

# Project Report

14 oct. 2024

Ayush Kumar Thakur

Mentor.- Kirankumar S.  
(CSI)

# **CYBER SECURED INDIA (CSI)**

## **Vulnerability Report for DIVA Android App Challenges**

*A Comprehensive Analysis of Vulnerabilities in the DIVA Android App Challenges*

**Ayush Kumar Thakur**  
**14-october-2024**

*This report highlights vulnerabilities in the DIVA Android app challenges, covering insecure logging, hardcoded secrets, insecure data storage, input validation issues, and access control problems. Each issue is rated for severity, impact, steps to reproduce, PoC, remediation suggestions, and CWE identifiers. The report underscores the need for secure coding practices and thorough security testing in Android app development, offering valuable insights for developers and security professionals.*

**Let's Start**  

# Challenge 1: Insecure Logging.

## • Title & Severity:

- **Title: Insecure Logging**
- **Severity: High**

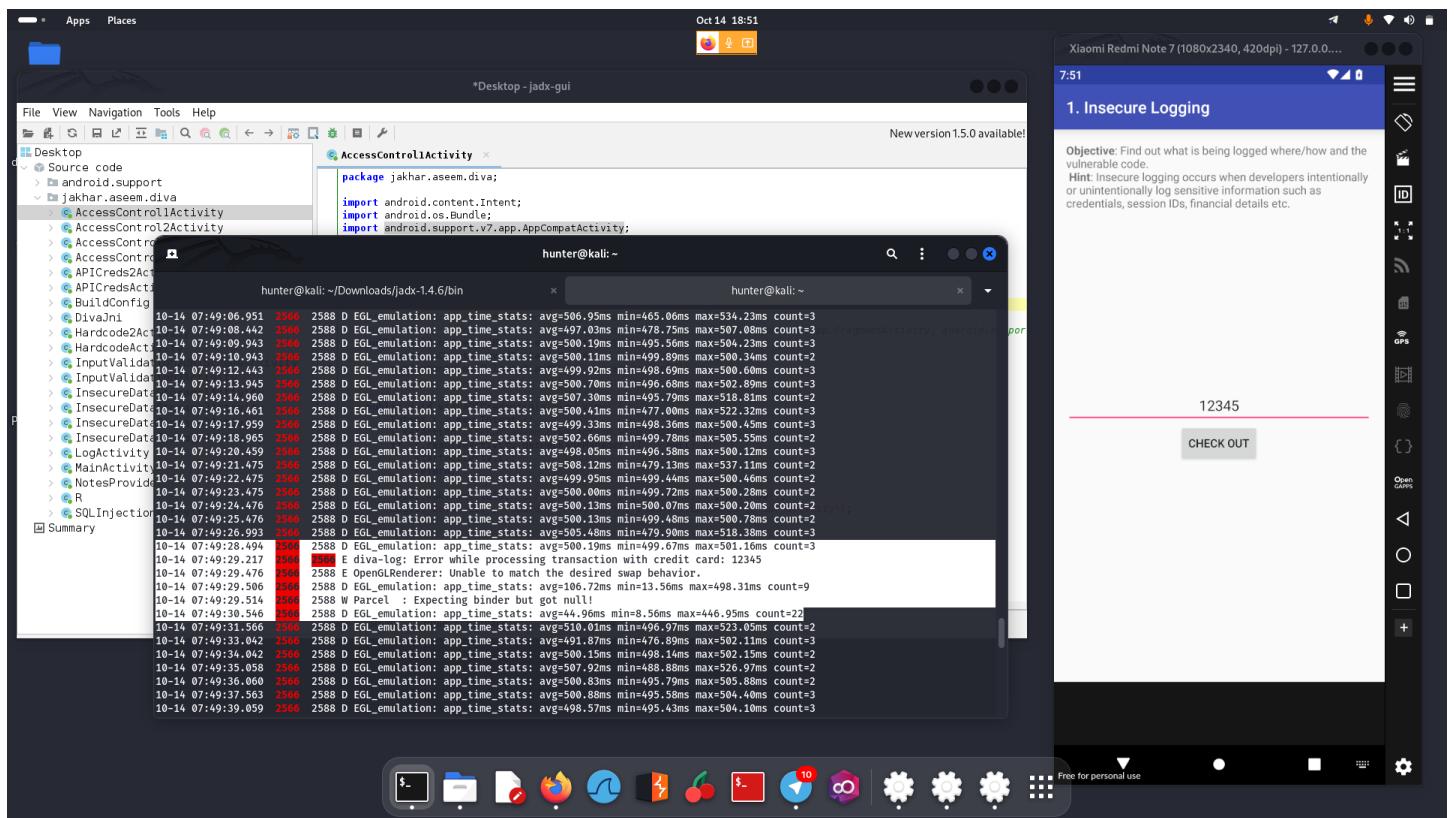
• **Description:** The application logs sensitive data, such as credit card information, without proper protection. This data is logged in plain text.

• **Impact:** The impact of this vulnerability is significant. It exposes users' credit card information to potential attackers and could lead to unauthorized access to sensitive data.

## • Steps to Reproduce:

1. Enter data into the application and trigger the "check out" action.
2. Observe the logcat output for logged data.

## • PoC (Proof of Concept):



E/diva-log( 6067): Error while processing transaction with credit card: 4711

## • Remediation:

- Avoid logging sensitive data.
- Implement proper access controls and encryption for log files.
- Educate developers on secure logging practices.

- **CWE (Common Weakness Enumeration):**

- CWE-532: Inclusion of Sensitive Information in Log Files

## Challenge 2: Hardcoding Issues - Part 1

- **Title & Severity:**

- **Title:** Hardcoding Issues - Part 1
- **Severity:** Medium

- **Description:** The application contains a hardcoded secret key, "**vendorsecretkey**," which provides access to restricted areas.

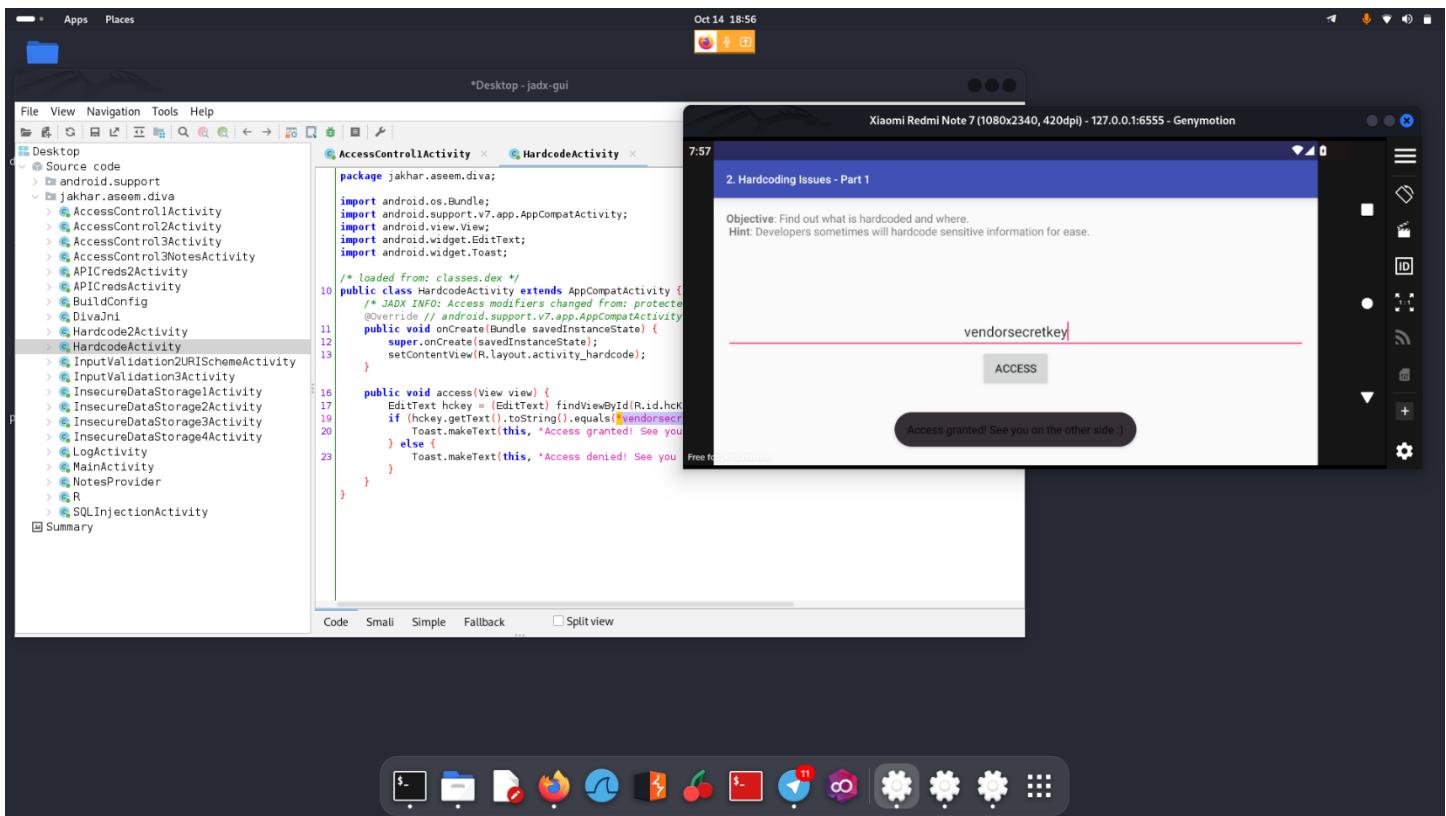
- **Impact:** The impact is moderate. Attackers who discover this hardcoded key can gain unauthorized access to restricted parts of the application.

- **Steps to Reproduce:**

1. Examine the source code of the application.
2. Identify the presence of the "**vendorsecretkey**."
3. Use the key to access restricted areas.

- **PoC (Proof of Concept):**

```
if (((EditText)this.findViewById(2131492987)).getText().toString().equals("vendorsecretkey")) {  
    Toast.makeText((Context)this, (CharSequence)"Access granted! See you on the other side :)",  
    (int)0).show();    return;  
}
```



- **Remediation:**

- Avoid hardcoding sensitive information like secret keys in the source code.
- Use secure storage mechanisms or environment variables for such data.
- Implement proper access controls.

- **CWE (Common Weakness Enumeration):**

- CWE-798: Use of Hard-coded Credentials

## Challenge 3: Insecure Data Storage - Part 1

- **Title & Severity:**

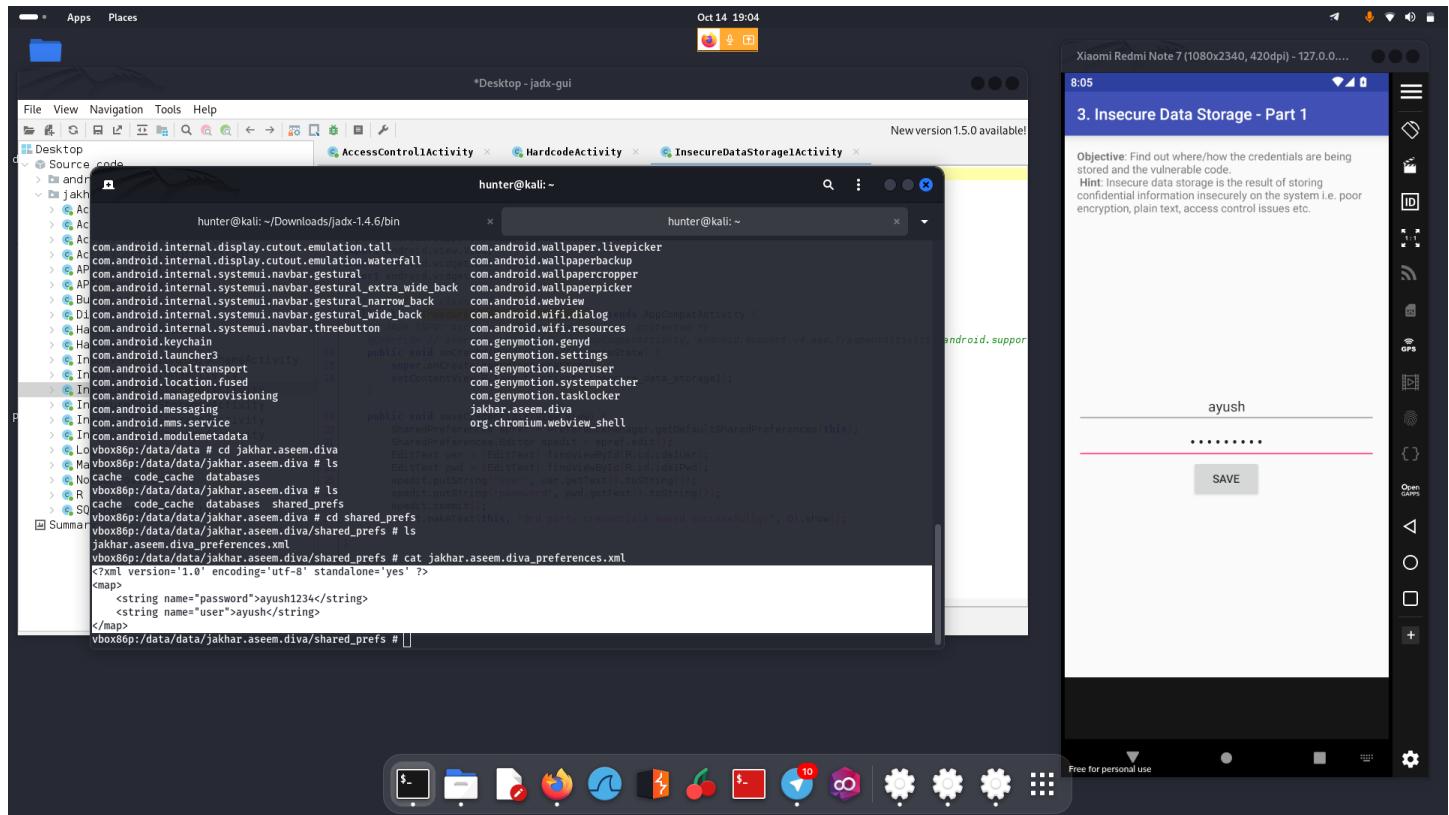
- **Title:** Insecure Data Storage - Part 1
- **Severity:** Medium

- **Description:** The application stores user credentials in plain text in the shared preferences folder.
- **Impact:** The impact is moderate. It exposes user credentials to potential attackers who have access to the device, risking unauthorized access.

- **Steps to Reproduce:**

1. Enter credentials into the application.
2. Review the source code and discover the use of shared preferences.
3. Use ADB shell to access the shared preferences folder.

- **PoC (Proof of Concept):**



- Using ADB shell to access shared preferences:

Bash adb shell cd

```
/data/data/jakhar.aseem.diva/shared_prefs
```

- **Remediation:**

- Avoid storing sensitive data in plain text in shared preferences.
- Implement encryption or hashing for stored credentials.
- Educate developers on secure data storage practices.

- **CWE (Common Weakness Enumeration):**

- CWE-312: Cleartext Storage of Sensitive Information

## Challenge 4: Insecure Data Storage - Part 2

- **Title & Severity:**

- **Title:** Insecure Data Storage - Part 2
- **Severity:** Medium

- **Description:** The application stores user credentials in a SQLite database without encryption.

- **Impact:** The impact is moderate. It exposes user credentials to potential attackers with access to the device, risking unauthorized access.

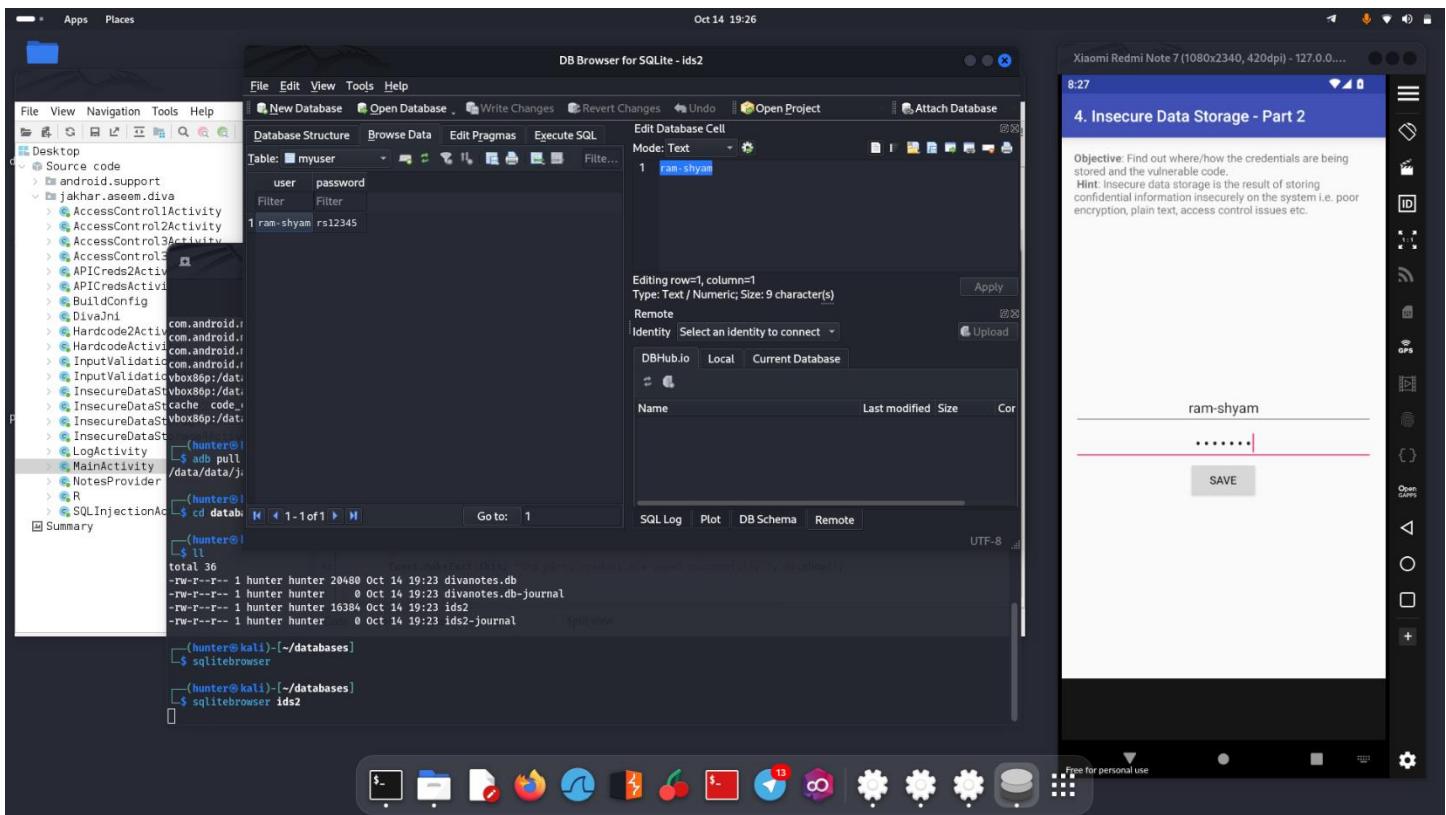
- **Steps to Reproduce:**

1. Enter credentials into the application.
2. Review the source code and discover the use of an SQLite database.
3. Use ADB shell to access and extract the database file.

- **PoC (Proof of Concept):**

- **Using ADB shell to pull the database:**

**adb pull /data/data/jakhar.aseem.diva/databases/ids .**



## • Remediation:

- Implement encryption or hashing for stored credentials in the database.
- Avoid storing sensitive data in plain text in databases.
- Educate developers on secure data storage practices.

## • CWE (Common Weakness Enumeration):

- CWE-313: Cleartext Storage in a File or on Disk

# Challenge 5: Insecure Data Storage - Part 3

## • Title & Severity:

- **Title:** Insecure Data Storage - Part 3
- **Severity:** Medium

- **Description:** The application stores user credentials in a temporary file without proper protection.

- **Impact:** The impact is moderate. It exposes user credentials to potential attackers with access to the device, risking unauthorized access.

- **Steps to Reproduce:**

1. Enter credentials into the application.
2. Review the source code and discover the use of temporary file creation.
3. Use ADB shell to locate and access the temporary file.

- **PoC (Proof of Concept):**

- **Using ADB shell to access the temporary file:**

**bash adb shell cat /data/data/jakhar.aseem.diva/app\_uinfo-838734269tmp**

The screenshot shows a dual-monitor setup. The left monitor displays the Jadx-GUI interface, which is a decompiler for Android applications. It shows the project structure for 'InsecureDataStorage2Activity' and 'InsecureDataStorage3Activity'. The code for 'InsecureDataStorage3Activity' is visible, showing imports for Log, View, EditText, Widget, Toast, File, and FileWriter, along with a public class definition extending AppCompatActivity. The right monitor displays a Xiaomi Redmi Note 7 smartphone screen with the title '5. Insecure Data Storage - Part 3'. The app interface includes a text input field with placeholder 'Objective: Find out where/how the credentials are being stored and the vulnerable code.' and a note 'Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.'. Below the input field is a password field with the placeholder 'hunter' and a redacted password entry. A 'SAVE' button is present. The bottom of the phone screen shows various notification icons.

```

import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.File;
import java.io.FileWriter;

/* Loaded from: classes.dex */
public class InsecureDataStorage3Activity extends AppCompatActivity {
    ...
}

$ adb shell
vbox86p:/ # cd /data/data/jakhar.aseem.diva/
vbox86p:/data/data/jakhar.aseem.diva # ls
cache  code_cache  databases  shared_prefs  uinfo2476000315759048188tmp
vbox86p:/data/data/jakhar.aseem.diva # ls -al
total 36
drwxr-x-x  6 u0_a87 u0_a87  4096 2024-10-16 08:29 .
drwxrwx-x 123 system system 12288 2024-10-16 07:32 ..
drwxrws-x  2 u0_a87 u0_a87 cache  4096 2024-10-16 07:32 cache
drwxrws-x  2 u0_a87 u0_a87 code  4096 2024-10-16 07:32 code
drwxrws-x  2 u0_a87 u0_a87 databases 4096 2024-10-16 07:32 databases
drwxrws-x  2 u0_a87 u0_a87 shared_prefs 4096 2024-10-16 07:32 shared_prefs
-rw-r--r--  1 u0_a87 u0_a87 19 2024-10-16 08:29 uinfo2476000315759048188tmp
vbox86p:/data/data/jakhar.aseem.diva # cat uinfo2476000315759048188tmp
hunter:hunter12345
vbox86p:/data/data/jakhar.aseem.diva # []

```

- **Remediation:**

- Avoid storing sensitive data in plain text in temporary files.
- Implement encryption or hashing for stored credentials.
- Educate developers on secure data storage practices.

- **CWE (Common Weakness Enumeration):**

- CWE-314: Cleartext Storage of Sensitive Information in a File

## Challenge 6: Insecure Data Storage - Part 4

- **Title & Severity:**

- **Title:** Insecure Data Storage - Part 4
  - **Severity:** Medium

- **Description:** The application stores user credentials in a local file on the SD card without proper protection.

- **Impact:** The impact is moderate. It exposes user credentials to potential attackers with access to the device's SD card, risking unauthorized access.

- **Steps to Reproduce:**

1. Enter credentials into the application.
2. Review the source code and discover the use of an SD card file.
3. Use ADB shell to locate and access the file on the SD card.

- **PoC (Proof of Concept):**

- **Using ADB shell to access the file on the SD card:**

```
bash adb shell cat /sdcard/.uinfo.txt
```

The screenshot shows a Kali Linux desktop environment with several windows open. In the foreground, a terminal window titled 'hunter@kali: ~' displays a file listing from the command line:

```
total 56
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17:16.464000000 +0000 Documents
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17:08.872000000 +0000 Downloads
drwxrws--- 3 u0_a82 media_rw 4096 2024-10-13 18:17:16.723000000 +0000 Movies
drwxrws--- 3 u0_a82 media_rw 4096 2024-10-13 18:17:16.722000000 +0000 Music
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17:16.456000000 +0000 Notifications
drwxrws--- 3 u0_a82 media_rw 4096 2024-10-13 18:17:16.724000000 +0000 Pictures
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17:16.456000000 +0000 Podcasts
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17:16.466000000 +0000 Recordings
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17:16.456000000 +0000 Ringtones
drwxrws--- 1 u0_a82 media_rw 4096 2024-10-14 08:43 uninfo.txt
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17:17 Albums
drwxrws--- 5 media_rw 4096 2024-10-13 23:46 Android
drwxrws--- 1 media_rw 4096 2024-10-13 18:17:17 Audiobooks
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17:17 DCIM
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17 Documents
drwxrws--- 3 u0_a82 media_rw 4096 2024-10-13 18:17 Downloads
drwxrws--- 3 u0_a82 media_rw 4096 2024-10-13 18:17 Movies
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17 Music
drwxrws--- 3 u0_a82 media_rw 4096 2024-10-13 18:17 Notifications
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17 Pictures
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17 Podcasts
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17 Recordings
drwxrws--- 2 u0_a82 media_rw 4096 2024-10-13 18:17 Ringtones
```

Below the terminal, a command is run: `vbox86p:/sdcard # cat .uninfo.txt`. The output is: `cat: .uninfo.txt: No such file or directory`.

In the background, a jadx-gui window shows the decompiled code of 'InsecureDataStorage2Activity'. The code imports android.util.Log, android.view.View, android.widget.EditText, and android.widget.Toast.

A mobile application interface window titled '6. Insecure Data Storage - Part 4' is also visible, displaying a note about insecure data storage.

## • Remediation:

- Avoid storing sensitive data in plain text on the SD card.
- Implement encryption or hashing for stored credentials.
- Educate developers on secure data storage practices.

## • CWE (Common Weakness Enumeration):

- CWE-313: Cleartext Storage in a File or on Disk

# Challenge 7: Input Validation Issues - Part 1

## • Title & Severity:

- **Title:** Input Validation Issues - Part 1
- **Severity:** Medium

• **Description:** The application does not properly validate user input when searching for users in the database, potentially exposing sensitive data.

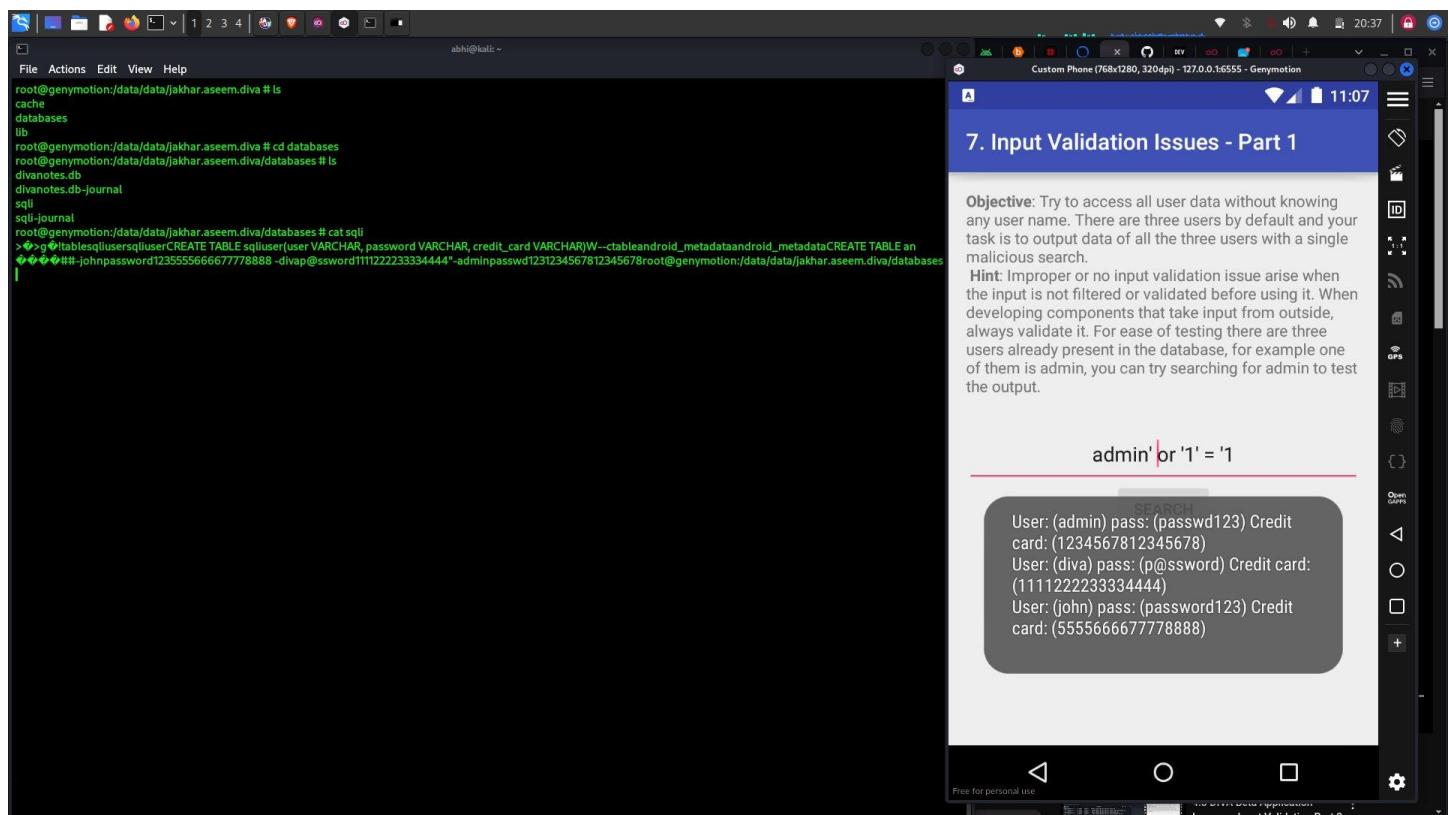
• **Impact:** The impact is moderate. An attacker can exploit this vulnerability to retrieve user data without proper authorization.

## • Steps to Reproduce:

1. Enter a malicious input (e.g., a single quote) into the search field.
2. Observe the error message or the database query in logcat.

## • PoC (Proof of Concept):

### Using a single quote as input: '1' or '1' != '2



## • Remediation:

- Implement proper input validation and sanitization to prevent SQL injection.
- Avoid executing user input as part of database queries.
- Educate developers on secure coding practices.

- **CWE (Common Weakness Enumeration):**

- CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

## Challenge 8: Input Validation Issues - Part 2

- **Title & Severity:**

- **Title:** Input Validation Issues - Part 2
- **Severity:** Medium

- **Description:** The application allows users to access sensitive information by entering a specific URL into the search field.

- **Impact:** The impact is moderate. It enables users to access potentially confidential data via URL manipulation.

- **Steps to Reproduce:**

1. Enter a specific URL (**e.g., "file:///sdcard/.uinfo.txt"**) into the search field.
2. Observe the retrieval of sensitive information.

- **PoC (Proof of Concept):**

**using the URL "file:///sdcard/.uinfo.txt" to access the file.**

The screenshot shows a Kali Linux terminal window titled "hunter@kali: ~" displaying a file listing from "/Downloads/adx-1.4.6/bin". The listing includes files like "debug\_ramdisk", "dev", "etc", "init", "init.environ.rc", "linkerconfig", "lost+found", "mnt", "odm", "odm\_dlm", "oem", "postinstall", "product", "sbin", "sdcard", "second\_stage\_resources", "storage", "sys", "system", "system\_dlm", "tmp", "vendor", and "vendor\_dlm". The terminal also shows code snippets related to Java and Android development, including a class named "SQLInjectionActivity" and a method involving a WebView.

On the right, there is a mobile application interface titled "8. Input Validation Issues - Part 2" running on a "Xiaomi Redmi Note 7 (1080x2340, 420dpi) - 127.0.0...." device. The app displays an objective: "Objective: Try accessing any sensitive information apart from a web URL." It includes a hint: "Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it." A red box highlights the URL "file:///data/data/jakhar.aseem.diva/uinfo24760" in the app's input field.

## • Remediation:

- Implement proper input validation to prevent unauthorized access to sensitive data via URL manipulation.
- Educate developers on secure coding practices.

## • CWE (Common Weakness Enumeration):

- CWE-264: Permissions, Privileges, and Access Controls

# Challenge 9: Access Control Issues - Part 1

- **Title & Severity:**

- **Title:** Access Control Issues - Part 1
- **Severity:** High

- **Description:** The application allows unauthorized access to a specific activity without proper authentication or access control.

- **Impact:** The impact is high. Attackers can gain unauthorized access to a specific activity without navigating through the normal user interface.

- **Steps to Reproduce:**

1. Discover the activity name ("jakhar.aseem.diva.APICredsActivity").
2. Use ADB shell or other means to directly start the activity.

- **PoC (Proof of Concept):**

arduino run app.activity.start --component jakhar.aseem.diva jakhar.aseem.diva.APICredsActivity

The screenshot shows a terminal window titled "hunter@kali: ~" running on a Kali Linux system. The user has run the command "adb shell am start -n jakhar.aseem.diva.action.VIEW\_CREDS" and is observing the output in the terminal. To the right of the terminal, a mobile device screen displays the "Vendor API Credentials" application with the API Key set to "123secretapikey123", API User name to "diva", and API Password to "p@ssword". The terminal also shows the Java code for the MainActivity class, which contains a method for handling API credentials.

```
--grant-persistent-uri-permission] [--grant-prefix-uri-permission]
[--debug-log-resolution] [--exclude-stopped-packages]
[--include-stopped-packages]
[--activity-brought-to-front] [--activity-clear-top]
[--activity-clear-when-task-reset] [--activity-exclude-from-recents]
[--activity-launched-from-history] [--activity-multiple-task]
[--activity-no-animation] [--activity-no-history]
[--activity-no-user-action] [--activity-previous-is-top]
[--activity-reorder-to-front] [--activity-reset-task-if-needed]
[--activity-single-top] [--activity-clear-task]
[--activity-task-on-home] [--activity-match-external]
[--receiver-registered-only] [--receiver-replace-pending]
[--receiver-foreground] [--receiver-no-abort]
[--receiver-include-background]
[--selector]
[URI] | <PACKAGE> | <COMPONENT>
<...>
<...>
<...>
(hunter@kali)-[~]
$ adb shell am start -n jakhar.aseem.diva.action.VIEW_CREDS
Starting: Intent { act=jakhar.aseem.diva.action.VIEW_CREDS }
(hunter@kali)-[~]
$ 
InsecureDataStorage2Activity
InsecureDataStorage3Activity
InsecureDataStorage4Activity
LogActivity
MainActivity
NotesProvider
R
SQLInjectionActivity
Summary
public void viewAPICredentials(View view) {
    Intent i = new Intent();
    i.setAction(jakhar.aseem.diva.action.VIEW_CREDS);
    if (!i.resolveActivity(getApplicationContext()) != null) {
        startActivity(i);
        return;
    }
    Toast.makeText(this, "Error while getting API details", 0).show();
    Log.e("Diva-ac11", "Couldn't resolve the Intent VIEW_CREDS to our activity");
}
```

- **Remediation:**

- Implement proper access controls and authentication mechanisms for activities.
- Ensure that sensitive functionalities are not accessible without proper authorization.
- Educate developers on secure coding practices.

- **CWE (Common Weakness Enumeration):**

- CWE-285: Improper Access Control

## Challenge 10: Access Control Issues - Part 2

- **Title & Severity:**

- **Title:** Access Control Issues - Part 2
- **Severity:** High

- **Description:** The application allows unauthorized access to another specific activity with a PIN-based protection without validating the PIN.

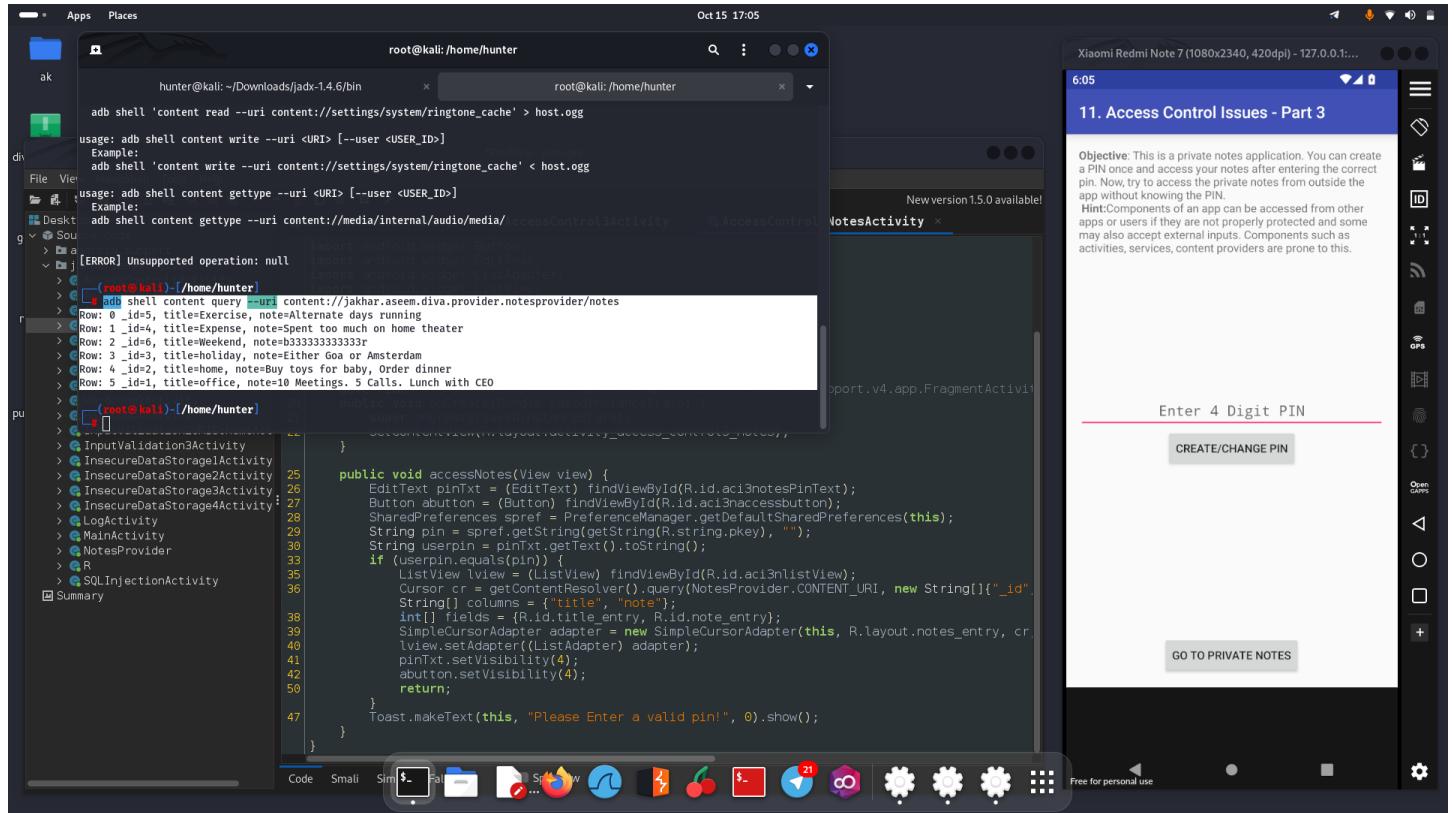
- **Impact:** The impact is high. Attackers can gain unauthorized access to a specific activity by bypassing the PIN protection.

- **Steps to Reproduce:**

1. Discover the activity name ("jakhar.aseem.diva.APICreds2Activity").
2. Observe that the PIN protection is not properly validated.
3. Use an intent to directly start the activity without a valid PIN.

- **PoC (Proof of Concept):**

```
arduino run app.activity.start --component jakhar.aseem.diva jakhar.aseem.diva.APICreds2Activity --  
extra boolean check_pin false
```



## • Remediation:

- Implement proper PIN validation for activities that require it.
- Ensure that sensitive functionalities are protected by strong access controls.
- Educate developers on secure coding practices.

## • CWE (Common Weakness Enumeration):

- CWE-306: Missing Authentication for Critical Function

# Challenge 11: Access Control Issues - Part 3

## • Title & Severity:

- **Title:** Access Control Issues - Part 3
- **Severity:** High

• **Description:** The application allows unauthorized access to a notes storage without requiring the PIN.

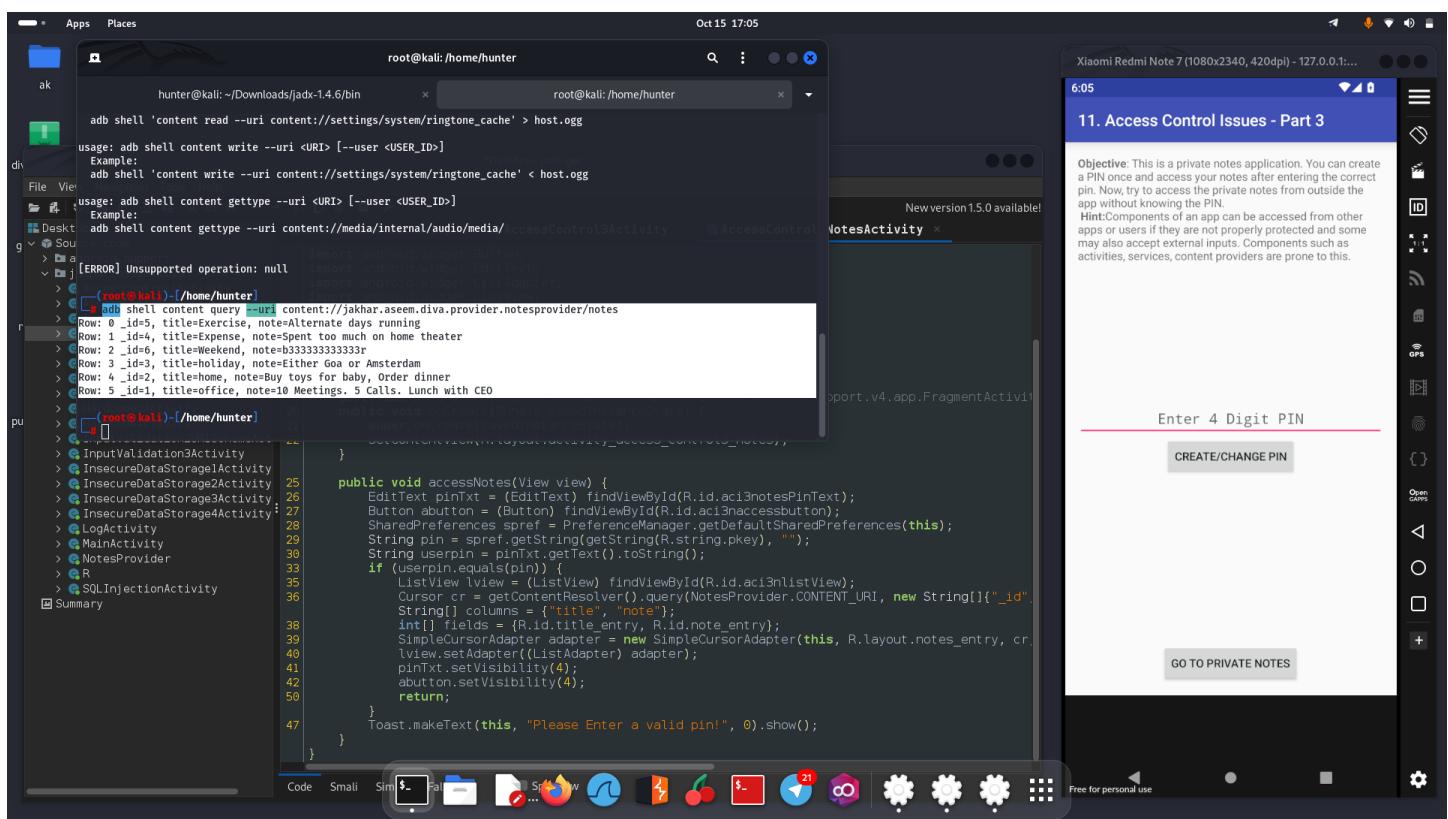
• **Impact:** The impact is high. Attackers can gain unauthorized access to sensitive notes without requiring the PIN.

## • Steps to Reproduce:

1. Discover the exported content provider ("content://jakhar.aseem.diva.provider.notesprovider/notes").
2. Observe that the content provider does not enforce the PIN requirement.
3. Query the content provider to retrieve the notes.

## • PoC (Proof of Concept):

arduino run app.provider.query content://jakhar.aseem.diva.provider.notesprovider/notes/ --projection "\* FROM notes;--"



- **Remediation:**

- Ensure that access to sensitive data is properly controlled and authenticated.
- Implement proper access controls and validation for content providers.
- Educate developers on secure coding practices

- **CWE (Common Weakness Enumeration):**

- CWE-284: Improper Access Control

## Challenge 12: Hardcoding Issues - Part 2

- **Title & Severity:**

- **Title:** Hardcoding Issues - Part 2
- **Severity:** Medium

- **Description:** The application loads a native library that contains a suspicious string "olsdfgad;lh."

- **Impact:** The impact is moderate. Attackers who discover this hardcoded string may have access to sensitive information or functionality.

- **Steps to Reproduce:**

1. Examine the native library used by the application.
2. Discover the suspicious hardcoded string.

## • PoC (Proof of Concept):

Use a strings command to examine the native library for the hardcoded string.

The screenshot shows two windows side-by-side. On the left is a terminal or browser window showing the GitHub URL <https://github.com/payatu/diva-android/blob/master/app/src/main/jni/divajni.c>. The code in the file includes several hard-coded strings, notably `VENDORKEY`, `CODE`, and `CODESIZEMAX`. On the right is a mobile application interface for a challenge titled "12. Hardcoding Issues - Part 2". The app displays the objective: "Find out what is hardcoded and where. Hint: Developers sometimes will hardcode sensitive information for ease." A text input field contains the string "olsdfgad;ih", and a button labeled "ACCESS" is visible. A message at the bottom says "Access granted! See you on the other side :)".

```

22 * BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
23 * PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
24 * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
25 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
26 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
27 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
28 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
29 * POSSIBILITY OF SUCH DAMAGE.
30 *
31 */
32
33 #include <jni.h>
34 #include <string.h>
35 #include "divajni.h"
36
37 #define VENDORKEY "1111111111111111"
38 #define CODE ".dotdot"
39 #define CODESIZEMAX 20
40 /*
41 * Verify the key for access
42 *
43 * @param jkey The key input by user
44 *
45 * @return 1 if jkey is valid, 0 otherwise. In other words
46 * 0 if the user key matches our key return 1, else return 0.
47 */
48 JNIEXPORT jint JNICALL Java_jakhar_aseem_diva_DivaJni_access(JNIEnv * env, jobject job, jstring jkey) {
49
50     const char * key = (*env)->GetStringUTFChars(env, jkey, 0);
51
52     return ((strcmp(VENDORKEY, key, strlen(VENDORKEY)))?0:1);
53 }
54
55 /*
56 * Launch the Friggin Nuke!
57 *
58 * @param code FINL launch code for the nuke

```

## • Remediation:

- Avoid hardcoding sensitive information in native libraries.
- Use secure storage mechanisms or environment variables for such data.
- Implement proper access controls and authentication.

## • CWE (Common Weakness Enumeration):

- CWE-798: Use of Hard-coded Credentials

# Challenge 13: Input Validation Issues - Part 3

- **Title & Severity:**

- **Title:** Input Validation Issues - Part 3
- **Severity:** Medium

- **Description:** The application crashes when it receives a very large input string, indicating inadequate input validation.

- **Impact:** The impact is moderate. Attackers can disrupt the application's functionality by sending oversized input strings.

- **Steps to Reproduce:**

1. Enter a very large input string (e.g., 30 times "A") into the application.

- **PoC (Proof of Concept):**

Using a very large input string to crash the application.

The screenshot illustrates the exploit development process. On the left, a terminal window on a Kali Linux system shows the command `./InputValidation3Activity` being run, which results in the error message "Diva keeps stopping". On the right, a Jadx-GUI window displays the Java source code of the `InputValidation3Activity`. The code contains logic for handling user input and launching a sequence if the input matches certain criteria. The error message "Diva keeps stopping" is shown in a toast notification on the Android device's screen, indicating that the application has crashed.

```
root@kali:~/Downloads/jadx-1.4.6/bin$ ./InputValidation3Activity
Diva keeps stopping
App info
Close app

Oct 15 17:13
root@kali: /home/hunter
root@kali: /home/hunter
*Desktop - Jadx-gui
File View Navigation Tools Help
New version 1.5.0 available!
Source code
InputValidation3Activity
jadx jahkar.aseem.diva;
android.os.Bundle;
android.support.v7.app.AppCompatActivity;
android.view.View;
android.widget.EditText;
android.widget.Toast;
added from: classes.dex */
13 public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_input_validation3);
    this.djni = new DivaJni();
}
JADX INFO: Access modifiers changed from: protected */
Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.R
14 public void push(View view) {
    EditText cTxt = (EditText) findViewById(R.id.iv13CodeText);
    if (this.djni.initiateLaunchSequence(cTxt.getText().toString()) != 0) {
        Toast.makeText(this, "Launching in T - 10 ...", 0).show();
    } else {
        Toast.makeText(this, "Access denied!", 0).show();
    }
}

Xiaomi Redmi Note 7 (1080x2340, 420dpi) - 127.0.0.1:...
6:14
Free for personal use
```

```
public class InputValidation3Activity extends AppCompatActivity {
    private DivaJni djni;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_input_validation3);
        this.djni = new DivaJni();
    }
    public void push(View view) {
        EditText cTxt = (EditText) findViewById(R.id.iv13CodeText);
        if (this.djni.initiateLaunchSequence(cTxt.getText().toString()) != 0) {
            Toast.makeText(this, "Launching in T - 10 ...", 0).show();
        } else {
            Toast.makeText(this, "Access denied!", 0).show();
        }
    }
}
```

- **Remediation:**

- Implement proper input validation and error handling to prevent application crashes.
- Set input size limits to prevent buffer overflows.
- Educate developers on secure coding practice.

- **CWE (Common Weakness Enumeration):**

- CWE-190: Integer Overflow or Wraparound

*This report provide an overview of each challenge, its impact, steps to reproduce, proof of concept, remediation recommendations, and the relevant Common Weakness Enumeration (CWE) identifiers.*