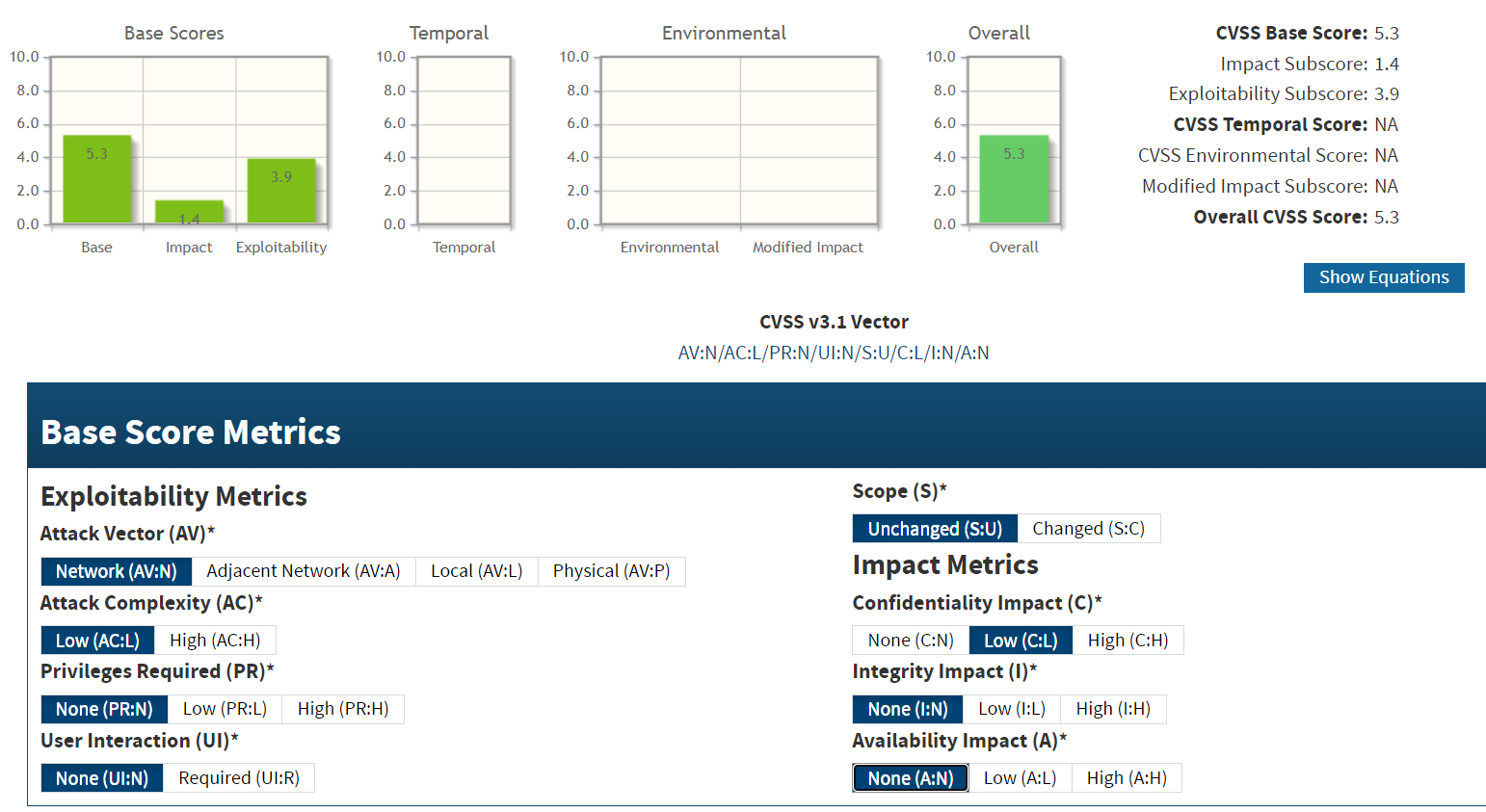
|  |  |  |
| --- | --- | --- |
| **No** | **VULNERABILITY LIST** | **STATUS** |
| **1** | **Developer Login** | **MEDIUM** |
| **2** | **Weak Cryptography in data storage** | **MEDIUM** |
| **3** | **Insecure Logging** | **MEDIUM** |
| **4** | **Application Backup Enabled** | **HIGH** |
| **5** | **Bypassing Login Screen using Exported Activity** | **HIGH** |
| **6** | **Hidden Create User Button for Admins** | **MEDIUM** |
| **7** | **Root Detection Bypass** | **MEDIUM** |
| **8** | **Debug Mode Enabled** | **CRITICAL** |
| **9** | **Flawed Broadcast Receivers** | **MEDIUM** |
| **10** | **Insecure Content Provider Access** | **MEDIUM** |
| **11** | **Insecure WebView Implementation** | **HIGH** |
| **12** | **Parameter Manipulation** | **MEDIUM** |
| **13** | **Username Enumeration** | **MEDIUM** |
| **14** | **Insecure HTTP Connections** | **MEDIUM** |

## **4. Vulnerability details**

### **Developer Login**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **MEDIUM** | **Business Impact** | **MEDIUM** |



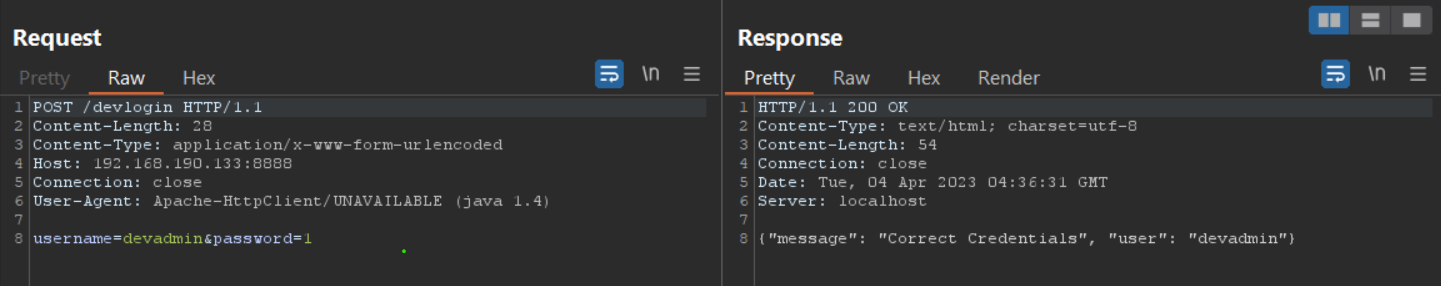
***Description:***

This is a type of security vulnerability that allows an attacker to gain unauthorized access to a system by exploiting a backdoor or other insecure access path used by developers. This type of vulnerability can allow an attacker to bypass the login system without any credentials.

In this case, the developer forgets to delete the test account that the attack can bypass login.

***Proof of Concept:***

Set up Burp Suite’s proxy to intercept the request. In the login screen, if we login with username ‘devadmin’ and any password, we will login successfully.



***Exploitation Tool:***

Burp Suite, Genymotion, jadx, Test Manual

***Recommendation:***

Remove the use of the “devadmin” account.

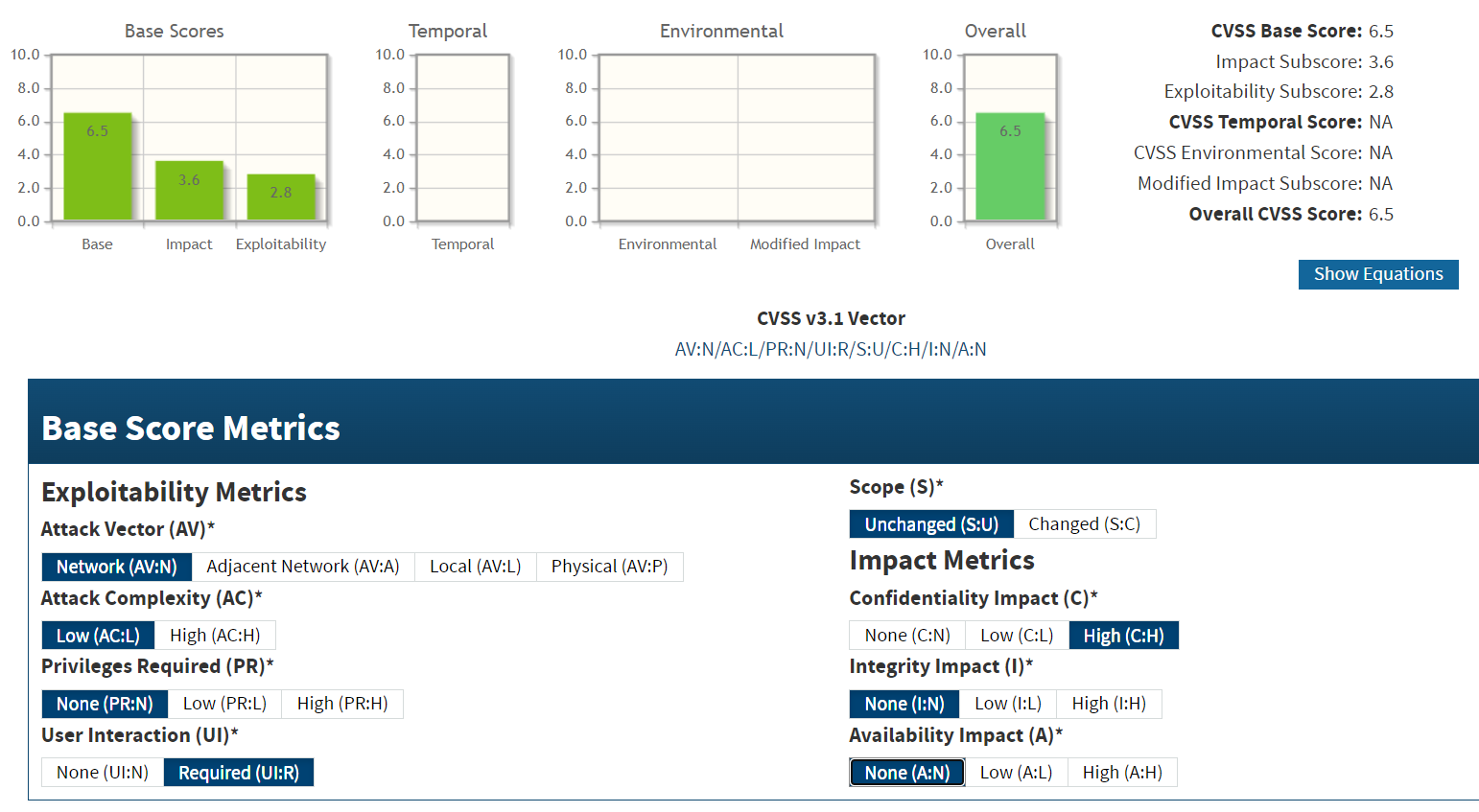
***References****:*

<https://auth0.com/blog/developers-guide-to-common-vulnerabilities-and-how-to-prevent-them/>

### **4.2 Weak Cryptography in data storage**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **MEDIUM** | **Business Impact** | **HIGH** |

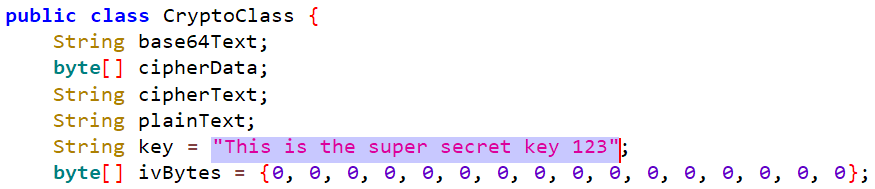


***Description:***

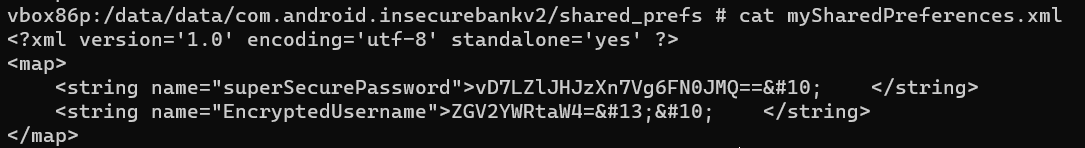
The user’s credentials are stored in the Shared Preferences which is easily access in a rooted device from /data/data/com.android.insecurebankv2/shared\_prefs. Moreover, the application use hard-coded key for encryption/decryption which can be seen when reversing the apk file.

***Proof of Concept:***

Using jadx to reversing the apk file. In the class named CryptoClass, we can see the key: "This is the super secret key 123"



Use adb from rooted device to get access to the Shared Preferences:



Use this python script to decrypt the credentials

from Crypto.Cipher import AES

import base64

def unpad(s):

    return s[:-ord(s[len(s)-1:])]

def decrypted(ciphertext, key, iv):

    cipher = AES.new(key, AES.MODE\_CBC, iv)

    return cipher.decrypt(ciphertext)

key = b'This is the super secret key 123'

iv = b'\x00' \* 16

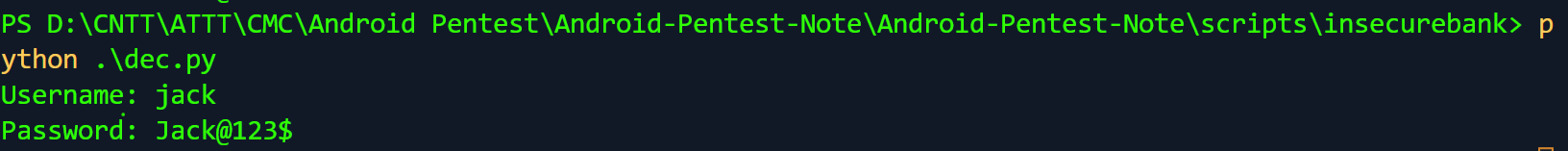
ciphertext\_password = base64.b64decode(">v/sJpihDCo2ckDmLW5Uwiw==")

username = base64.b64decode("amFjaw==").decode('utf-8')

password = unpad(decrypted(ciphertext\_password, key, iv)).decode('utf-8')

print(f'Username: {username}')

print(f'Password: {password}')



***Exploitation Tool:***

Adb, Genymotion, jadx, Test Manual

***Recommendation:***

Use android Keystore system to store key securely

Get the key:

KeyGenerator keyGenerator;

SecretKey secretKey;

try {

    keyGenerator = KeyGenerator.getInstance("AES");

    keyGenerator.init(256);

    secretKey = keyGenerator.generateKey();

 } catch (Exception e) {

    e.printStackTrace();

 }

Initialize the IV:

byte[] IV = new byte[16];

SecureRandom random;

random = new SecureRandom();

random.nextBytes(IV);

Encryption:

public static byte[] encrypt(byte[] plaintext, SecretKey key, byte[] IV) throws Exception {

    Cipher cipher = Cipher.getInstance("AES");

    SecretKeySpec keySpec = new SecretKeySpec(key.getEncoded(), "AES");

    IvParameterSpec ivSpec = new IvParameterSpec(IV);

    cipher.init(Cipher.ENCRYPT\_MODE, keySpec, ivSpec);

    byte[] cipherText = cipher.doFinal(plaintext);

    return cipherText;

 }

Decryption:

public static String decrypt(byte[] cipherText, SecretKey key, byte[] IV) {

    try {

        Cipher cipher = Cipher.getInstance("AES");

        SecretKeySpec keySpec = new SecretKeySpec(key.getEncoded(), "AES");

        IvParameterSpec ivSpec = new IvParameterSpec(IV);

        cipher.init(Cipher.DECRYPT\_MODE, keySpec, ivSpec);

        byte[] decryptedText = cipher.doFinal(cipherText);

        return new String(decryptedText);

    } catch (Exception e) {

        e.printStackTrace();

    }

    return null;

 }

***References****:*

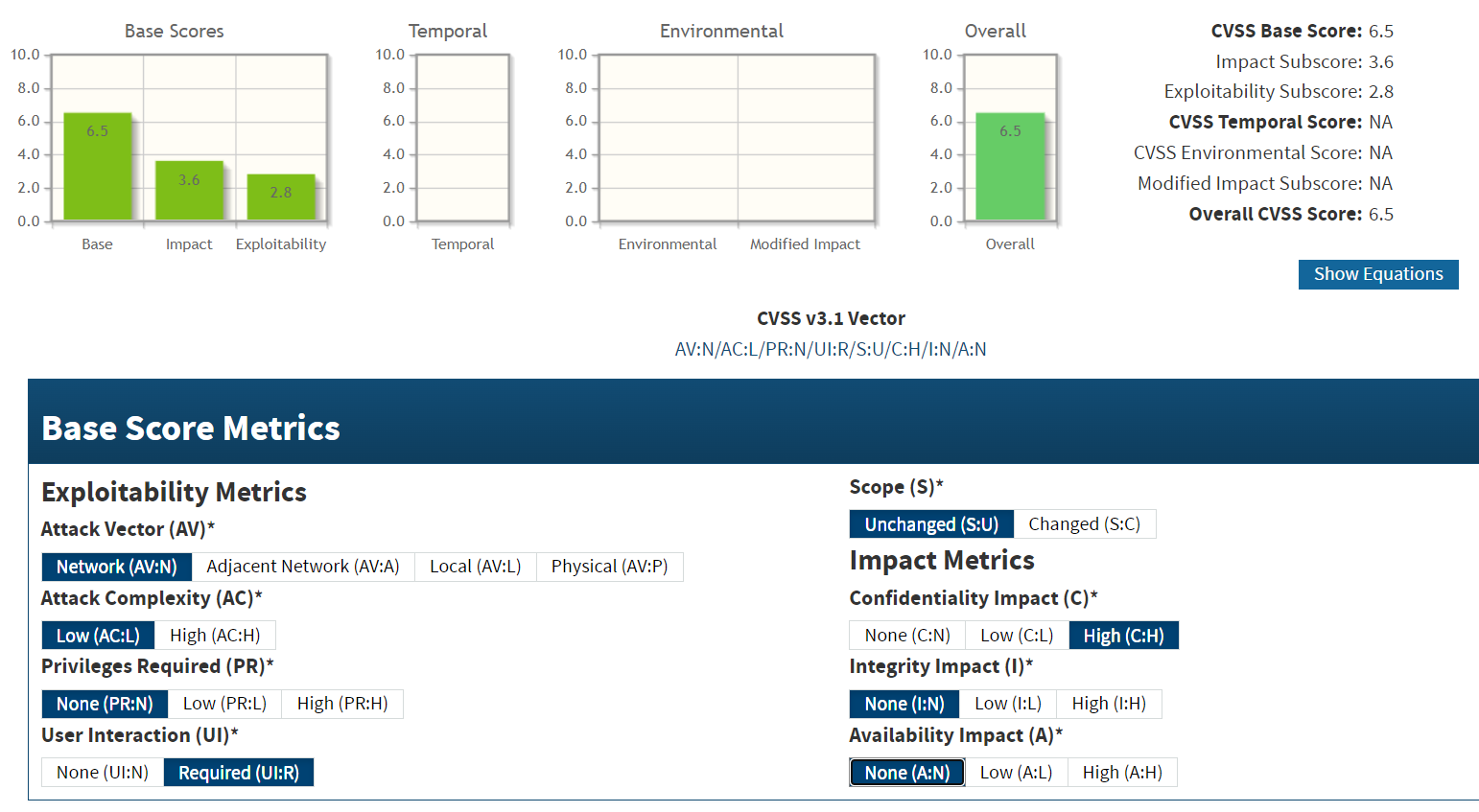
[AES 256 Encryption and Decryption in Android with Example (amarinfotech.com)](https://amarinfotech.com/how-to-do-aes-256-encryption-decryption-in-android.html)

[Android Keystore system  |  Android Developers](https://developer.android.com/training/articles/keystore)

### **4.3  Insecure Logging**

***The following summarises the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **MEDIUM** | **Business Impact** | **HIGH** |



***Description:***

The user’s credentials can be accessed through logcat. When a victim login successfully, the application write his/her credentials in the logcat. The attacker can get access to it using adb logcat.

This vulnerability will lead to the leakage of sensitive information

***Proof of Concept:***

Login as jack:Jack@123$. Use adb logcat on rooted device, we can see user jack’s credential.



***Exploitation Tool:***

adb, Genymotion, jadx, Test Manual

***Recommendation:***

Do not print sensitive information in the log

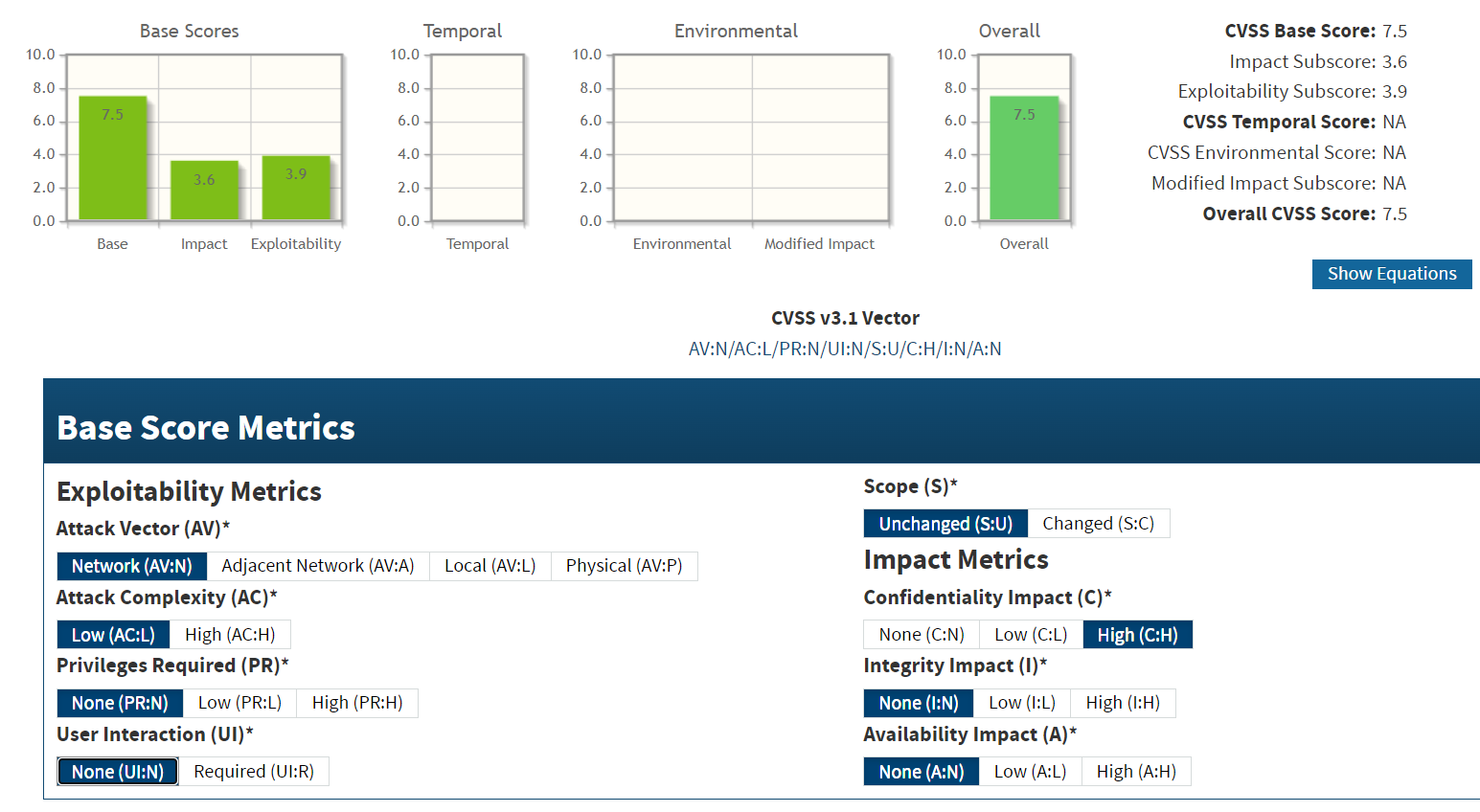
***References****:*

[01. Insecure Logging. Logging is a method that developers use… | by Galilei | Mobile Penetration Testing | Medium](https://medium.com/mobile-penetration-testing/01-insecure-logging-a41c41a8e648)

### **4.4  Application Backup Enabled**

***The following summarises the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **MEDIUM** | **Business Impact** | **HIGH** |



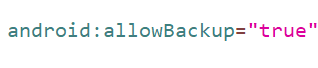
***Description:***

**This is a feature that is used to enable a backup storage device such as an external hard drive or an online cloud storage account. When enabled, a copy of the data stored on the primary storage device is backed up to the secondary device on a regular basis. This provides an extra layer of protection if the primary storage device fails or is damaged.**

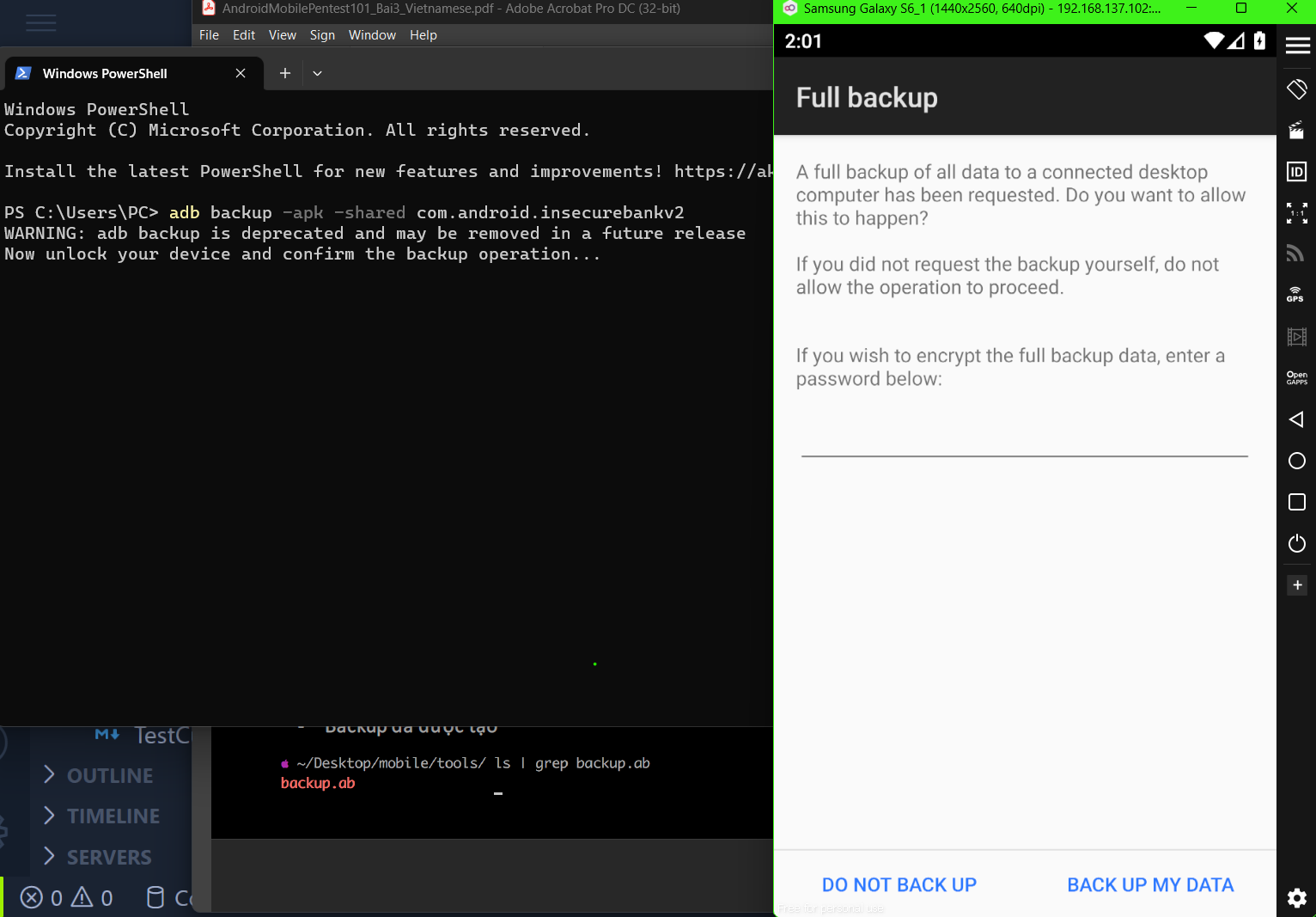
**In this case, the application did not control which kind of data will be backed up. Therefore, the attacker can backup the data the get the sensitive information.**

***Proof of Concept:***

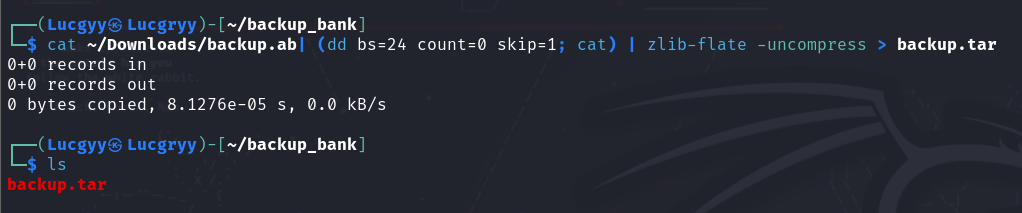
Use jadx to reverse the apk file. In AndroidManifest.xml, we can see that this application allow backup feature

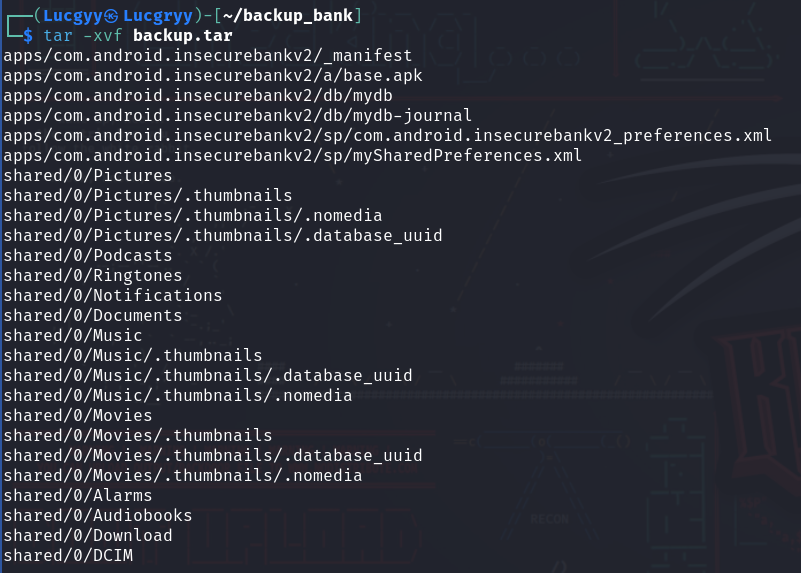


Use adb backup com.android.insecurebankv2 to create a backup file. In the device, it will ask for permission to backup, choose backup my data, we will get the file named backup.ab

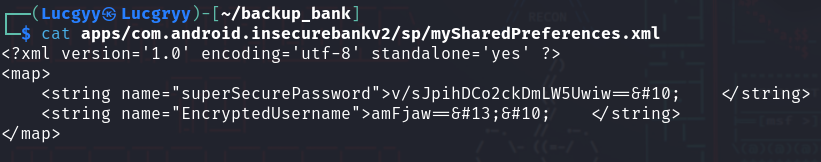


Convert the backup.ab into backup.tar like below



Extract that .tar file

We can read the content of the Shared Preferences



***Exploitation Tool:***

adb, Genymotion, jadx, Test Manual

***Recommendation:***

Set up the back up rule to control which kind of information in backed up. More detail here: [Back up user data with Auto Backup  |  Android Developers](https://developer.android.com/guide/topics/data/autobackup)

If the application do not allow backup, set the allowBackup to false in AndroidManifest.xml

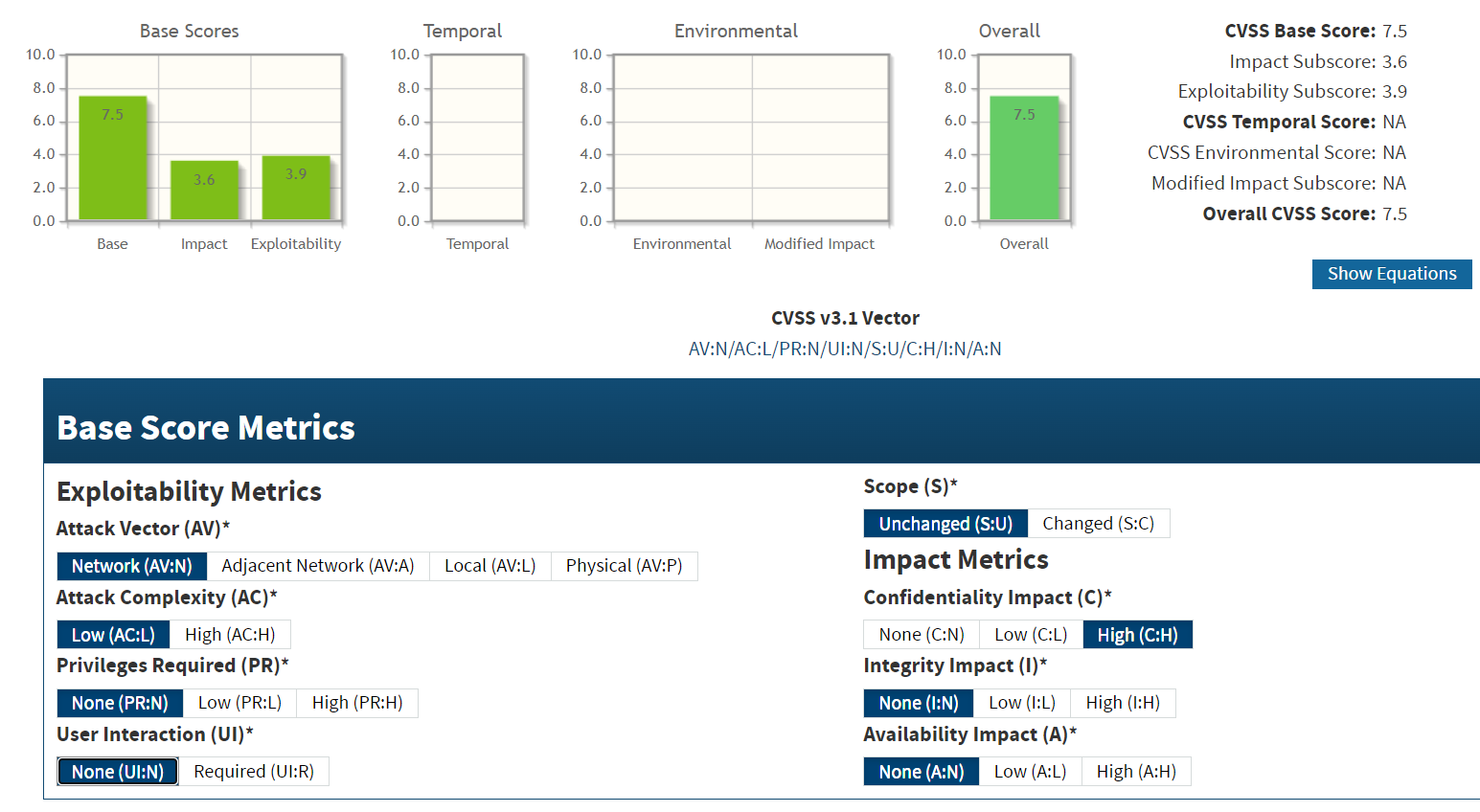
***References****:*

[Back up user data with Auto Backup  |  Android Developers](https://developer.android.com/guide/topics/data/autobackup)

### **4.5  Bypassing Login Screen using Exported Activity**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **HIGH** | **Business Impact** | **HIGH** |

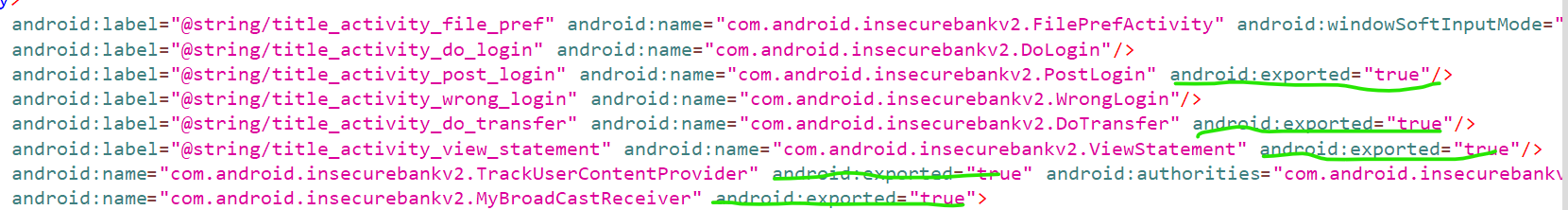


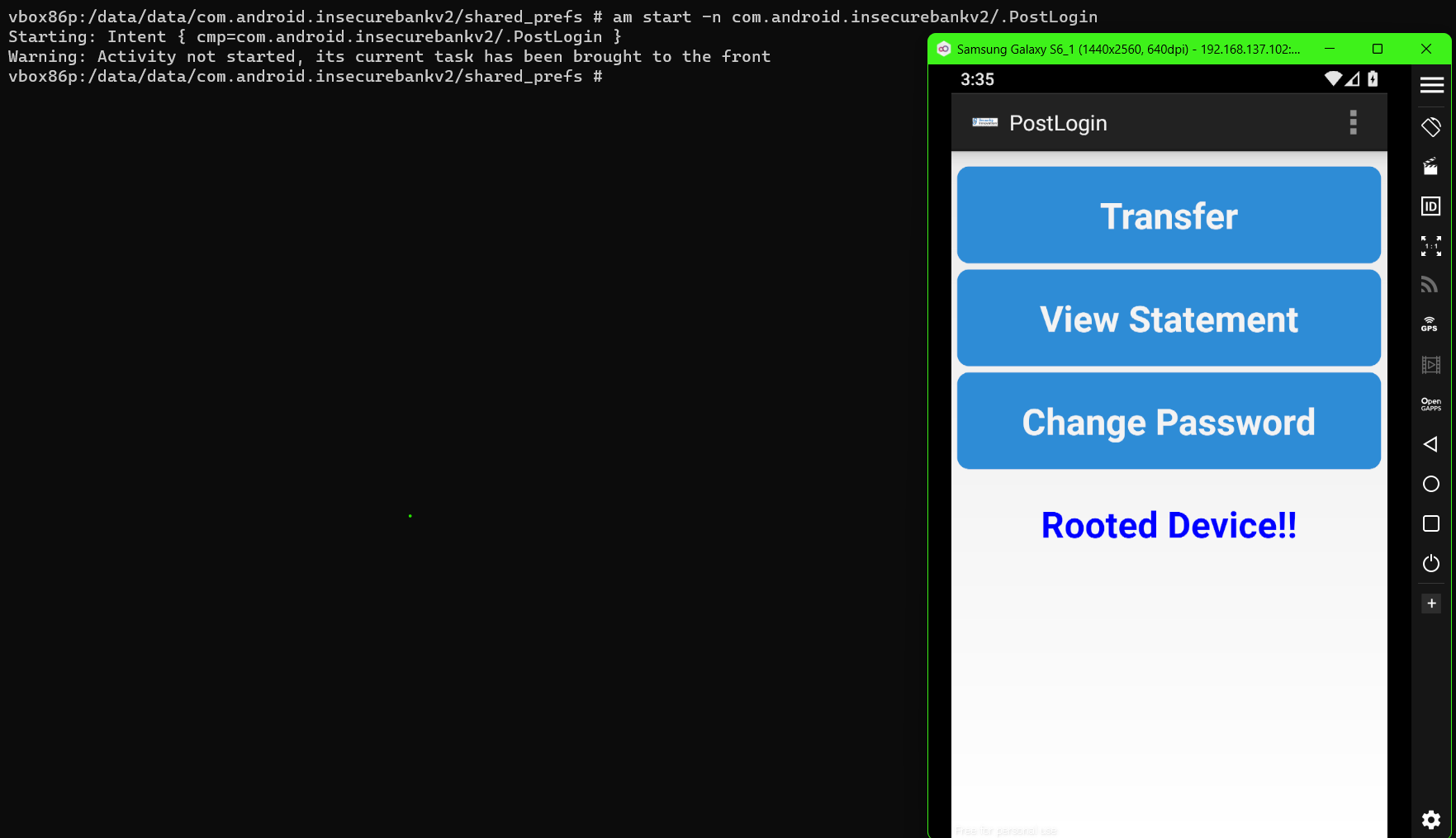
***Description:***

This application has some exported Activities which can be navigated through without application’s user interaction. The attacker can easily bypass the login screen by go to those exported Activities (for example PostLogin activities)

***Proof of Concept:***

Use jadx to reverse the apk file. In AndroidManifest.xml, we can see that this application has some exported Activitites



  Use adb shell to go to the devices command line. The execute: am start -n com.android.insecurebankv2/.PostLogin to navigate to PostLogin activity without provide any credentials.

***Exploitation Tool:***

adb, Genymotion, jadx, Test Manual

***Recommendation:***

Set the exported attribute in the AndroidManifest.xml to false

***References****:*

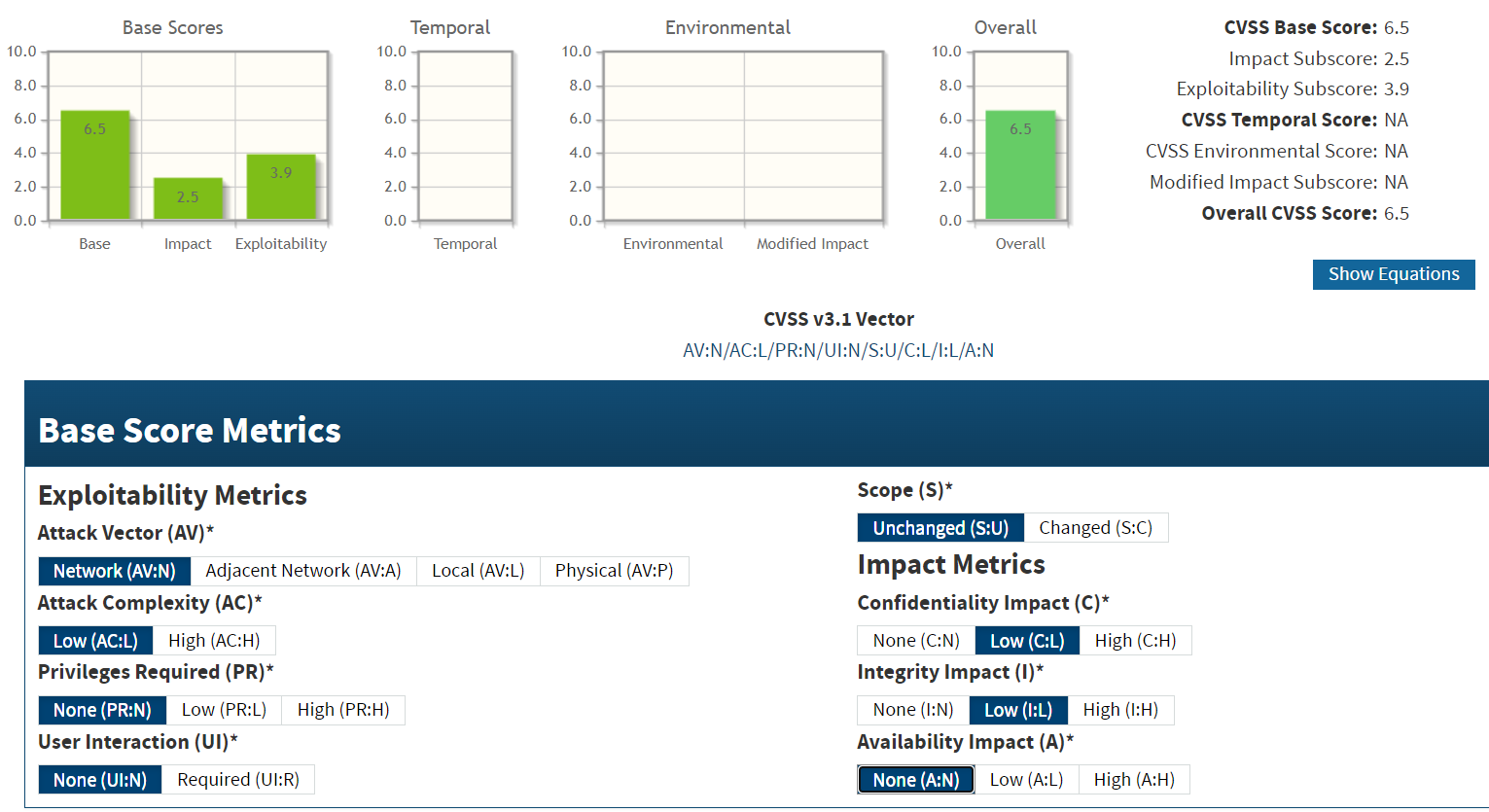
[Exploiting Android Activity “<activity android:exported=”true”>“ | by Ashish Upsham | Medium](https://aupsham98.medium.com/exploiting-android-activity-activity-android-exported-true-93ffeb263682)

[CWE - CWE-926: Improper Export of Android Application Components (4.10) (mitre.org)](https://cwe.mitre.org/data/definitions/926.html)

### **4.6  Hidden Create User Button for Admins**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **MEDIUM** | **Business Impact** | **MEDIUM** |



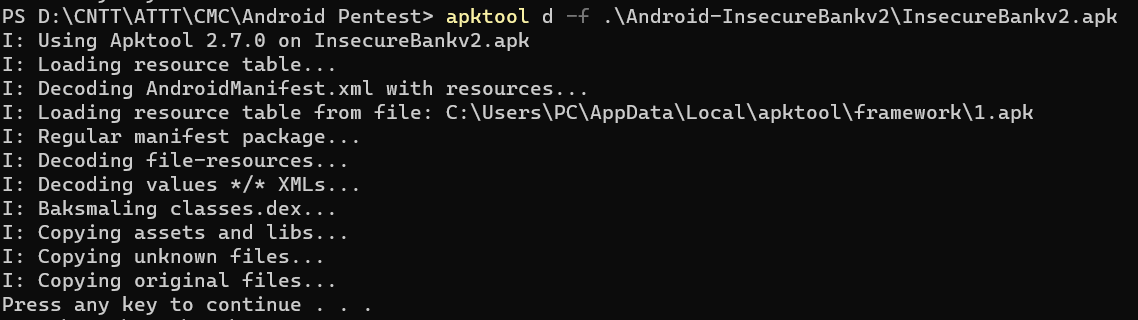
***Description:***

This application has a hidden button that can be used to create a user and only admin can get access to it.

The attacker can reverse engineer the apk file and patch the app so that this application is used by an admin. Therefore, it will make that hidden button appear and potentially exploit the app by utilizing the admin’s functionality.

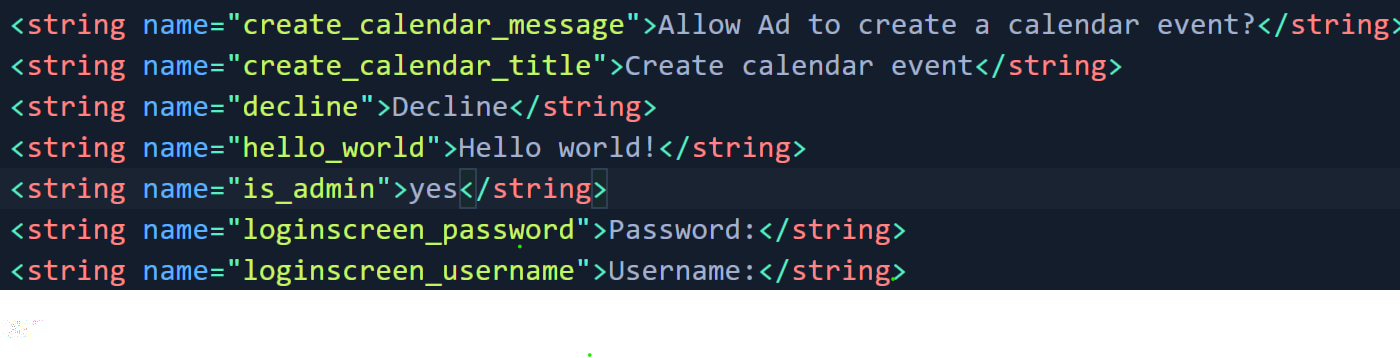
***Proof of Concept:***

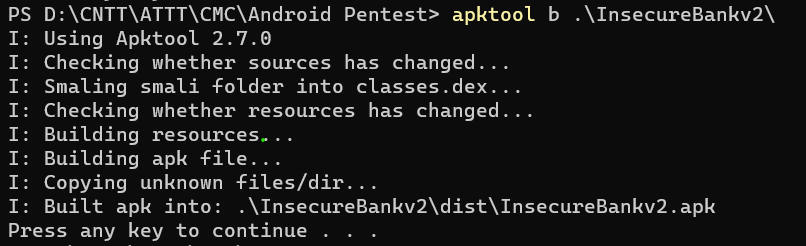
Decompile the app using apktool, you will get a folder named InsecureBankv2:



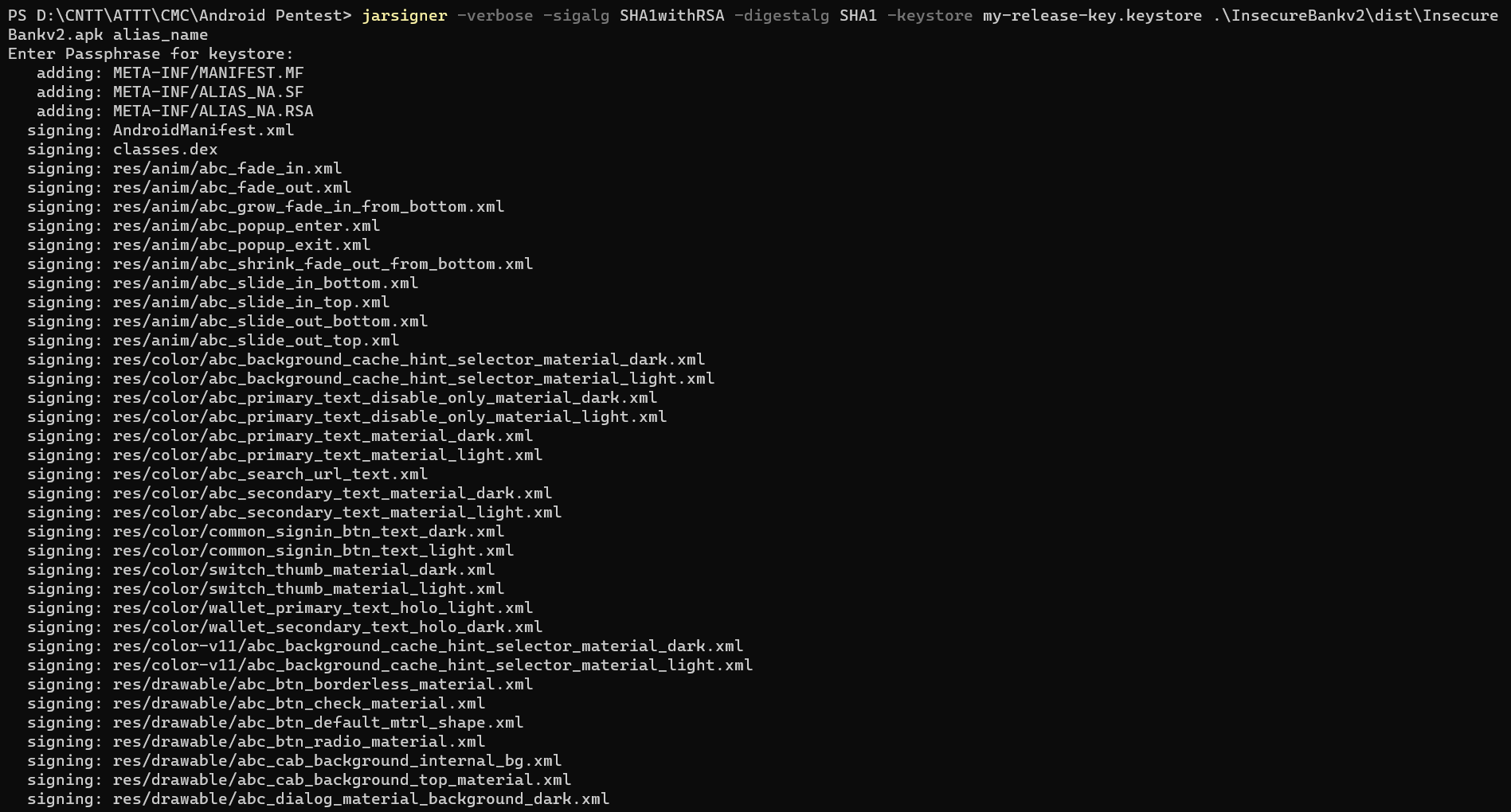
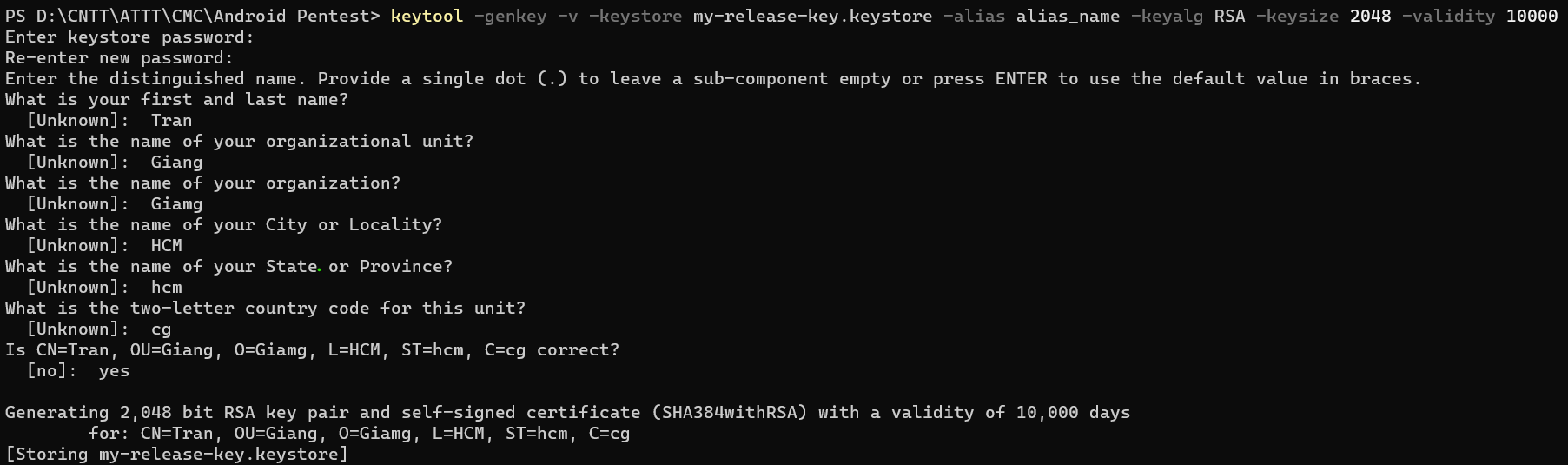
  In InsecureBankv2/res/values/strings.xml, change the value of ‘is\_admin’ from ‘no’ to ‘yes’.

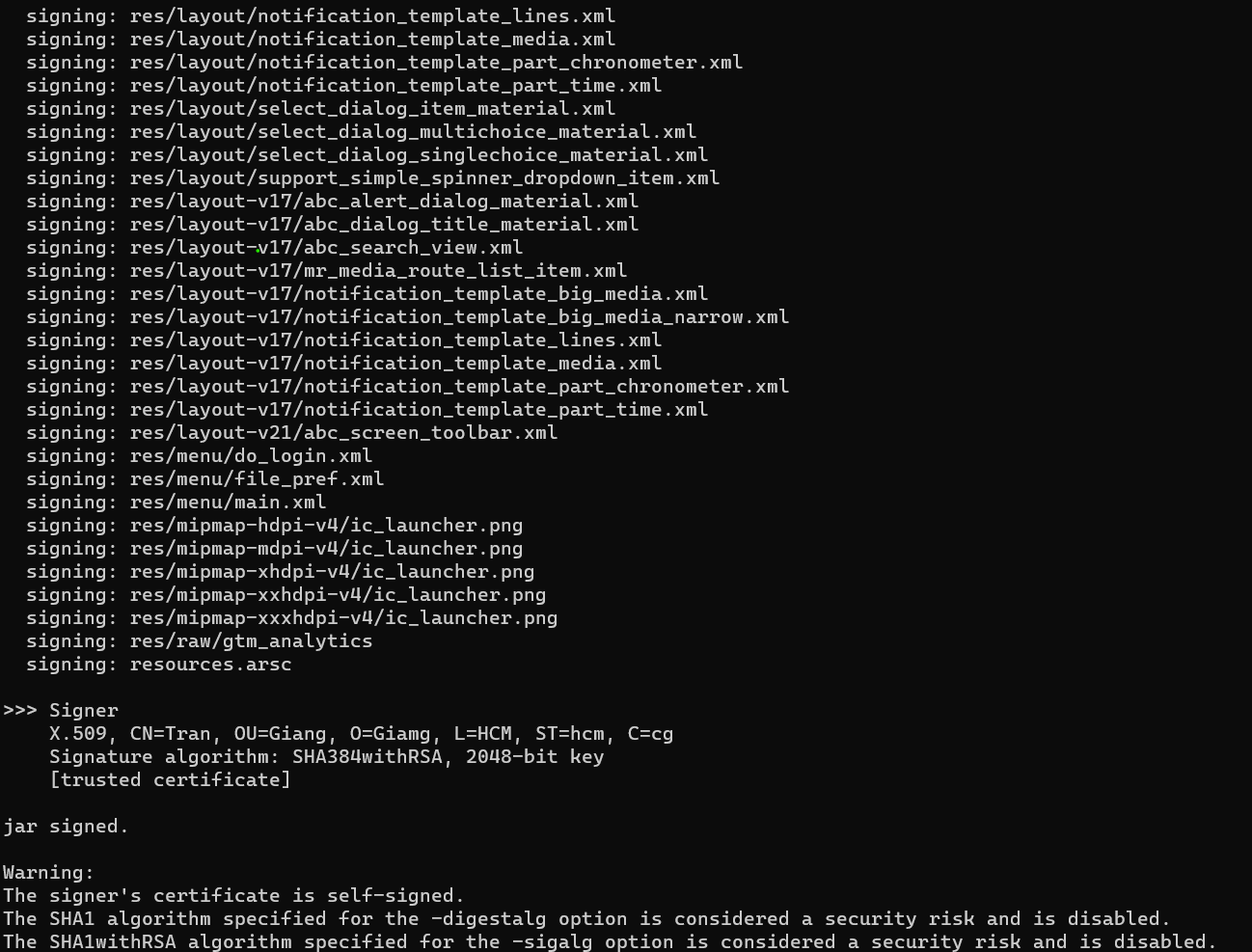




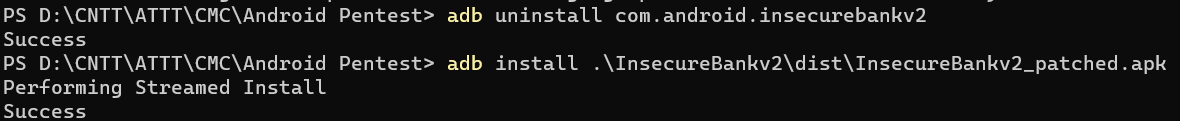
Rebuild the app using apktool:

Signing the app

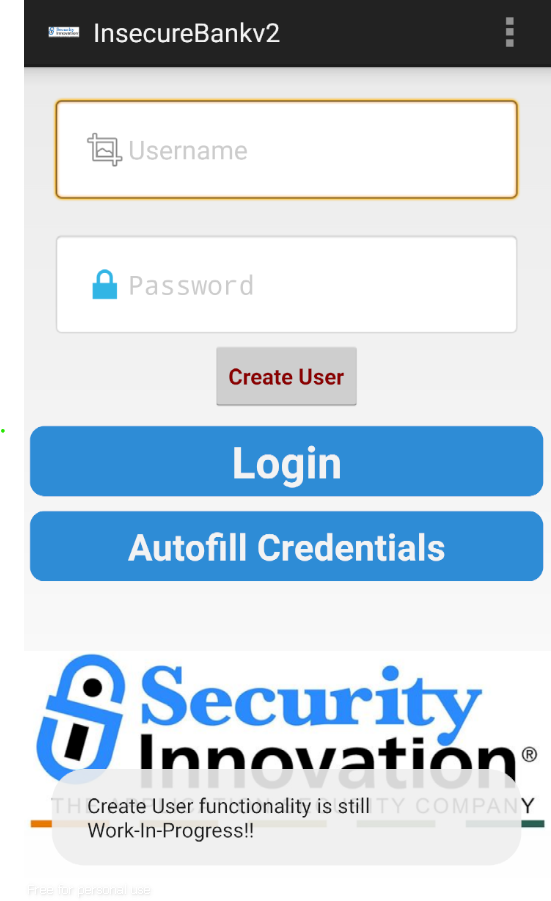




The patched app will be in the InsecureBankv2/dist folder. Install that patched app into the device.



Run that app, we can see the button appear.



***Exploitation Tool:***

Apktool, adb, Genymotion, jadx, Test Manual

***Recommendation:***

**Implement File Integrity Checks by checking these files: Android Manifest.xml, class files \*.dex, and native libraries (\*.so)**

Implement role-based access control (RBAC) to ensure that only users with the appropriate permissions can access the admin’s functionality

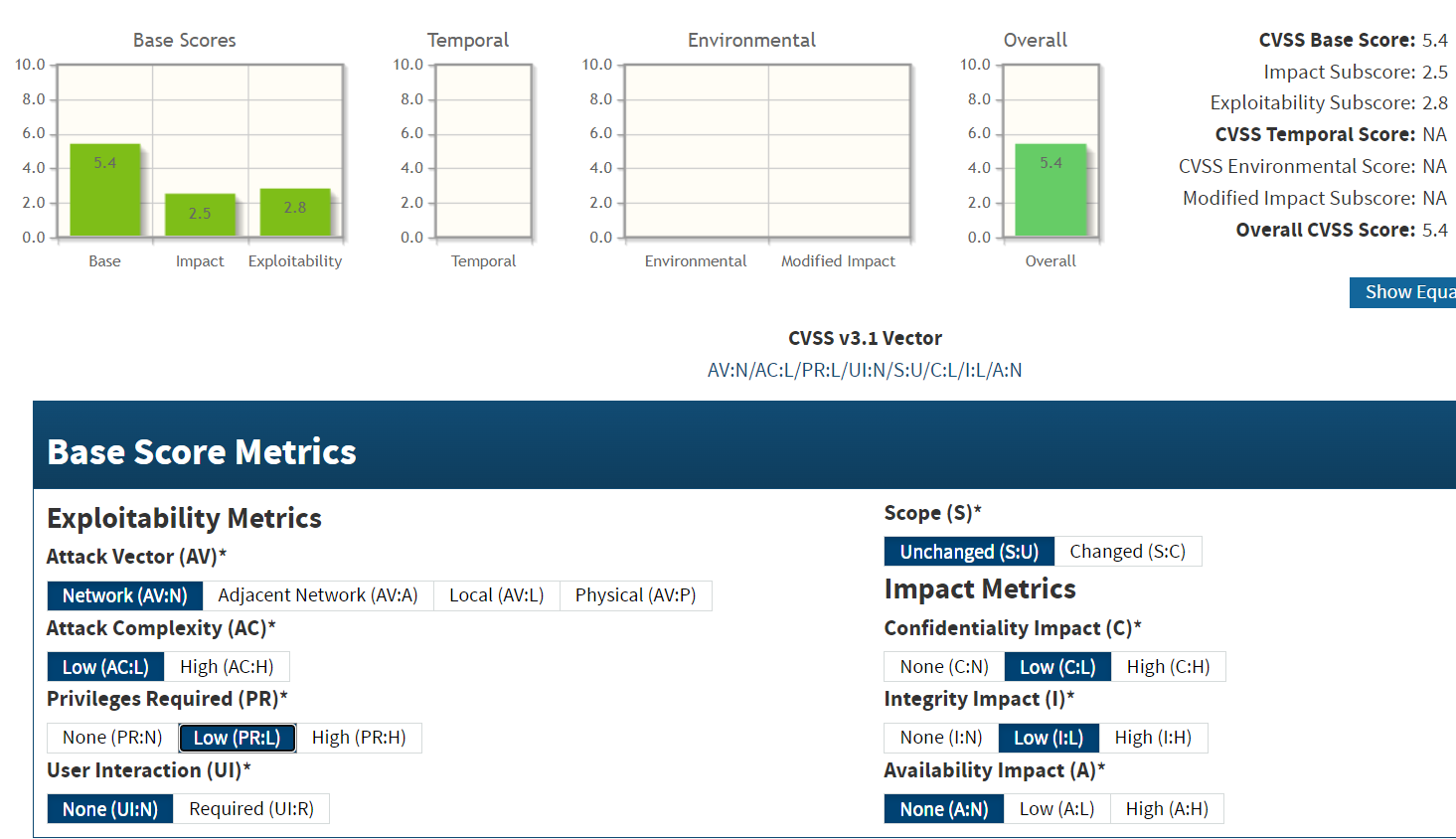
***References****:*

[Android Tampering and Reverse Engineering - OWASP Mobile Application Security](https://mas.owasp.org/MASTG/Android/0x05c-Reverse-Engineering-and-Tampering/)

### **4.7  Root Detection Bypass**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **MEDIUM** | **Technical Impact** | **MEDIUM** | **Business Impact** | **MEDIUM** |



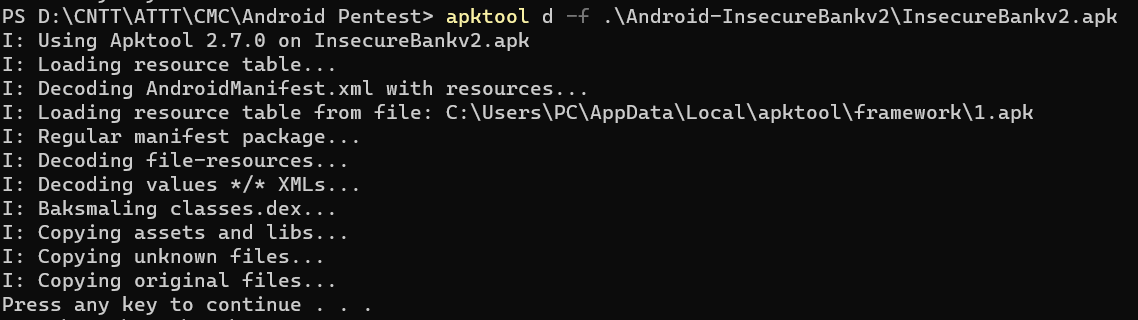
***Description:***

This is a type of vulnerability that allows an attacker to bypass security measures and gain access to a system or application by exploiting a flaw in the root detection mechanism. This type of attack can be used to gain access to sensitive information or to launch malicious code on the system. Root Detection Bypass can be accomplished by exploiting a flaw in the root detection system, such as an incorrect configuration or a programming error.

In this case, the attack use this application in a rooted device.

***Proof of Concept:***

Decompile the app using apktool, you will get a folder named InsecureBankv2:

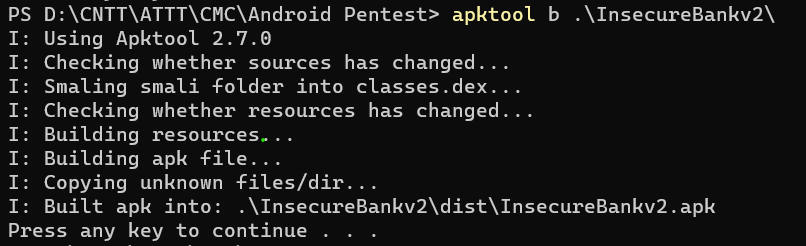


In InsecureBankv2/smali/com/android/insecurebankv2/PostLogin.smali, in method ‘showRootStatus()’. We can see that when if-ne v0, v1, it will jump to cond\_2 which is the ‘Device not Rooted!!’.

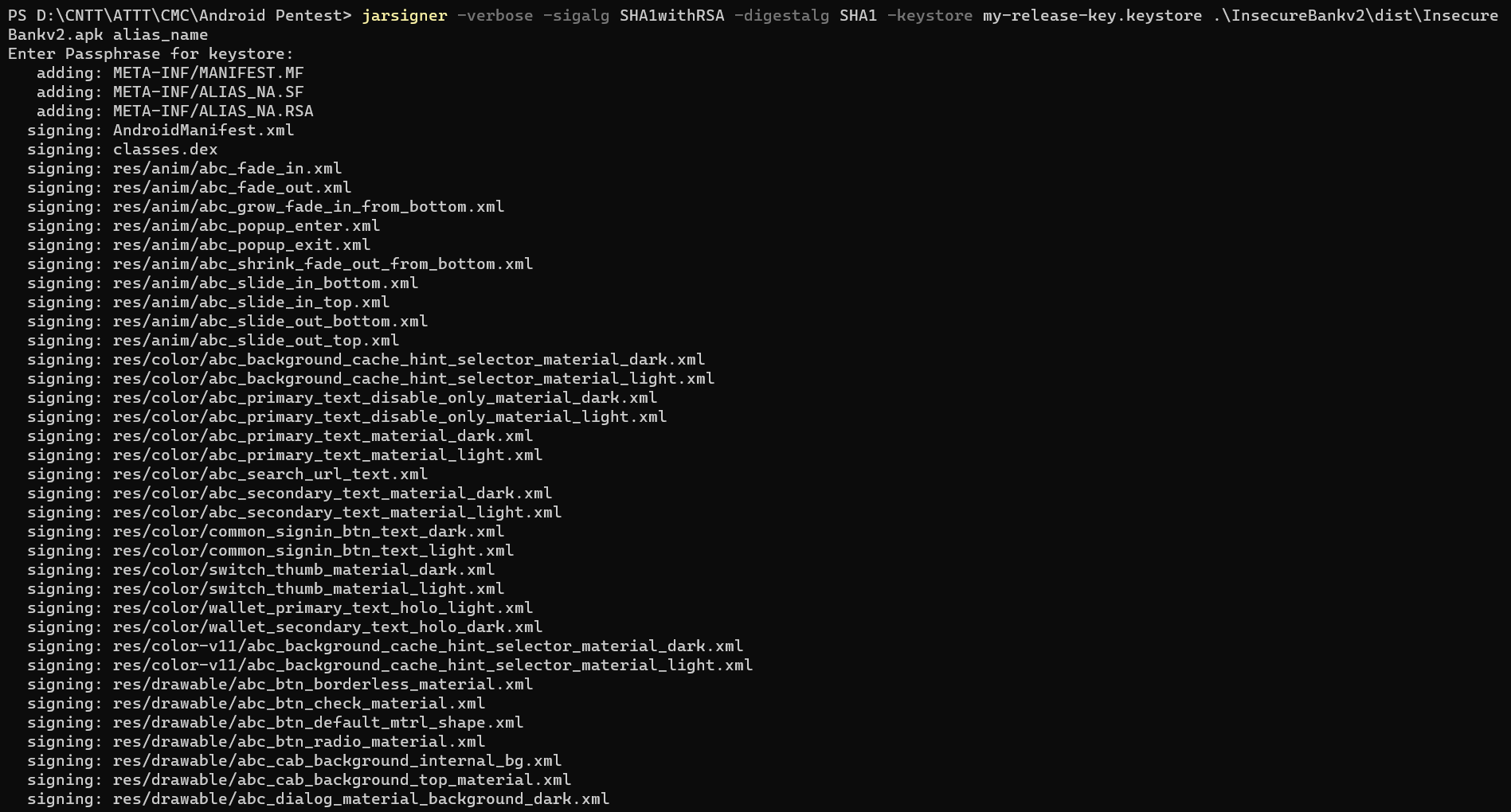
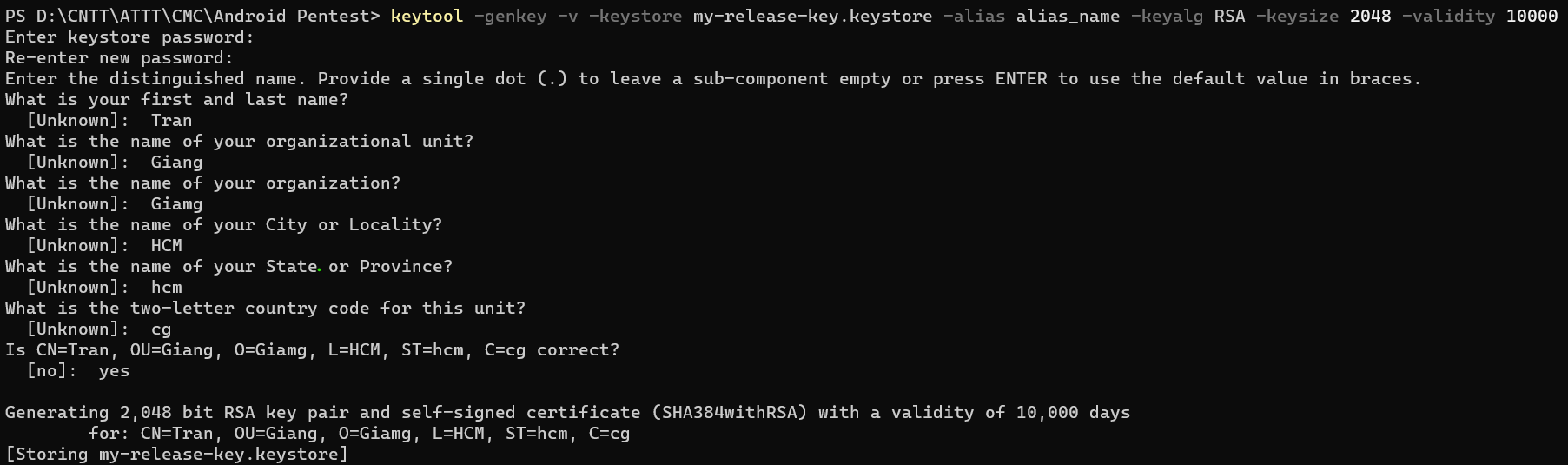


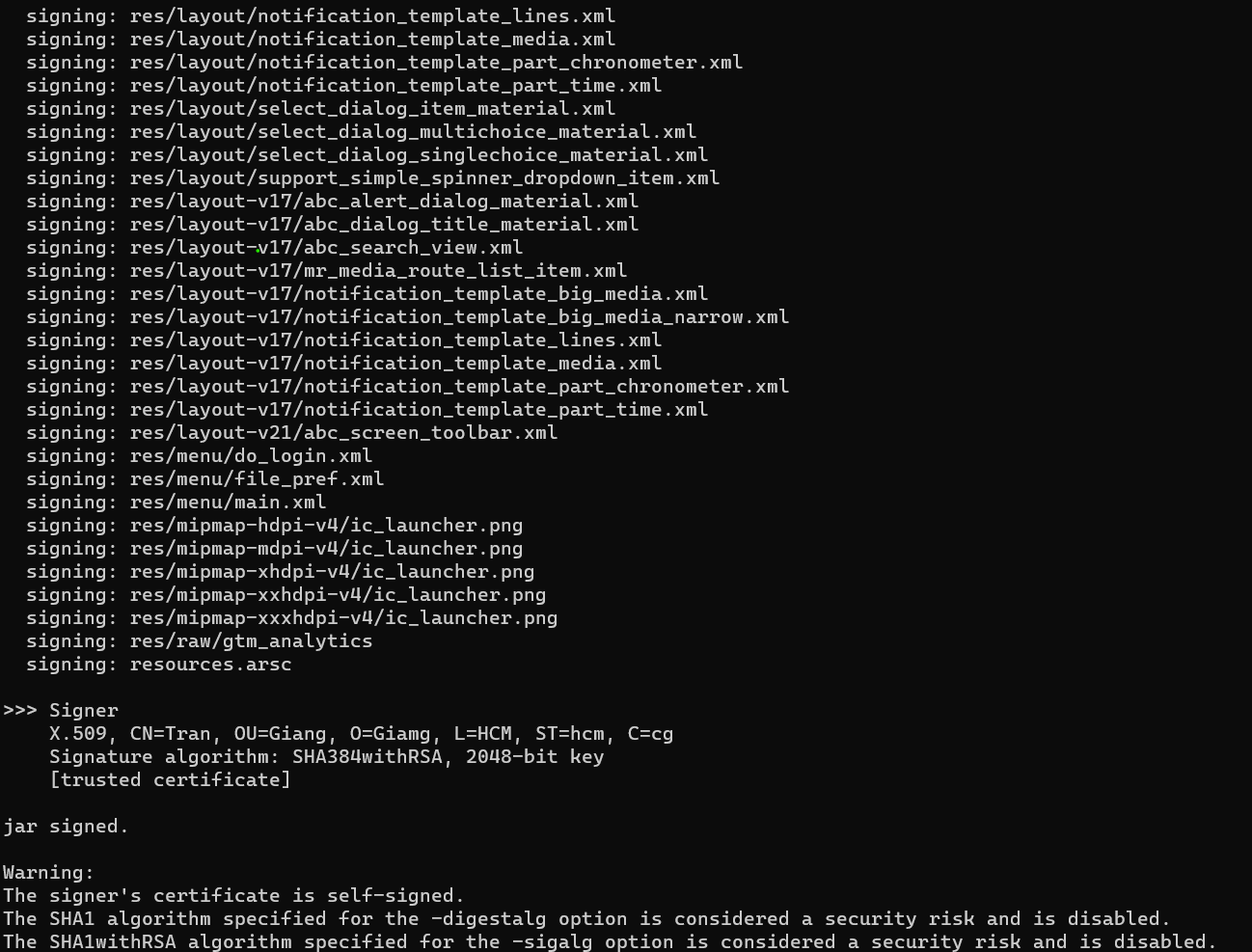
Change the code into ‘goto :cond\_2’ to bypass the root detection.



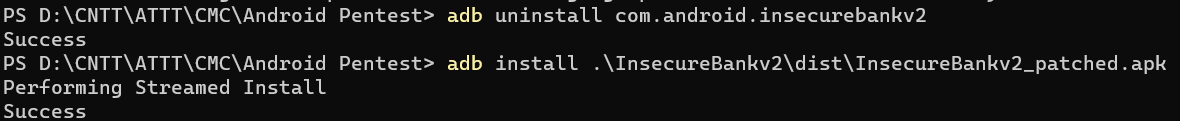
Rebuild the app using apktool:

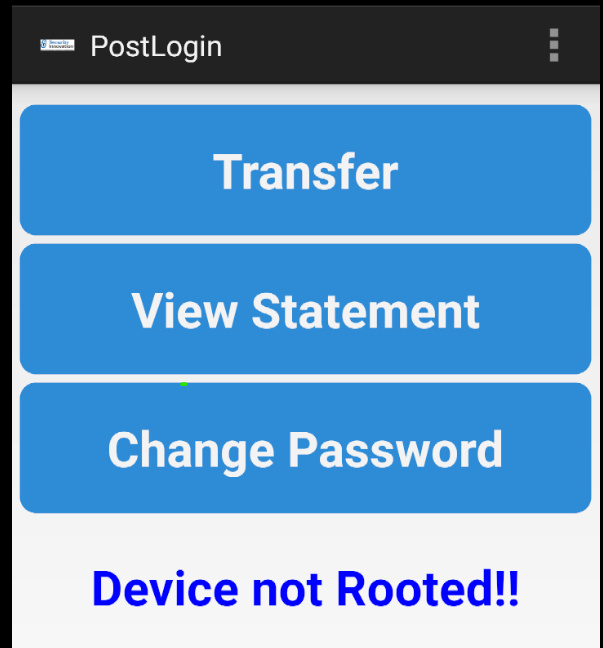
Signing the app





The patched app will be in the InsecureBankv2/dist folder. Install that patched app into the device.

Run that app, we can see that the root detection is bypassed

.

***Exploitation Tool:***

adb, Genymotion, jadx, Test Manual

***Recommendation:***

**Implement File Integrity Checks by checking these files: Android Manifest.xml, class files \*.dex, and native libraries (\*.so)**

Use root-detection libraries (Rootbeer).

***References****:*

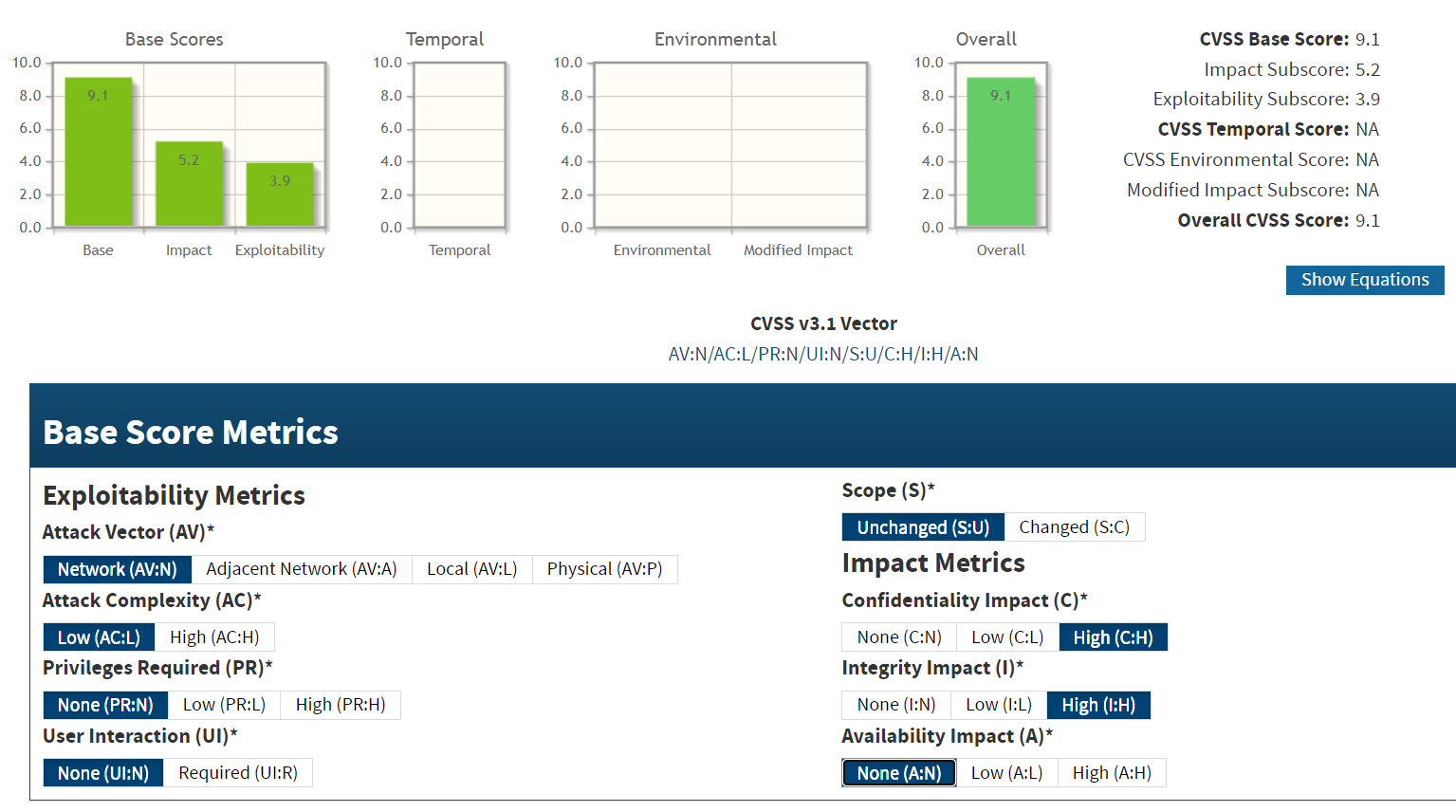
<https://mas.owasp.org/MASTG/Android/0x05j-Testing-Resiliency-Against-Reverse-Engineering/>

[scottyab/rootbeer: Simple to use root checking Android library and sample app (github.com)](https://github.com/scottyab/rootbeer)

### **4.8 Debug Mode Enabled**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **HIGH** | **Business Impact** | **HIGH** |



***Description:***

This is a security vulnerability that allows a user to access the application’s debug mode and bypass certain security checks. This can be exploited to gain access to sensitive data, modify application configurations, or execute malicious code

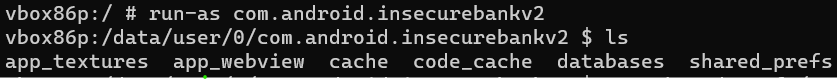
In this case, the developer forget to disable the debugging feature

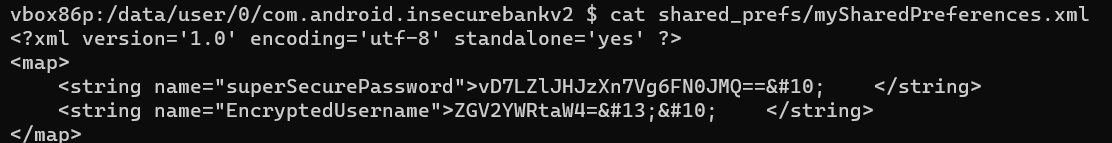
***Proof of Concept:***

Use jadx to reverse the apk file. In AndroidManifest.xml, we can see that this application enabled debugging



  With that feature, the attacker can access the application’s data as non-rooted user for example sqlite, shared preferences,...





***Exploitation Tool:***

adb, Genymotion, jadx, Test Manual

***Recommendation:***

In AndroidManifest.xml, set the ‘debuggable’ attribute to ‘false’

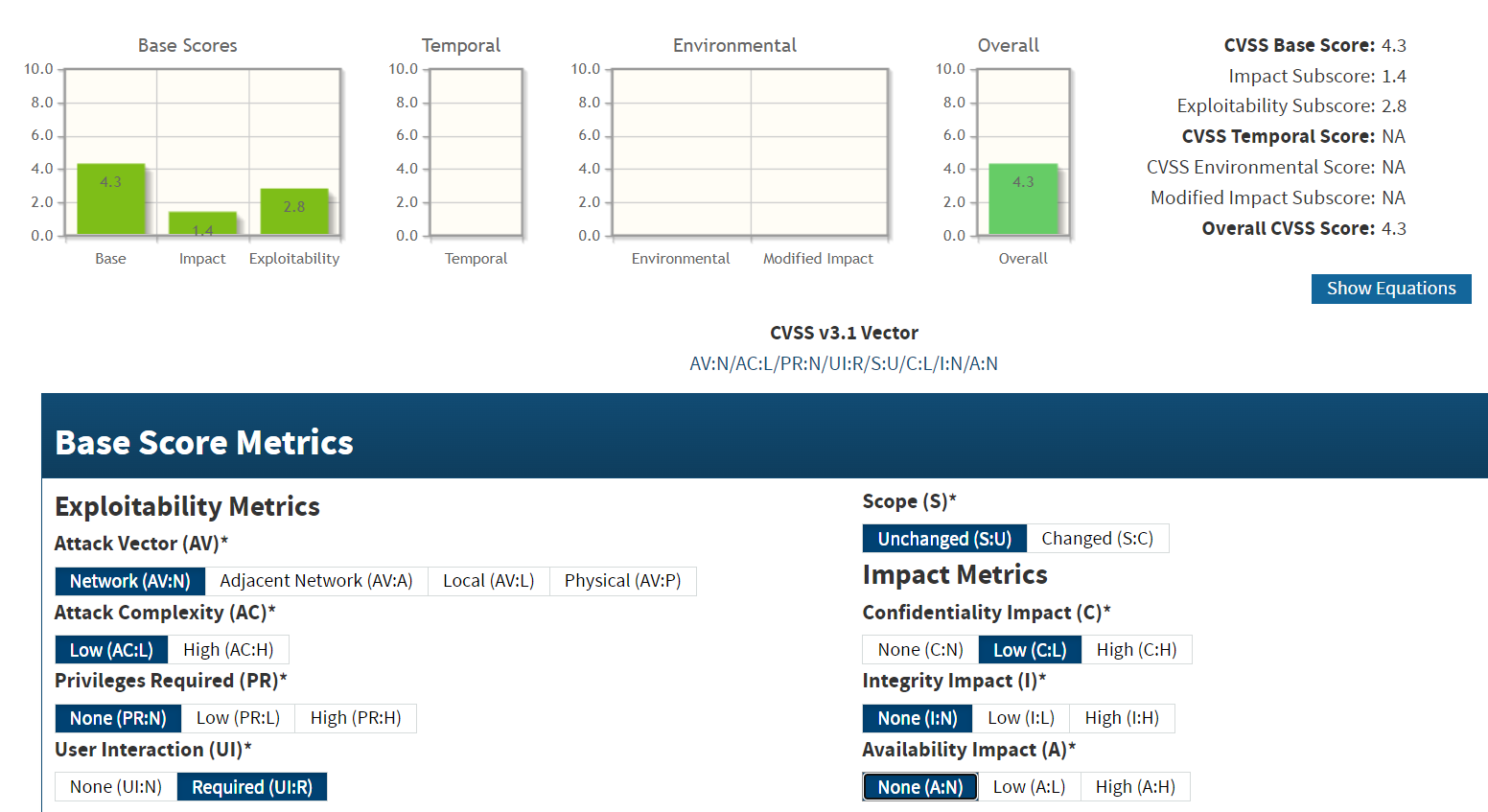
***References****:*

[Exploiting debuggable android applications | Infosec Resources (infosecinstitute.com)](https://resources.infosecinstitute.com/topic/android-hacking-security-part-6-exploiting-debuggable-android-applications/)

### **4.9 Flawed Broadcast Receivers**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **MEDIUM** | **Business Impact** | **MEDIUM** |

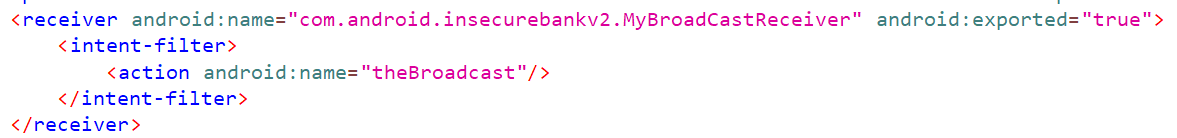


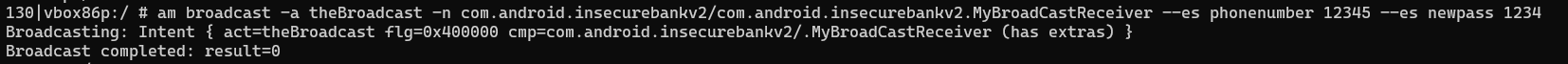
***Description:***

This application use Broadcast Receiver to send an SMS message to the user when he/she change password successfully. The message contain his/her old password and new password. The attacker can make the application to send an SMS message to the attacker’s phone number contains the victim’s old password

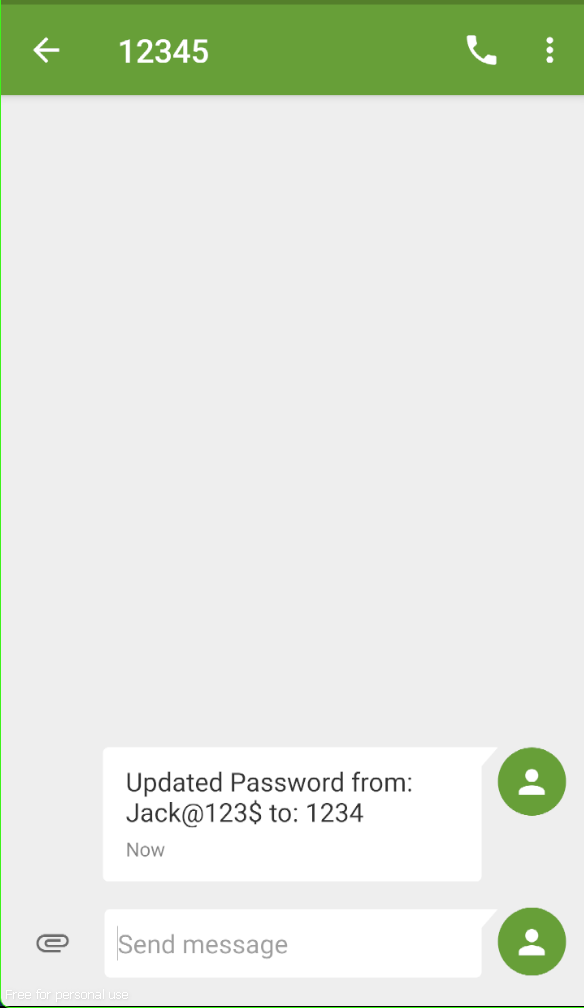
***Proof of Concept:***

Use jadx to reverse the apk file. In AndroidManifest.xml, we can see that this application has Broadcast Receiver



Use the command like below to make an application send an SMS message which contains the user’s current password  
 

Here is the result



***Exploitation Tool:***

adb, Genymotion, jadx, Test Manual

***Recommendation:***

**Set the permission for the broadcast by adding custom permission in the AndroidManifest.xml. For example**

****

****

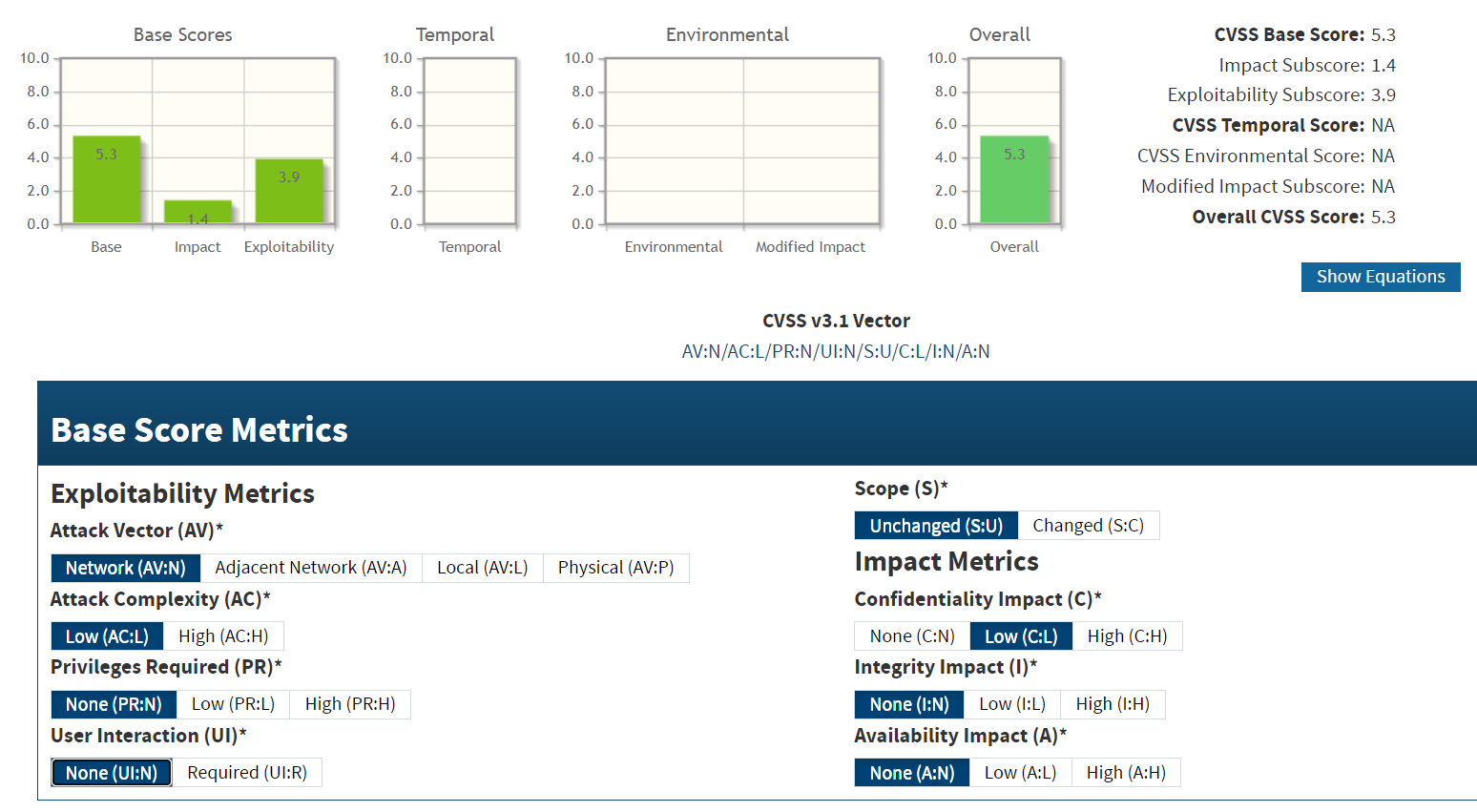
***References****:*

[Vulnerable Android Broadcast Receivers (oldbam.github.io)](https://oldbam.github.io/android/security/android-vulnerabilities-insecurebank-broadcast-receivers)

### **4.10 Insecure Content Provider Access**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **MEDIUM** | **Business Impact** | **MEDIUM** |

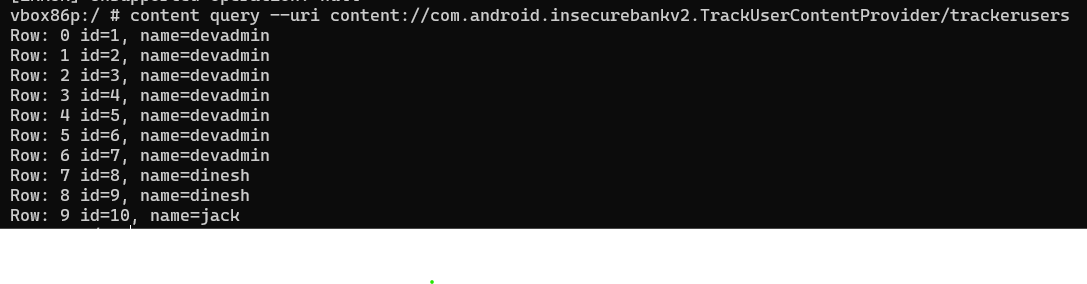


***Description:***

This is a type of security vulnerability that occurs when a malicious actor can access and manipulate a content provider's data. The content provider could be a third-party API, a website, or a mobile app. Content provider vulnerabilities can be exploited to gain unauthorized access to confidential information, or to manipulate data or applications. Common vulnerabilities include lack of authentication, weak authentication, weak encryption, and insufficient access control.

In this case, the attacker can get the data about the user who is logged in.

***Proof of Concept:***

Execute the command in the rooted device like below. The attacker can see the login history of the user

***Exploitation Tool:***

adb, Genymotion, jadx, Test Manual

***Recommendation:***

Set the permission to prevent the attacker exploits the content provider vulnerable

***References****:*

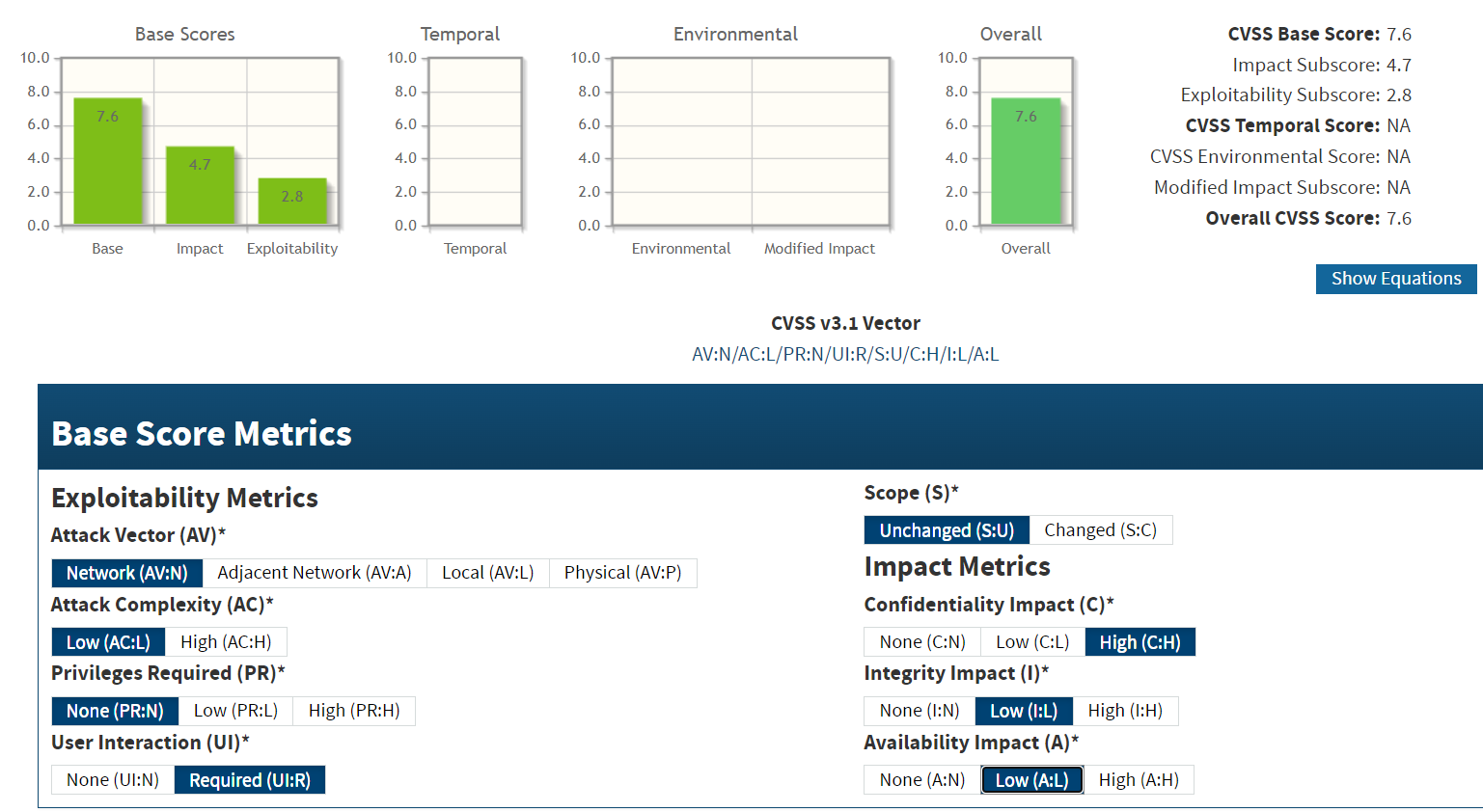
[Exploiting Content Providers - HackTricks](https://book.hacktricks.xyz/mobile-pentesting/android-app-pentesting/drozer-tutorial/exploiting-content-providers)

[Content provider basics  |  Android Developers](https://developer.android.com/guide/topics/providers/content-provider-basics)

### **4.11 Insecure WebView Implementation**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **HIGH** | **Business Impact** | **HIGH** |



***Description:***

In the View Statement functionality, this application loads an html file in /storage/emulated/0/ and display as a WebView to user. The attacker can make a malicious html file and send it to the device so that it will be loaded to the user instead

With that, the attacker can get access to the user’s sensitive data or execute malicious code on the user’s device.

***Proof of Concept:***

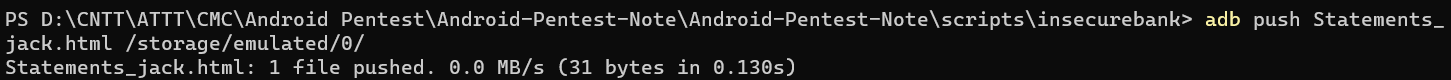
Use adb logcat on rooted device to identify the name of the file and it’s storage location (for example: /storage/emulated/0/Statements\_jack.html)



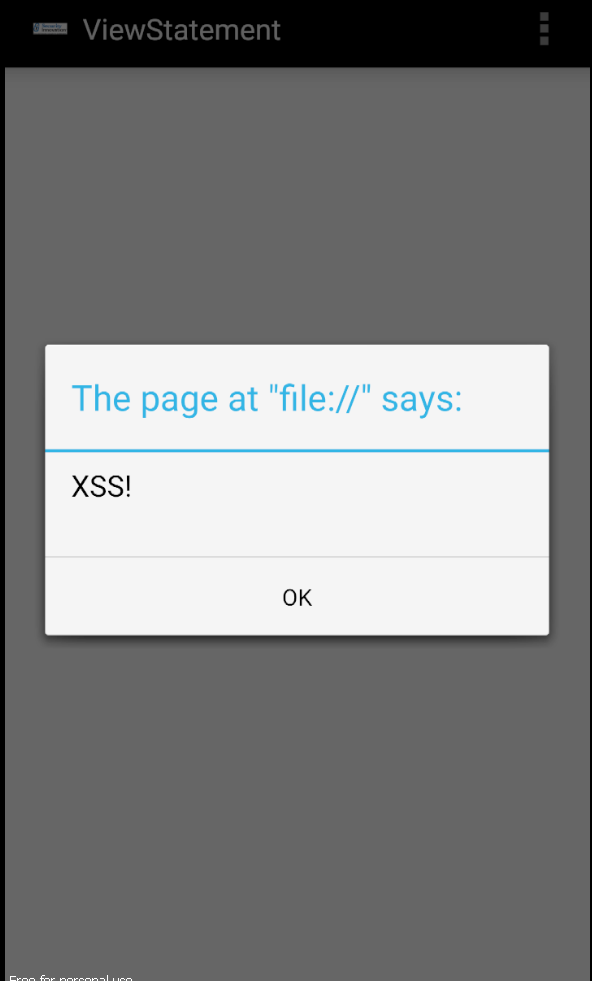
Make a file named Statements\_jack.html contain this code:

<script>alert("XSS!");</script>

Send that file to the device using adb



Login and go to the View Statement, we will see the alert notification



***Exploitation Tool:***

adb, Genymotion, jadx, Test Manual

***Recommendation:***

Set the permission to prevent the attacker exploits the content provider vulnerable

***References****:*

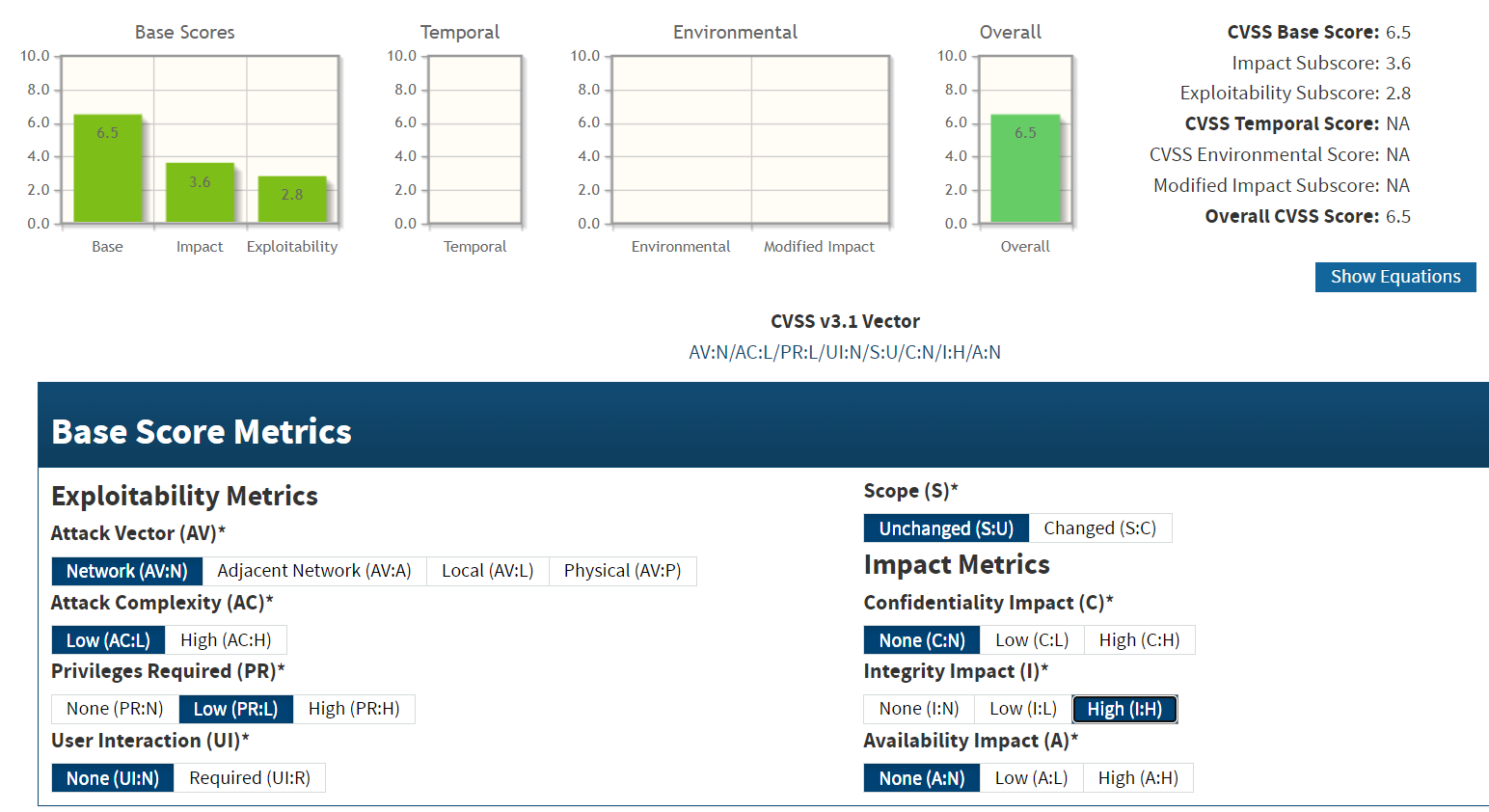
[Exploiting Content Providers - HackTricks](https://book.hacktricks.xyz/mobile-pentesting/android-app-pentesting/drozer-tutorial/exploiting-content-providers)

[Content provider basics  |  Android Developers](https://developer.android.com/guide/topics/providers/content-provider-basics)

### **4.12 Parameter Manipulation**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **HIGH** | **Technical Impact** | **MEDIUM** | **Business Impact** | **MEDIUM** |



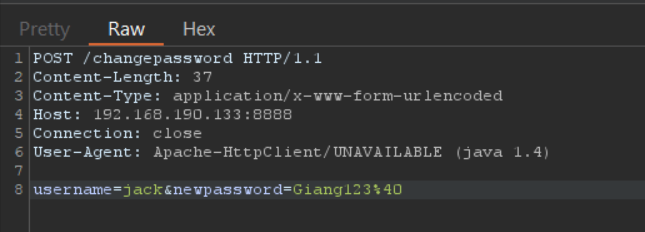
***Description:***

This is a type of vulnerability that occurs when an attacker is able to alter the values of parameters in a URL or Web form, in order to change the outcome of an application. This manipulation can be used to access restricted data, execute arbitrary commands, or cause a denial of service.

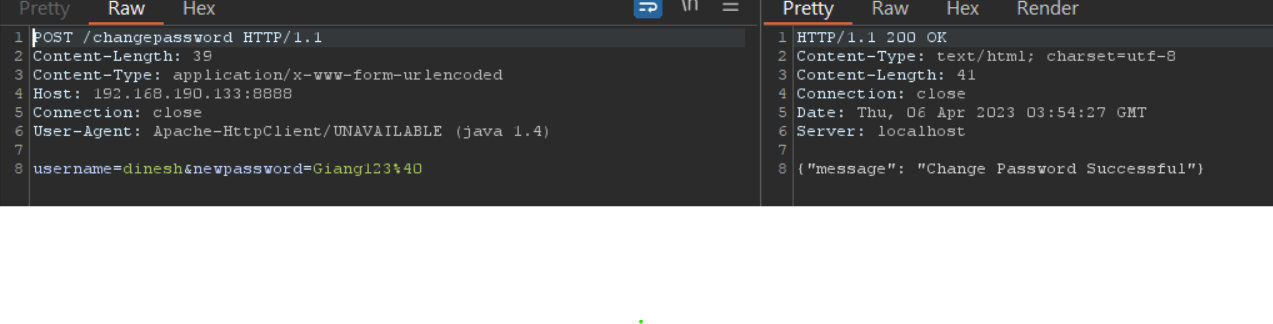
In this case, the attacker can change the password of another user.

***Proof of Concept:***

Login as a valid credential: jack:Jack@123$

Set up Burp Suite’s proxy to intercept the request. Use the Change Password functionality. Then looking at burp’s proxy. We can see a request to change password is sending to the server

Changing the value of the username to dinesh, the password is changed for user dinesh.



***Exploitation Tool:***

Burp Suite, adb, Genymotion, jadx, Test Manual

***Recommendation:***

Apply some authentication mechanism (jwt, session,…) to the server to check for which user is requesting to change the password

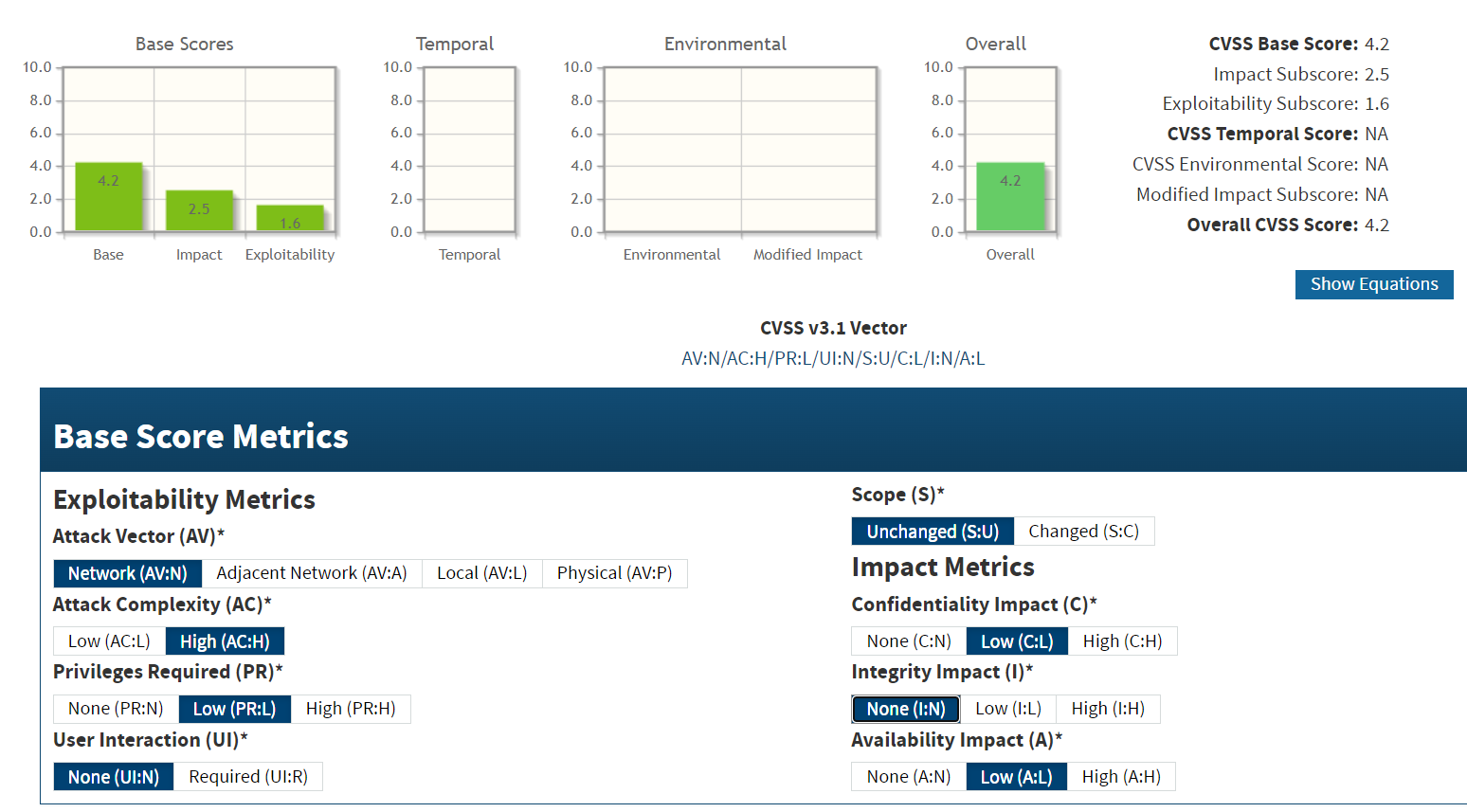
***References****:*

[Web Parameter Tampering | OWASP Foundation](https://owasp.org/www-community/attacks/Web_Parameter_Tampering)

### **4.13 Username Enumeration**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **MEDIUM** | **Technical Impact** | **MEDIUM** | **Business Impact** | **MEDIUM** |



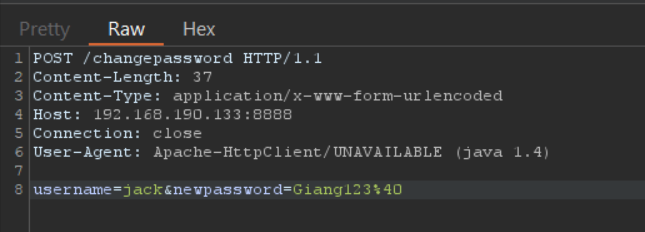
***Description:***

This is a type of security flaw that allows attackers to identify valid user accounts on a system. This can be used to gain access to data or attempt to guess passwords, leading to a breach of security. In some cases, even if the accounts are not valid, the same results may be returned, allowing the attacker to gain information on valid usernames without even needing to guess.

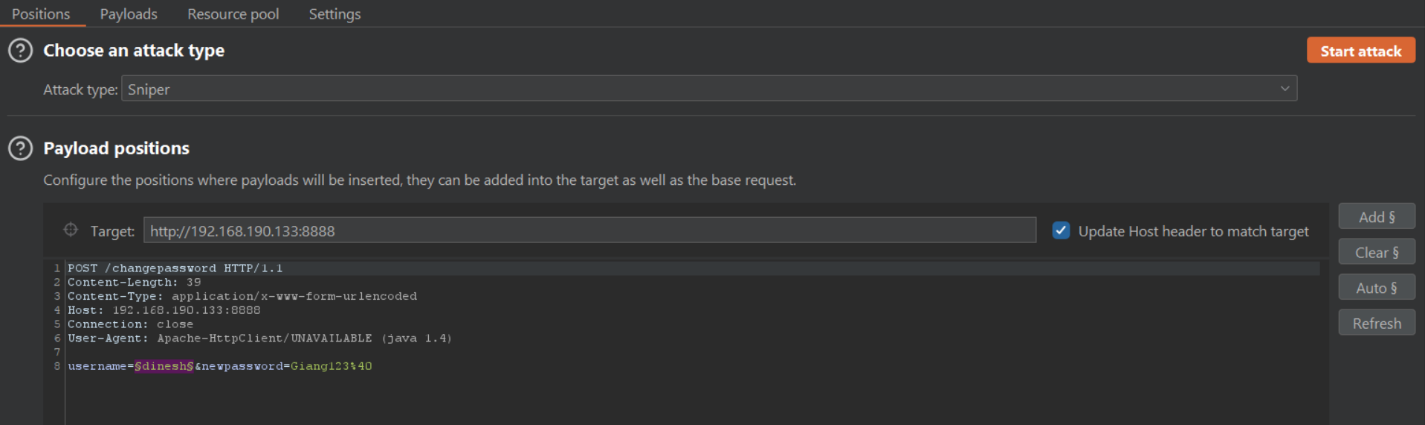
In this case, the attacker can utilize the change password feature to get the list of valid username.

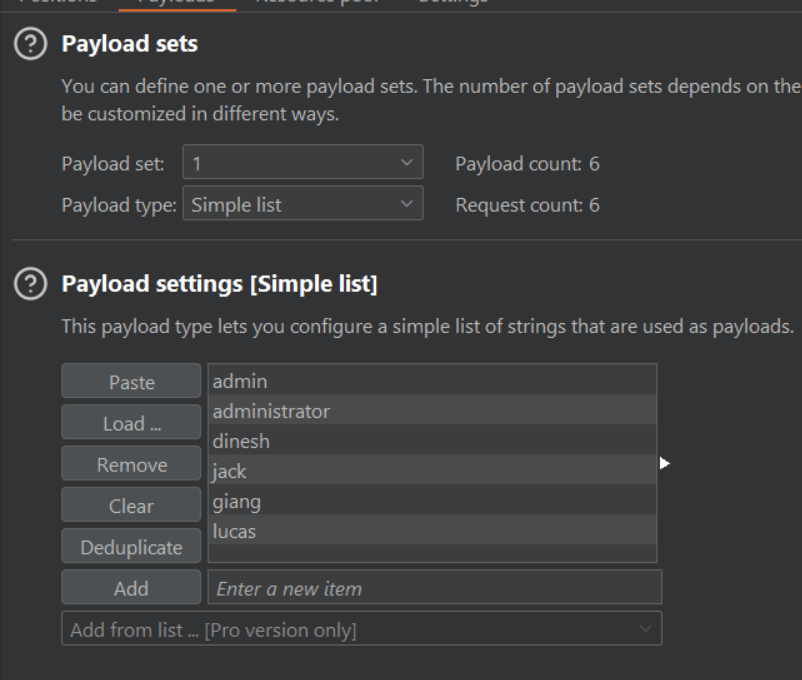
***Proof of Concept:***

Login as a valid credential: jack:Jack@123$

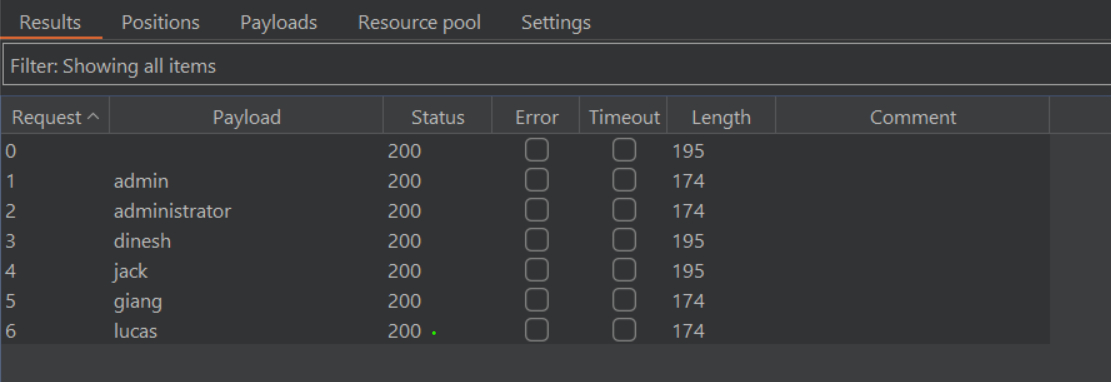
Set up Burp Suite’s proxy to intercept the request. Use the Change Password functionality. Then looking at burp’s proxy. We can see a request to change password is sending to the server

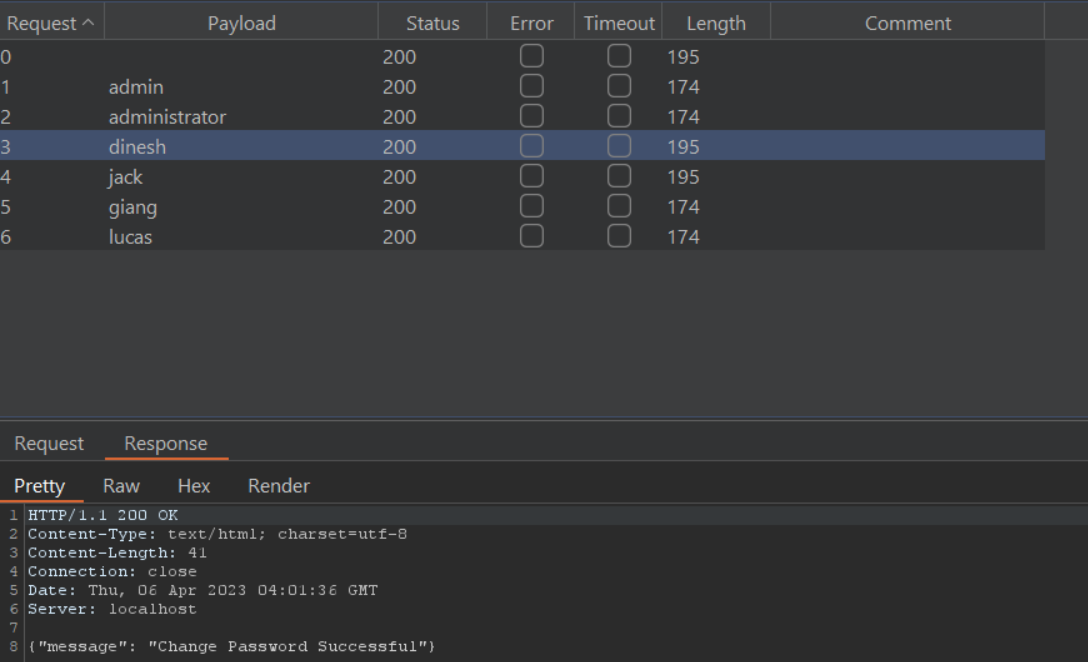
  Use Burp Intruder to enumerate the usernames.

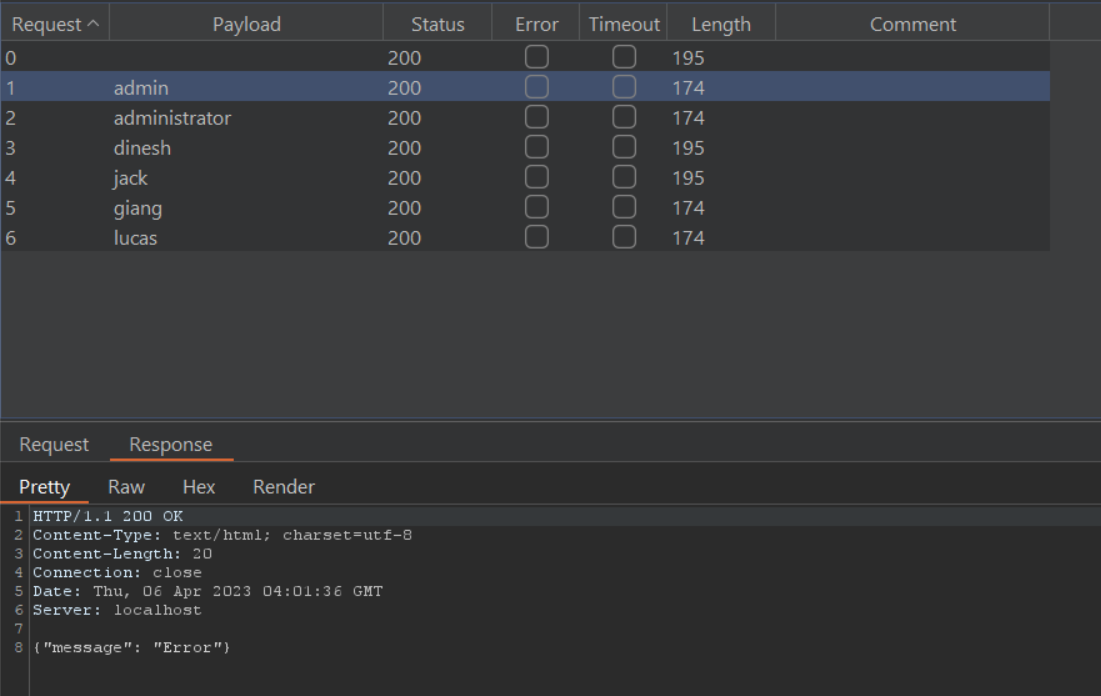




Look at the result, we can see that there are two valid username: dinesh and jack







***Exploitation Tool:***

Burp Suite, adb, Genymotion, jadx, Test Manual

***Recommendation:***

Use only one generic error message when change password fail: ‘Invalid Information’ for example

***References****:*

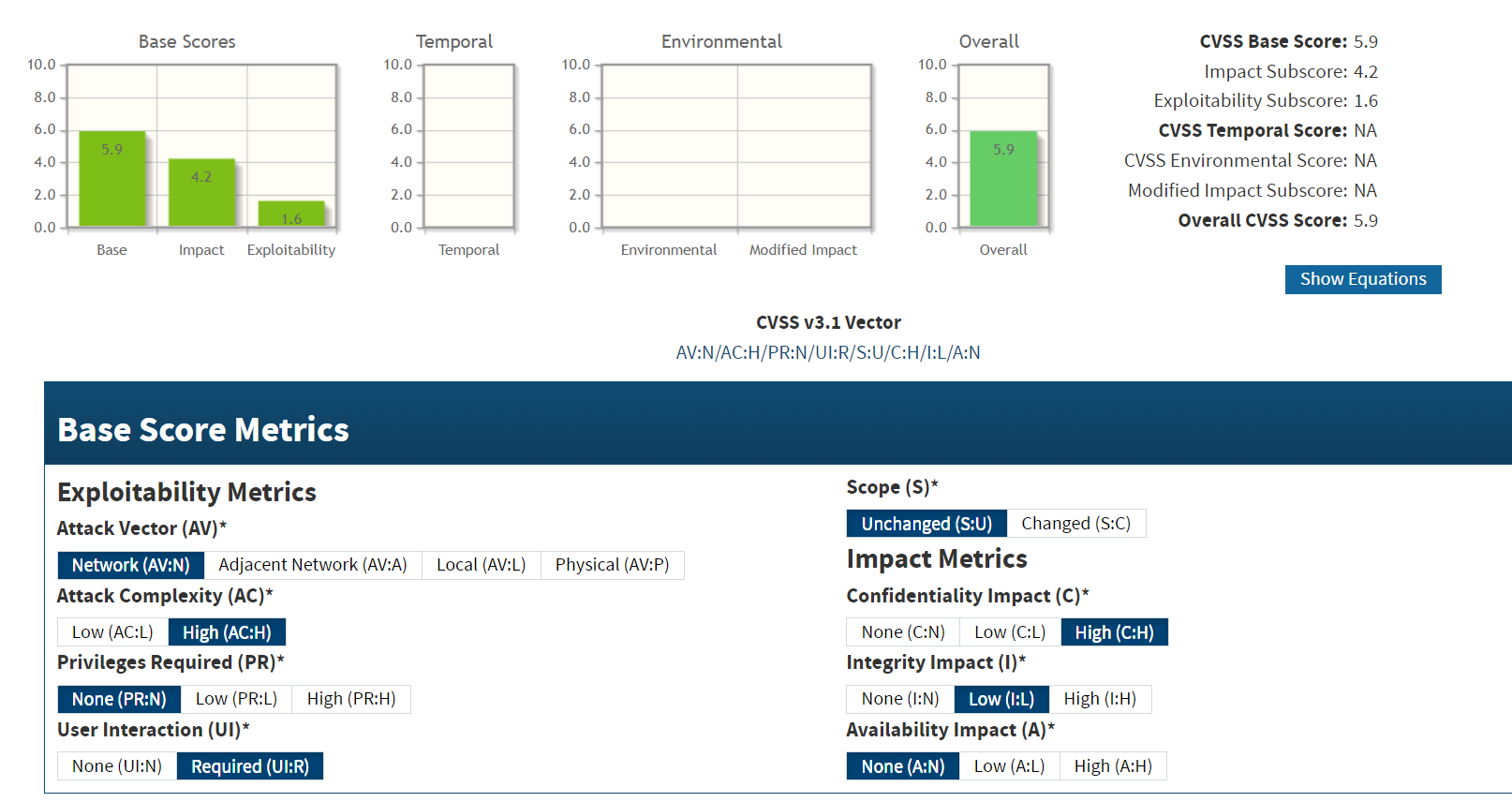
[User Enumeration Explained: Techniques and Prevention Tips | Rapid7 Blog](https://www.rapid7.com/blog/post/2017/06/15/about-user-enumeration/)

[Username Enumeration - Virtue Security](https://www.virtuesecurity.com/kb/username-enumeration/)

### **4.14 Insecure HTTP Connections**

***The following summaries the vulnerability’s severity ratings.***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Possibility** | **LOW** | **Technical Impact** | **HIGH** | **Business Impact** | **MEDIUM** |

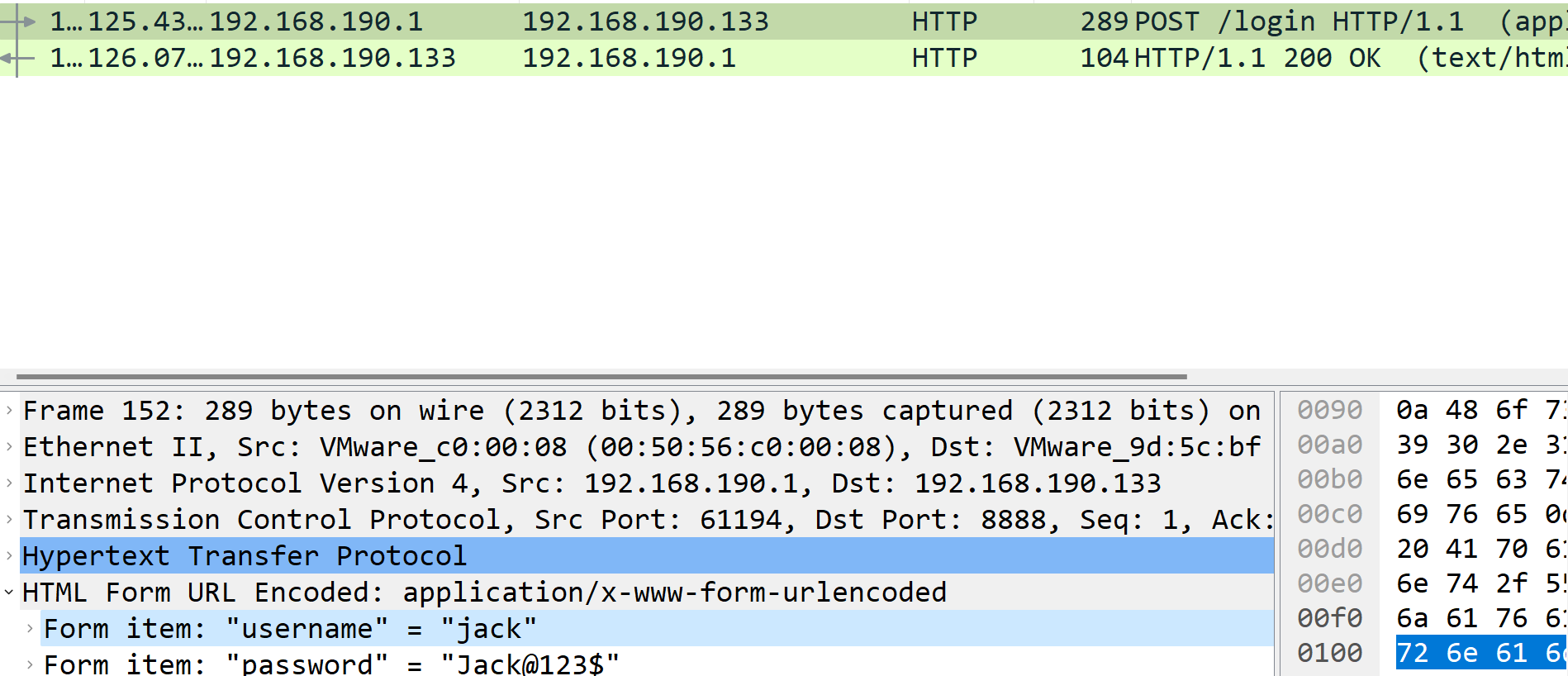


***Description:***

Using HTTP as an unencrypted communication channel is a security vulnerability that arises when an application transmits sensitive data, such as usernames, passwords, and other sensitive information, in plain text over the network. This allows an attacker to intercept the traffic and read or modify the data, compromising the confidentiality and integrity of the communication.

***Proof of Concept:***

When a user logs in, using Wireshark to capture the http request packet, we can clearly see the user’s credential



***Exploitation Tool:***

Wire Shark, Genymotion, jadx, Test Manual

***Recommendation:***

Use HTTPS (HTTP over SSL/TLS) to encrypt the network communication and protect the sensitive data transmitted over the network.

Disable HTTP

***References****:*

[Why is HTTP not secure? | HTTP vs. HTTPS | Cloudflare](https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/)