Mobile Application Security Audit Report of Public Health Web EHealth

Audit Report No: KSITM/CERT-K/VAPT/93-1123-I

Computer Emergency Response Team-Kerala

08/12/2023

Document Control

Application Name	**Mobile Application of e-Health Public Health Web**
App Name	**eHealth PHS**
Auditor Name	**Arathi,Sarania- Engineer CERT-K**
Reviewed By	**Ayswaria.R.C, Senior Security Engineer CERT-K**
Approved By	**Aswathi.S, Manager CERT-K**
Audit Completion Date	**08.12.2023**

Table of Contents

1.0	Client Details	
1.1	Name of client	eHealth
1.2	Address of client	eHealth Directorate of Health Services, General Hospital Junction, Thiruvananthapuram – 695035
2.0	Details of Application	
2.1	Product nomenclature	Mobile Application of Public Health Web eHealth & Web service
2.2	Version No:	-NA-
2.3	Date of release	-NA-
2.4	Application description	Mobile Application of Public Health Web eHealth
2.5	Developing organization & Point of contact	Self
2.6	App Name	eHealth PHS
2.7	Date of receipt for audit	10.11.2023
3.0	Audit description	
3.1	Name and Address of auditing agency	CERT-K, Kerala State IT Mission Saankethika, Vrindavan Gardens, Pottakkuzhi, Pattom.P.O, Thiruvananthapuram, Kerala 695004
3.2	Scope of work	The scope of audit is limited to above mentioned web application. This security assessment is carried out to gauge the security posture Mobile Application of **e-Health PHS.** The result of the assessment is then analyzed for vulnerabilities. The vulnerabilities are assigned a risk rating based on threat, vulnerability and impact.
3.3	Audit standard	The application is audited based on OWASP Top 10 vulnerabilities.
3.4	Audit report version	1.0
3.4	Audit start date	23.11.2023
3.5	Audit completion date	08.12.2023
3.6	Tools used	Burpsuite Professional QARK Frida Android Studio MobSF Genymotion Postman Nmap - Zenmap GUI

1. Executive Summary

This document provides the result of the Phase 1 Vulnerability Assessment performed by CERT-K against the APK **e-Health PHS** and its related APIs (temporarily hosted at SDC). The APK has been deployed temporarily at local testing environment at KSITM, and tested using automated tools followed by manual verification of discovered security vulnerabilities.

The major goal of this study is to offer the mobile application security audit report for **e-Health PHS**, which was undertaken to analyze vulnerabilities and potential threats. The assessment covered several aspects of the application's security architecture, code, and data handling methods. The review sought to discover weaknesses and potential threats related to the Mobile Application security. This report details the findings of the White Box Mobile Application Security Audit activity. The information contained within this document is considered extremely confidential and intended only for private use of e-Health, Government of Kerala.

2. Scope

The scope of the assessment was limited to performing the Mobile Application Testing on the APK mentioned below:

Details of Mobile Application	
App Name	eHealth PHS
Package Name	**in.gov.kerala.ehealth.phs**
Main Activity	**in.gov.kerala.ehealth.phs.MainActivity**
File Name	app-release_uat1.apk
Android Version	1.0.0
Android Version Code	81123

3. Summary of Vulnerabilities

The Mobile Application Security Audit carried out based on OWASP Mobile Top 10 categories-2016, utilizing their framework to assess the security posture of **e-Health PHS.** The audit methodology encompassed thorough evaluations across various dimensions to identify vulnerabilities and potential risks.

#	OWASP Top 10 Vulnerabilities	Risk Severity	Phase1
1	Improper Platform Usage	Medium	1
2	Insecure Data Storage	High/Low	2
3	Insecure Communication	High	2
4	Insecure Authentication	Medium/Low	14
5	Insufficient Cryptography	Medium	3
6	Insecure Authorization	-NA-	0
7	Client Code Quality	High	2
8	Code Tampering	-NA-	0
9	Reverse Engineering	-NA-	0
10	Extraneous Functionality	-NA-	0

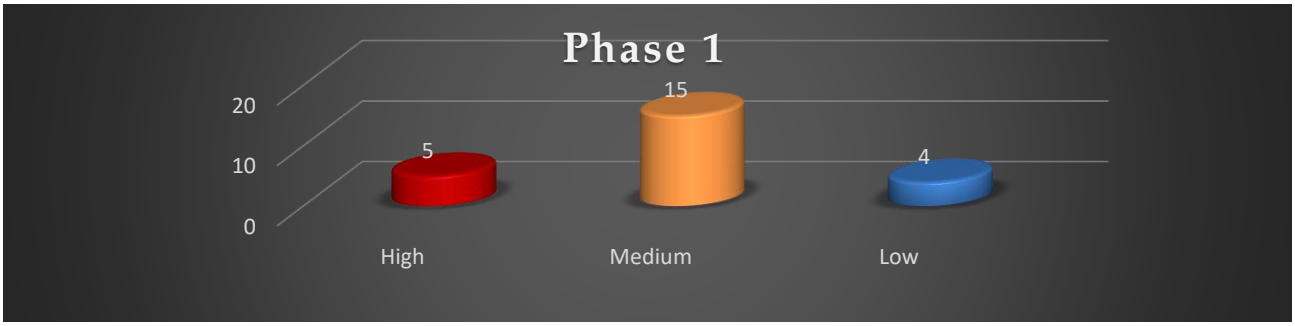
3.1 Vulnerability Severity Definition

High: *-A vulnerability, which, if exploited would allow malicious native-code to execute, potentially without a user being aware.*

Medium: - A vulnerability that is limited to a significant degree by factors such as default configuration, auditing, or is difficult to exploit.

Low: - A vulnerability that has minimal impact and is extremely difficult to exploit

The graphical representations of the vulnerabilities are detailed as below.



3.2 Methodology

3.2.1 Scoping and Planning

- Define the scope of the assessment, including the mobile platforms (iOS, Android, etc.), specific app versions, and functionalities to be tested.
- Identify the goals, objectives, and constraints of the assessment.
- Plan the resources, tools, and timelines required for the assessment.

3.2.2 Exploitation

Utilizing the information gathered in planning, we try to find any vulnerability in the software and services identified and try to exploit it.

3.2.3 Reporting and Recommendations

- Document findings, vulnerabilities, and their potential impact on the application's security.
- Provide recommendations and actionable steps to mitigate identified risks.
- Prioritize recommendations based on severity and potential impact

4. Detailed Vulnerability Description

4.1 Improper Platform Usage

4.1.1 Application Permission		
Affected File	**EHealth PHS (AndroidManifest.xml)**	
Severity	**Medium**	
CVSS	Base	4.5
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-284: Improper Access Control CWE-285: Improper Authorization	

Description

- android.permission.ACCESS_FINE_LOCATION: fine (GPS) location: Access fine location sources, such as the Global Positioning System on the phone, where available. Malicious applications can use this to determine where you are and may consume additional battery power.
- android.permission.CAMERA: Allows application to take pictures and videos with the camera. This allows the application to collect images that the camera is seeing at any time.
- android.permission.READ_EXTERNAL_STORAGE: Allows an application to read from external storage.
- android.permission.WRITE_EXTERNAL_STORAGE: Allows an application to write to external storage.

Impact

Apps with excessive or unnecessary permissions can pose security risks, potentially allowing unauthorized access to data or system resources.

Recommendation

Request only the permissions necessary for the app's functionality. Avoid asking for excessive permissions that aren't directly related to the app's core features.

Proof of Concept (POC)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="81123" android:versionName="1.0.0" android:compileSdkVersion="33" android:compileSdkVersionCodename="13" package="in.go
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="20" android:targetSdkVersion="30" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <permission android:name="in.gov.kerala.ehealth.phs.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION" android:protectionLevel="signature" />
    <uses-permission android:name="in.gov.kerala.ehealth.phs.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" android:required="false" />
    <uses-feature android:name="android.hardware.camera.front" android:required="false" />
    <uses-feature android:name="android.hardware.camera.autofocus" android:required="false" />
    <uses-feature android:name="android.hardware.camera.flash" android:required="false" />
    <uses-feature android:name="android.hardware.screen.landscape" android:required="false" />
    <uses-feature android:name="android.hardware.wifi" android:required="false" />
    <application android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:debuggable="true" android:allowBackup="false" android:spl
```

Reference

https://cwe.mitre.org/data/definitions/285.html

4.2 Insecure Data Storage

4.2.1 App can read/write to External Storage

Affected File	**d/d/d/a.java** **d/d/d/b.java** **io/flutter/plugins/c/g.java** **io/flutter/plugins/c/i.java**	
Severity	**High**	
CVSS	Base	7.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-276: Incorrect Default Permissions	

Description

When an app has the ability to read from and write to external storage on an Android device, it means the app can access files, documents, media, and other data stored on the device's external storage.

Impact

Access to external storage can potentially expose sensitive user data to other apps or malicious entities. If the app isn't properly secured, other apps or attackers might access this data without permission.

Recommendation

Ensure that the app requests and utilizes the appropriate permissions (READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE) only when absolutely necessary. Target the specific files or directories required, rather than requesting broad access to the entire storage.

Proof of Concept (POC)

Instance 1

```
116.      }
117.
118.      public static File[] e(Context context) {
119.          return Build.VERSION.SDK_INT >= 19 ? b.a(context) : new File[]{context.getExternalCacheDir()};
120.      }
121.
122.      public static File[] f(Context context, String str) {
123.          return Build.VERSION.SDK_INT >= 19 ? b.b(context, str) : new File[]{context.getExternalFilesDir(str)};
124.      }
125.
126.      public static Executor g(Context context) {
127.          return Build.VERSION.SDK_INT >= 28 ? e.a(context) : d.d.h.d.a(new Handler(context.getMainLooper()));
128.      }
```

Reference

https://www.digitalocean.com/community/tutorials/android-external-storage-read-write-save-file

4.2.2 App creates temp file. Sensitive information should never be written into a temp file.

Affected File	**com/journeyapps/barcodescanner/j.java** **d/h/a/a.java**	
Severity	**Low**	
CVSS	Base	2.5

	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)
CVE	CWE-276: Incorrect Default Permissions

Description

Apps may create temporary files to store data temporarily, often for caching, processing, or other temporary purposes during runtime.

Impact

Storing sensitive information in temporary files exposes it to potential unauthorized access or retrieval by other apps or malicious entities with access to the file system.

Recommendation

- Avoid Storing Sensitive Data: Design the app to avoid storing sensitive information in temporary files whenever possible. Use in-memory storage or encrypted storage mechanisms instead.
- Secure File Handling: If temporary files are necessary, ensure secure file handling practices. Encrypt the file content or use appropriate access controls and file permissions to restrict access.

Proof of Concept (POC)

```
private String l(h hVar) {
    if (this.f1147d) {
        Bitmap b2 = hVar.b();
        try {
            File createTempFile = File.createTempFile("barcodeimage", ".jpg", this.a.getCacheDir());
            FileOutputStream fileOutputStream = new FileOutputStream(createTempFile);
            b2.compress(Bitmap.CompressFormat.JPEG, 100, fileOutputStream);
            fileOutputStream.close();
            return createTempFile.getAbsolutePath();
        } catch (IOException e2) {
            String str = o;
            Log.w(str, "Unable to create temporary file and store bitmap! " + e2);
        }
```

Reference

https://cwe.mitre.org/data/definitions/276.html

4.3 Insecure Communication

4.3.1 App can be installed on a vulnerable Android version

App Name	**E-Health PHS**	
Severity	**High**	
CVSS	Base	7.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-326: Inadequate Encryption Strength CAPEC-94: Man in the Middle Attack CAPEC-157: Sniffing Attacks	

Description

An app can be installed on a vulnerable Android version, it means that the app can run on an Android operating system that contains known security flaws or weaknesses. Vulnerabilities in the Android

system can be exploited by malicious entities to compromise the device's security or gain unauthorized access to sensitive information.

Impact

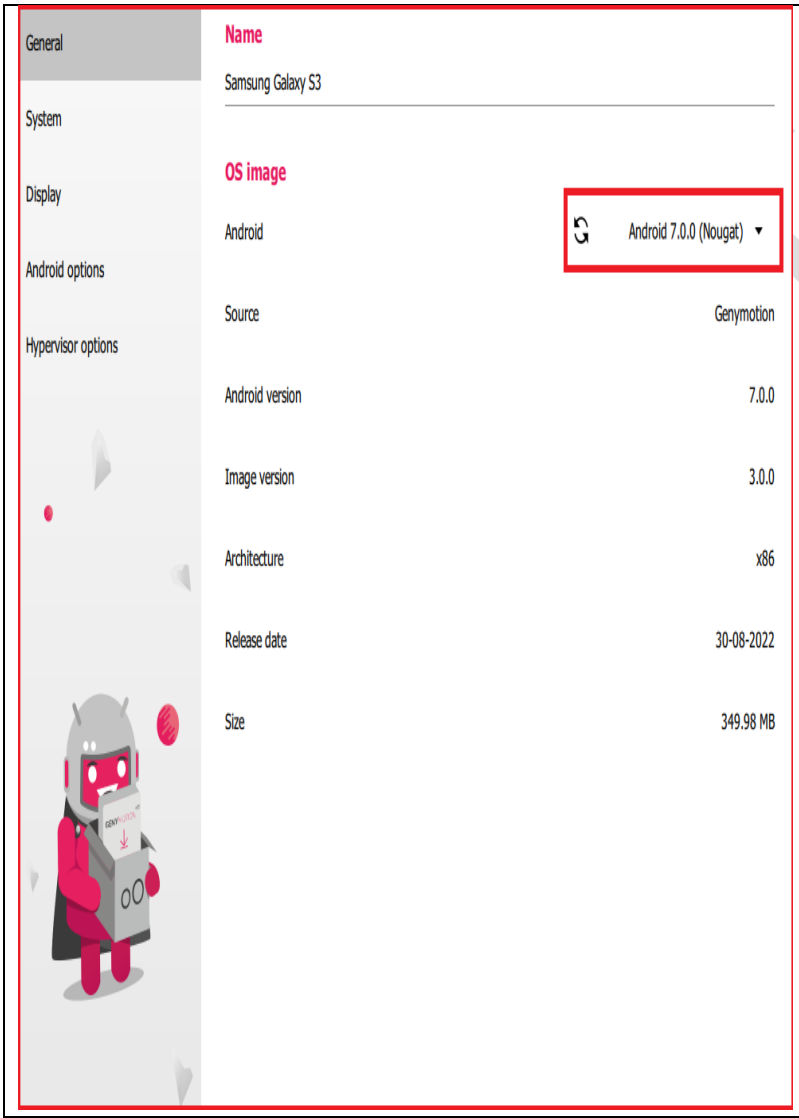
- **Security Breach**: Vulnerabilities might allow malicious actors to gain unauthorized access to sensitive information stored on your device.
- **Malware Installation**: Attackers can exploit vulnerabilities to install malware or malicious applications, compromising your device's security and privacy.
- **Data Theft**: Vulnerable Android versions can potentially expose personal data, including contacts, messages, and login credentials, to unauthorized parties.
- **Device Compromise**: In severe cases, vulnerabilities can lead to complete compromise of the device, allowing attackers to take control.

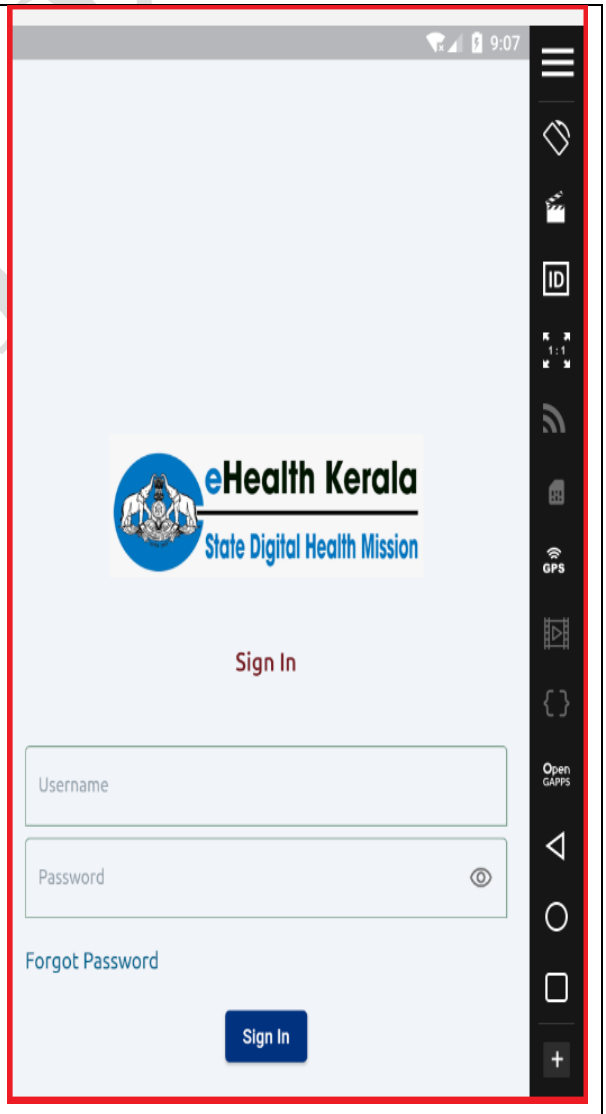
Recommendation

Use an Android version higher than 8 & API 26 to receive reasonable security updates.

Proof of Concept (POC)

Step 1: App can be Installed in Vulnerable Android version





Reference

https://source.android.com/docs/security/bulletin/2022-12-01

https://developer.android.com/about/versions/oreo/android-8.0-migration

4.3.2 Clear Text Traffic is enabled for App		
App Name	**E-Health PHS**	
Severity	**High**	
CVSS	Base	7.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-319: Clear text Transmission of Sensitive Information	

Description

When an app allows clear text traffic, it means that the app sends and receives data without encrypting it. This data transmission occurs as plain text, which can be intercepted and viewed by anyone with access to the network traffic.

Impact

Clear text communication increases the vulnerability of the app to man-in-the-middle attacks, where an attacker intercepts and potentially alters the data being transmitted between the app and its server.

Recommendation

Set the android:usesCleartextTraffic flag to false in the Android Manifest. This will result in the app asking the platform and third-party libraries to prevent the app from using clear text traffic.

Proof of Concept (POC)

Reference

https://www.geeksforgeeks.org/android-cleartext-http-traffic-not-permitted/

https://trendoceans.com/how-to-fix-cleartext-http-traffic-not-permitted-in-android/

4.4 Insecure Authentication

4.4.1 Application vulnerable to Janus Vulnerability		
App Name	**E-Health PHS**	
Severity	**Medium**	
CVSS	Base	7.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CVE-2017–13156- Exploiting Apps vulnerable to Janus	

Description

The Janus vulnerability is a security flaw discovered in Android applications that can allow attackers to modify the APK (Android application package) file without affecting the app's cryptographic signature. Application is signed with v1 signature scheme, making it vulnerable to Janus vulnerability on Android 5.0-8.0, if signed only with v1 signature scheme. Applications running on Android 5.0-7.0 signed with v1, and v2/v3 scheme is also vulnerable.

Impact

Janus allows attackers to modify legitimate apps, leading to potential security breaches, data theft, or unauthorized access to sensitive information.

Recommendation

Keep apps and the Android operating system updated with the latest security patches. Many vulnerabilities, including Janus, are addressed through updates.

Proof of Concept (POC)

```
Binary is signed
v1 signature: True
v2 signature: True
v3 signature: False
v4 signature: False
X.509 Subject: C=INDIA, ST=KERALA, L=TVM, O=EHEALTH, OU=EHEALTH, CN=PHS
Signature Algorithm: rsassa_pkcs1v15
Valid From: 2023-09-19 08:34:41+00:00
Valid To: 2051-02-04 08:34:41+00:00
Issuer: C=INDIA, ST=KERALA, L=TVM, O=EHEALTH, OU=EHEALTH, CN=PHS
Serial Number: 0x2992cd0b819760ba
Hash Algorithm: sha256
md5: f4858f0880222eb9590ecc8410bd57eb
sha1: 98e6654ff825855e92f0585b5aa9cbcb79e65b29
sha256: 03260415043c8cb39c4a89418011f27c7ecdcd3c3bee1c7ebf9b34c3ad1d89d2
sha512: f770cca98cfc573d393f7cedba655312b7bf457b8fc382026670241e5dcf22ddc27d3cc2b9b8bbfadeba42219e3aecff80c6bf0f087322e850a92df4706fd218
PublicKey Algorithm: rsa
Bit Size: 2048
Fingerprint: ae23491f57c0fecb2c2ca1642828840bd4431fe7d58b9a6fc3606ce8b080831d
Found 1 unique certificates
```

Reference

https://medium.com/mobis3c/exploiting-apps-vulnerable-to-janus-cve-2017-13156-8d52c983b4e0

4.4.2 The App logs information. Sensitive information should never be logged.

Affected File	**com/baseflow/geolocator/GeolocatorLocationService.java** **com/baseflow/geolocator/m.java** **com/baseflow/geolocator/n.java** **com/baseflow/geolocator/o.java** **com/baseflow/geolocator/q/k.java** **com/baseflow/geolocator/r/b.java** **com/journeyapps/barcodescanner/i.java** **com/journeyapps/barcodescanner/j.java** **com/journeyapps/barcodescanner/n.java** **com/journeyapps/barcodescanner/x/e.java** **com/journeyapps/barcodescanner/x/f.java** **com/journeyapps/barcodescanner/x/g.java** **com/journeyapps/barcodescanner/x/h.java**

	com/journeyapps/barcodescanner/x/l.java com/journeyapps/barcodescanner/x/n.java com/journeyapps/barcodescanner/x/q.java d/a/n/g.java d/d/d/e/e.java d/d/d/e/f.java d/d/d/e/j.java d/d/e/d.java d/d/e/f.java d/d/e/g.java d/d/e/h.java d/d/e/k.java d/d/e/l.java d/d/h/i.java d/d/l/a0.java d/d/l/b0.java d/d/l/d0.java d/d/l/e.java d/d/l/i0.java d/d/l/j.java d/d/l/k0/b.java d/d/l/l0/c.java d/d/l/m.java d/h/a/a.java d/h/a/b.java d/k/a/b.java d/l/a.java d/m/a/a/h.java e/a/a/j.java e/a/a/n.java e/a/a/o.java e/a/a/p.java e/d/a/a/d/b/a.java e/d/b/s/a/f.java e/d/b/s/a/h.java e/d/b/s/a/p/a/a.java e/e/a/a0.java e/e/a/c0.java e/e/a/r.java

	g/a/b.java **io/flutter/plugins/a/a.java** **io/flutter/plugins/c/i.java** **io/flutter/plugins/imagepicker/b.java** **io/flutter/plugins/imagepicker/g.java** **io/flutter/plugins/urllauncher/a.java** **io/flutter/plugins/urllauncher/b.java** **io/flutter/plugins/urllauncher/c.java**	
Severity	**Medium**	
CVSS	Base	4.6
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-532: Insertion of Sensitive Information into Log File	

Description

When an app logs sensitive information, it means that it records potentially confidential or private data in its logs.

Impact

Logging sensitive information exposes it to potential unauthorized access. If these logs are not properly secured or managed, they could be accessed by malicious actors or unauthorized users.

Recommendation

Develop logging mechanisms that exclude sensitive information by design. Implement filters or checks to ensure that sensitive data is not included in logs.

Proof of Concept (POC)

```
88.
89.     public boolean a(boolean z) {
90.         return z ? this.f821h == 1 : this.f820g == 0;
91.     }
92.
93.     public void b(com.baseflow.geolocator.q.j jVar) {
94.         com.baseflow.geolocator.q.h hVar = this.n;
95.         if (hVar != null) {
96.             hVar.f(jVar, this.f819f);
97.             i(jVar);
98.         }
99.     }
100.
101.    public void c() {
102.        if (this.f819f) {
103.            Log.d("FlutterGeolocator", "Stop service in foreground.");
104.            if (Build.VERSION.SDK_INT >= 24) {
105.                stopForeground(75415);
106.            } else {
107.                stopForeground(true);
108.            }
109.            j();
110.            this.f819f = false;
111.            this.n = null;
112.        }
113.    }
114.
115.    public void d(com.baseflow.geolocator.q.j jVar) {
116.        if (this.n != null) {
117.            Log.d("FlutterGeolocator", "Service already in foreground mode.");
118.            b(jVar);
119.        } else {
120.            Log.d("FlutterGeolocator", "Start service in foreground mode.");
121.            com.baseflow.geolocator.q.h hVar = new com.baseflow.geolocator.q.h(getApplicationContext(), "geolocator_channel_01", 75415, jVar);
122.            this.n = hVar;
```

Reference

https://neotechland.blogspot.com/2018/09/the-app-logs-information-sensitive.html

4.4.3 No Session Time Out: Ability to login with two devices		
App Name	**E-Health PHS**	
Severity	**Medium**	
CVSS	Base	4.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-613: Insufficient Session Expiration CWE-384: Session Fixation	

Description

If a user logs in to the program, their session remains active until they choose to log out. The application does not offer a session time out feature. Several devices can run the same application for a single user concurrently.

Impact

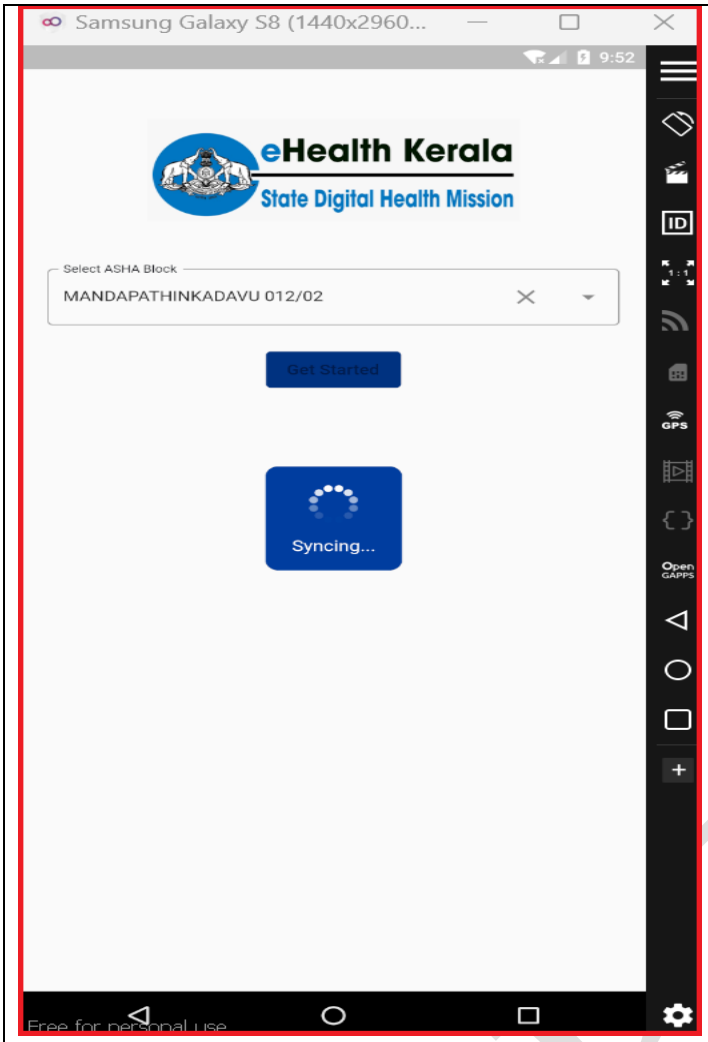
If a device is left unattended or accessed by someone else, the open session can be exploited to access sensitive information or perform actions without proper authentication.

Recommendation

- Define and implement a session timeout policy based on security best practices. Users should be automatically logged out after a defined period of inactivity.
- Disable the user interaction with multiple devices at the same time.

Proof of Concept (POC)

Instance 1: The user logged-In with in Emulator



Instance 2: The user logged-In with in Another Android device

Reference

https://developer.android.com/guide/topics/connectivity/cross-device-sdk/sessions

4.4.4 Received Multiple OTPs in a Minute

Affected URL	**https://hds.kerala.gov.in/phs_api/api/vr1/generateOTP**	
Severity	**Medium**	
CVSS	Base	4.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-307: Improper Restriction of Excessive Authentication Attempts CWE-770: Allocation of Resources Without Limits or Throttling	

Description

Receiving multiple OTPs (One-Time Passwords) in a minute can indicate several potential scenarios, including system errors, user-initiated requests, or potentially malicious activities.

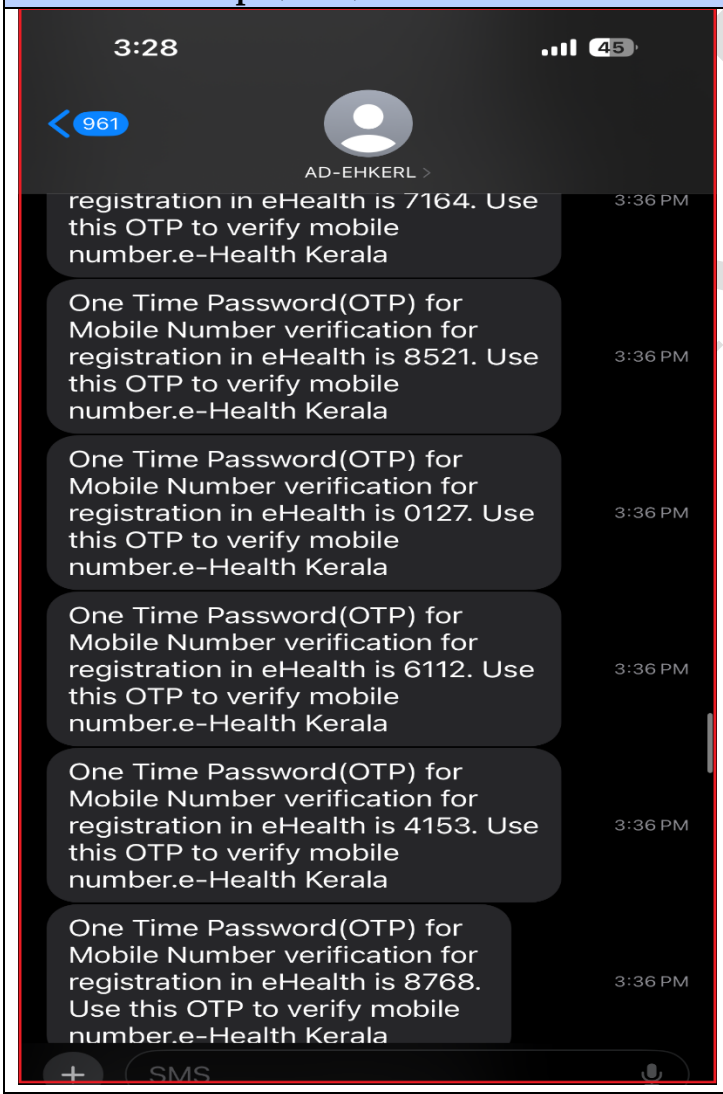
Impact

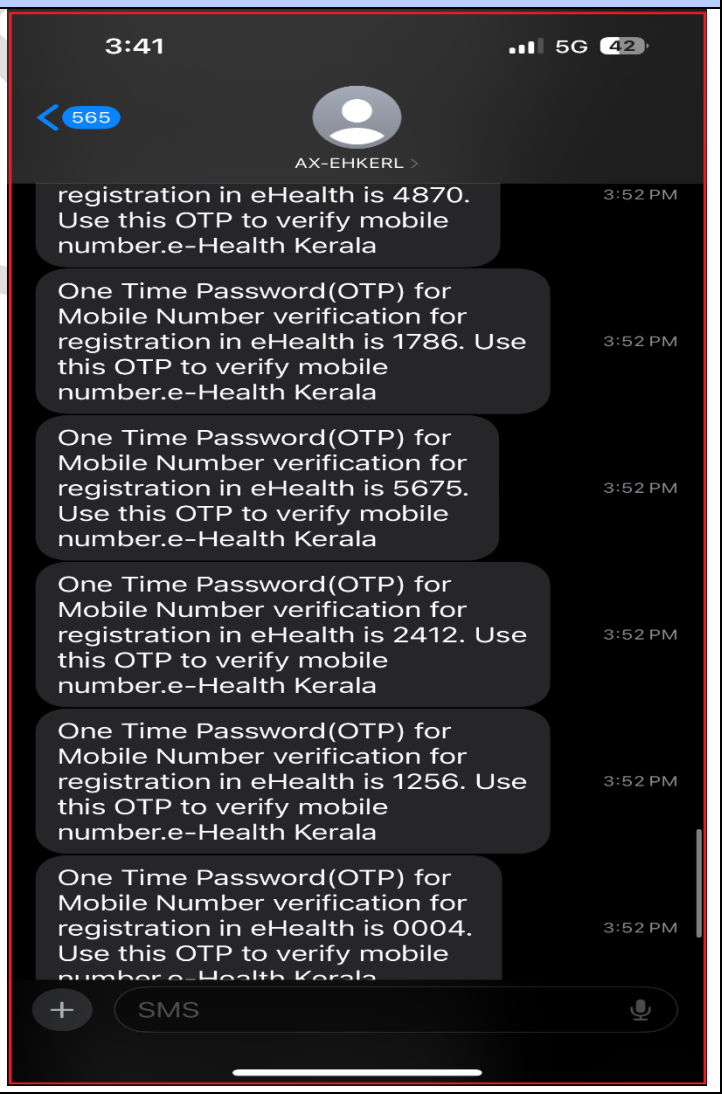
It could be a result of delays or errors within the OTP delivery system, causing multiple OTPs to be sent for the same request.

Recommendation

Implement robust error handling mechanisms in the OTP generation and delivery system to prevent duplicate OTPs being sent for a single request.

Proof of Concept (POC)





Reference

https://cwe.mitre.org/data/definitions/799.html

4.4.5 Input Returned in Response

Affected URL	**https://hds.kerala.gov.in/phs_api/api/vr1/getHospitalTypeId**	
Severity	**Medium**	
CVSS	Base	4.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	

Description

When an application returns input directly in its response, it means that user-provided input is reflected back without proper validation or sanitization

Impact

User-supplied input reflected in responses without proper sanitation can expose sensitive information or inadvertently leak internal application details.

Recommendation

Implement strict input validation to ensure that user-provided data is sanitized and validated before being reflected back in responses. Use input encoding to prevent script injections.

Proof of Concept (POC)

Request

Pretty Raw Hex \n

```
POST /phs_api/apiwaykojw29b/vr1/getHospitalTypeId HTTP/2
Host: hds.kerala.gov.in
Authorization: Basic
ZWhlYWx0aHBoc2FkbWlu0mVoZWFsdGhwaHNAMjAyMw==
User-Agent: PostmanRuntime/7.29.0
Accept: */*
Postman-Token: ccdb78d4-f5b3-4d13-8824-2a15826ee583
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data;
boundary=--------------------------476650614613099246229
905
Cookie: Path=/
Content-Length: 209

----------------------------476650614613099246229905
Content-Disposition: form-data; name="data"

{"user_id":29371,"hospital_id":"(1974,8914,8912)"}
----------------------------476650614613099246229905--
```

Response

Pretty Raw Hex Render

```
HTTP/2 404 Not Found
Server: nginx
Date: Tue, 28 Nov 2023 10:01:46 GMT
Content-Type: application/json
Vary: Accept-Encoding
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
X-Content-Type-Options: nosniff
X-Xss-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Referrer-Policy: strict-origin-when-cross-origin

{
  "timestamp":"2023-11-28T10:01:45.475+00:00",
  "status":404,
  "error":"Not Found",
  "path":"/phs_api/apiwaykojw29b/vr1/getHospitalTypeId"
}
```

Reference

https://cwe.mitre.org/data/definitions/79.html

4.4.6 Insecure Direct Object Reference		
Affected URL	https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserData	
Severity	High	
CVSS	Base	7.6
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-472: External Control of Assumed-Immutable Web Parameter	

Description

Insecure direct object references (IDOR) are a type of access control vulnerability that arises when an application uses user-supplied input to access objects directly.

Impact

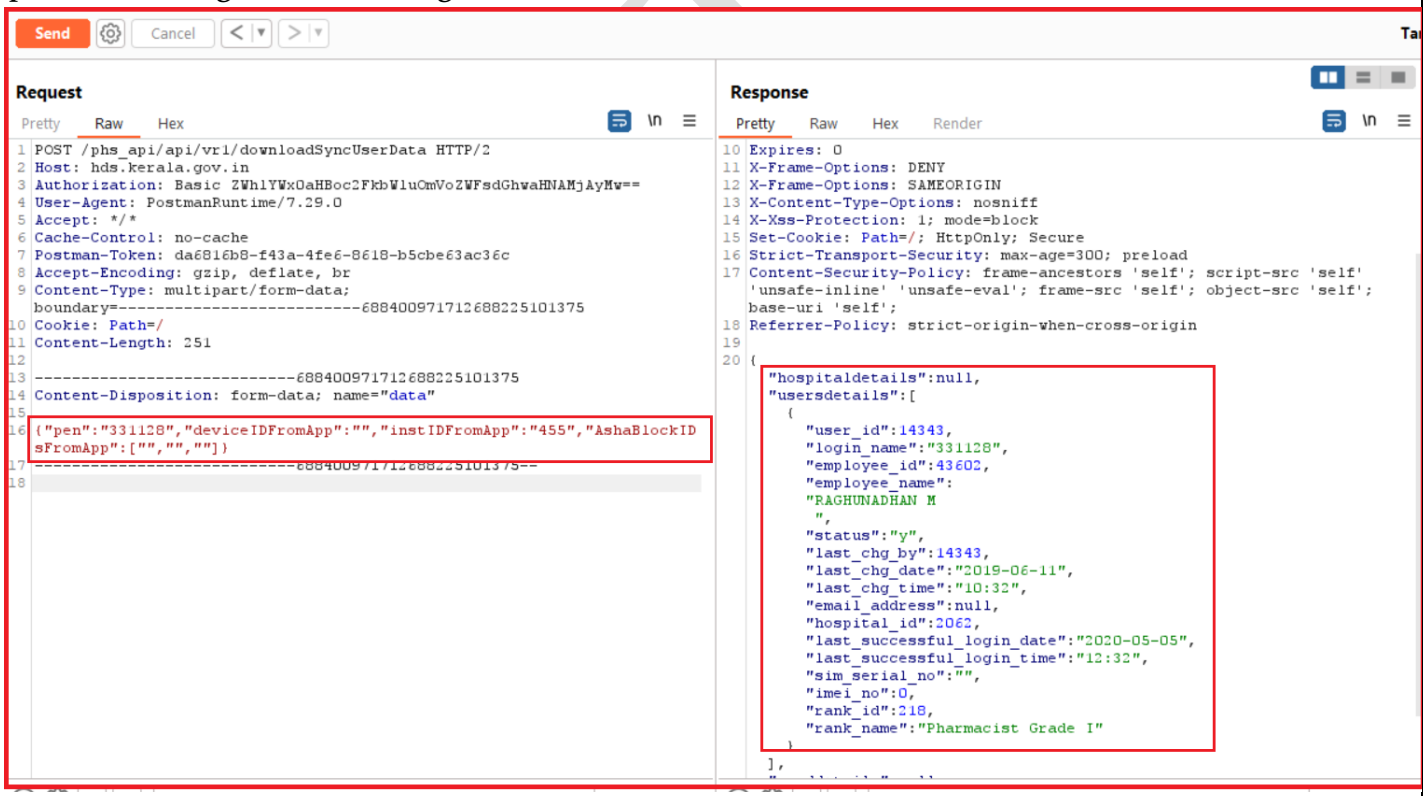
Attackers may access sensitive data belonging to other users or gain unauthorized access to restricted information.

Recommendation

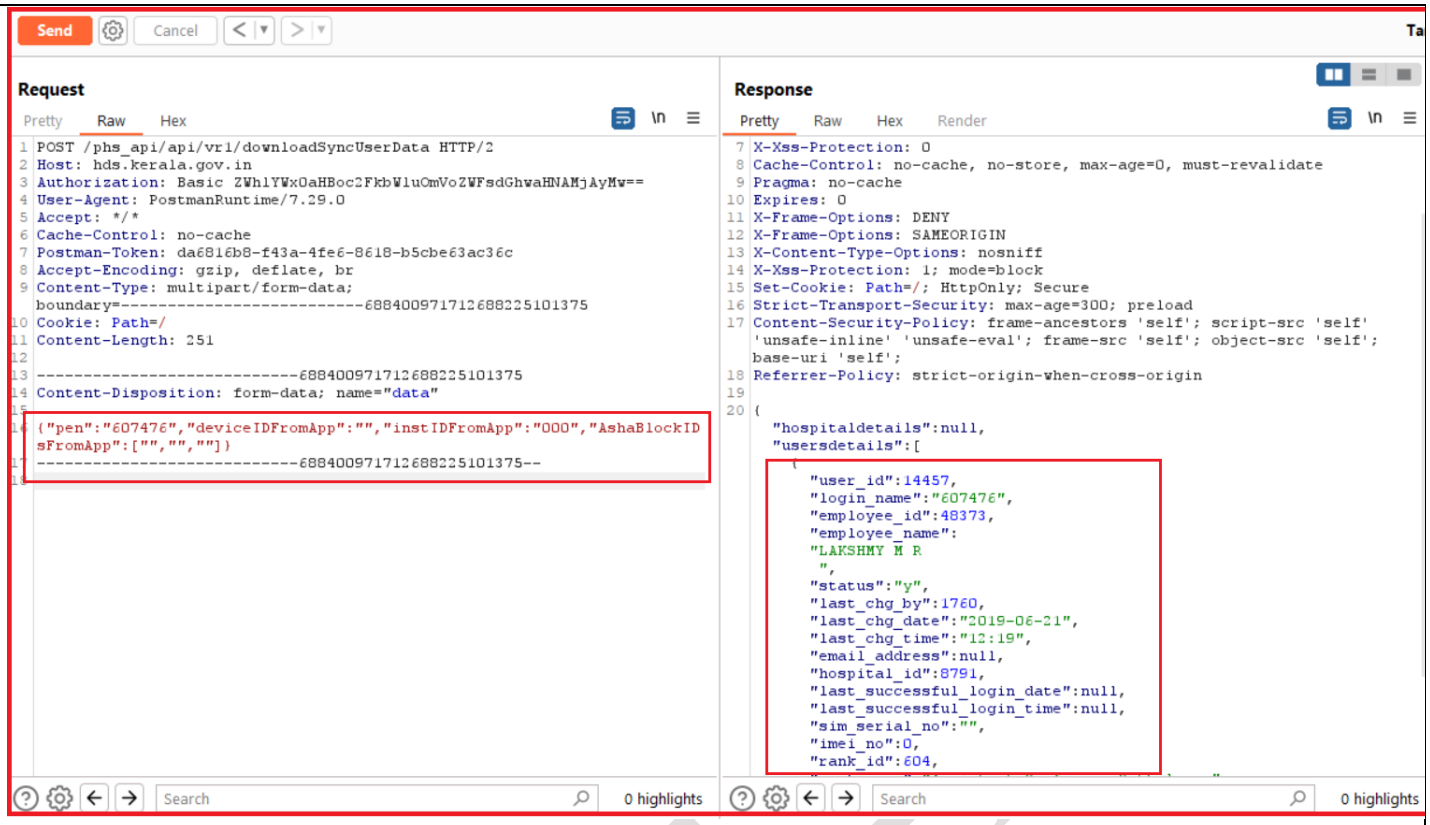
- Ensure that proper access controls and authorization mechanisms are in place to restrict users' access to only the data and functionalities they are authorized to use.
- Avoid using direct references (such as file paths, database keys) in URLs or user input. Instead, use indirect references that are mapped to the corresponding objects on the server side.
- Validate and sanitize all user-supplied input to prevent malicious manipulation. Input validation should be performed on the server side.

Proof of Concept (POC)

Step 1: After calling the URL, Change the value of Pen Number from 331128 to 607476)

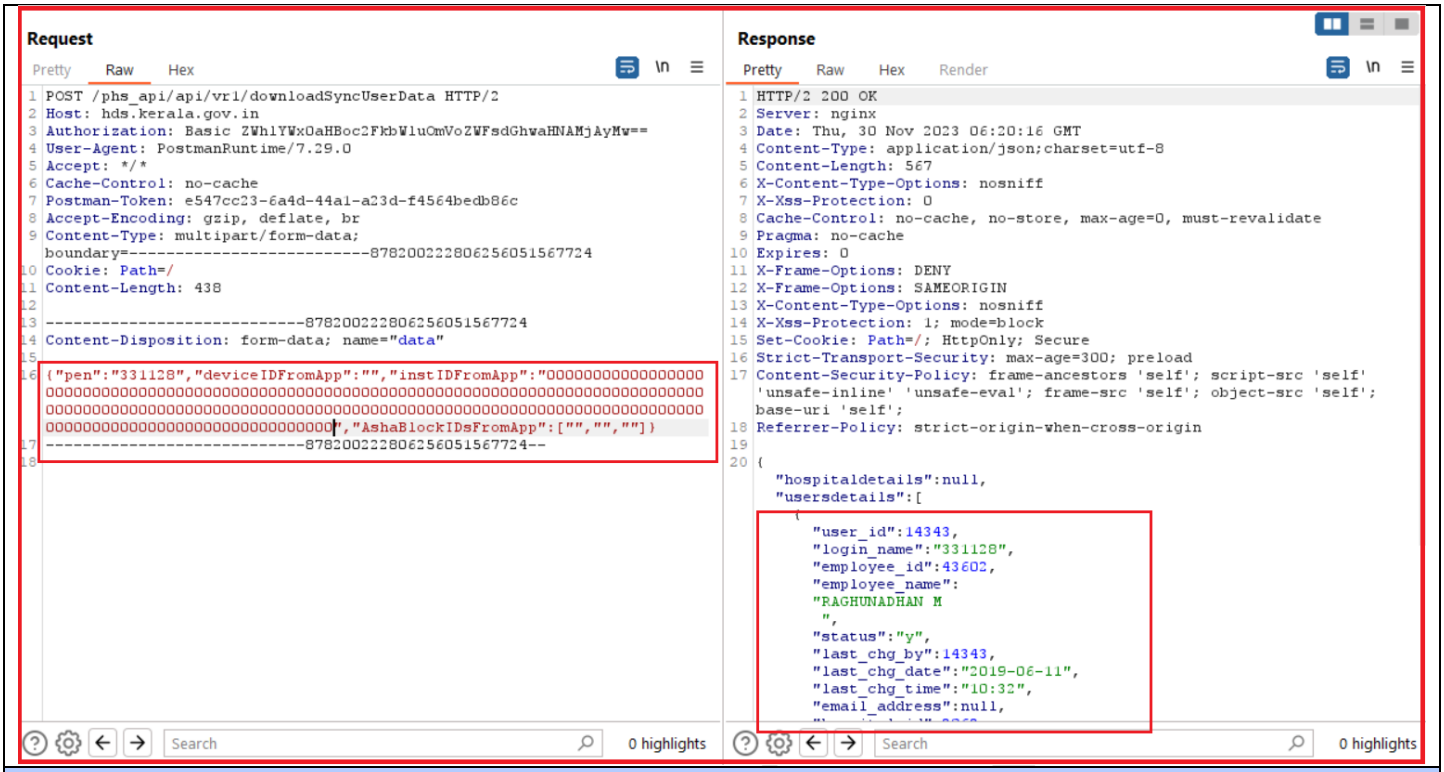


Step 2: The server accepts the request and get the response as 200 ok



Reference
https://portswigger.net/web-security/access-control/idor

4.4.7 Buffer Overflow		
Affected URL	**https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserData**	
Severity	**Medium**	
CVSS	Base	4.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-121: Stack-based Buffer Overflow	
Description		
A buffer overflow is a type of software vulnerability that occurs when a program or process attempts to store more data in a buffer (a temporary storage area in memory) than it can hold. This extra data can overflow into adjacent memory locations, corrupting or overwriting the contents of other buffers or critical data structures.		
Impact		
Buffer overflows are commonly exploited by attackers to inject malicious code into a vulnerable program's memory. This can lead to the execution of arbitrary commands, allowing attackers to gain unauthorized access to systems, compromise data, or install malware.		
Recommendation		
To mitigate the impact of buffer overflows should follow secure coding practices, employ robust input validation techniques		
Proof of Concept (POC)		



Reference
https://cwe.mitre.org/data/definitions/121.html

4.4.8 Email ID Disclosure

Affected URL	**https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserUpdateData**	
Severity	**Medium**	
CVSS	Base	2.6
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-200: Information Exposure	

Description

Email ID disclosure refers to instances where an application, website, or system inadvertently exposes users' email addresses, either through design flaws, improper configurations, or vulnerabilities.

Impact

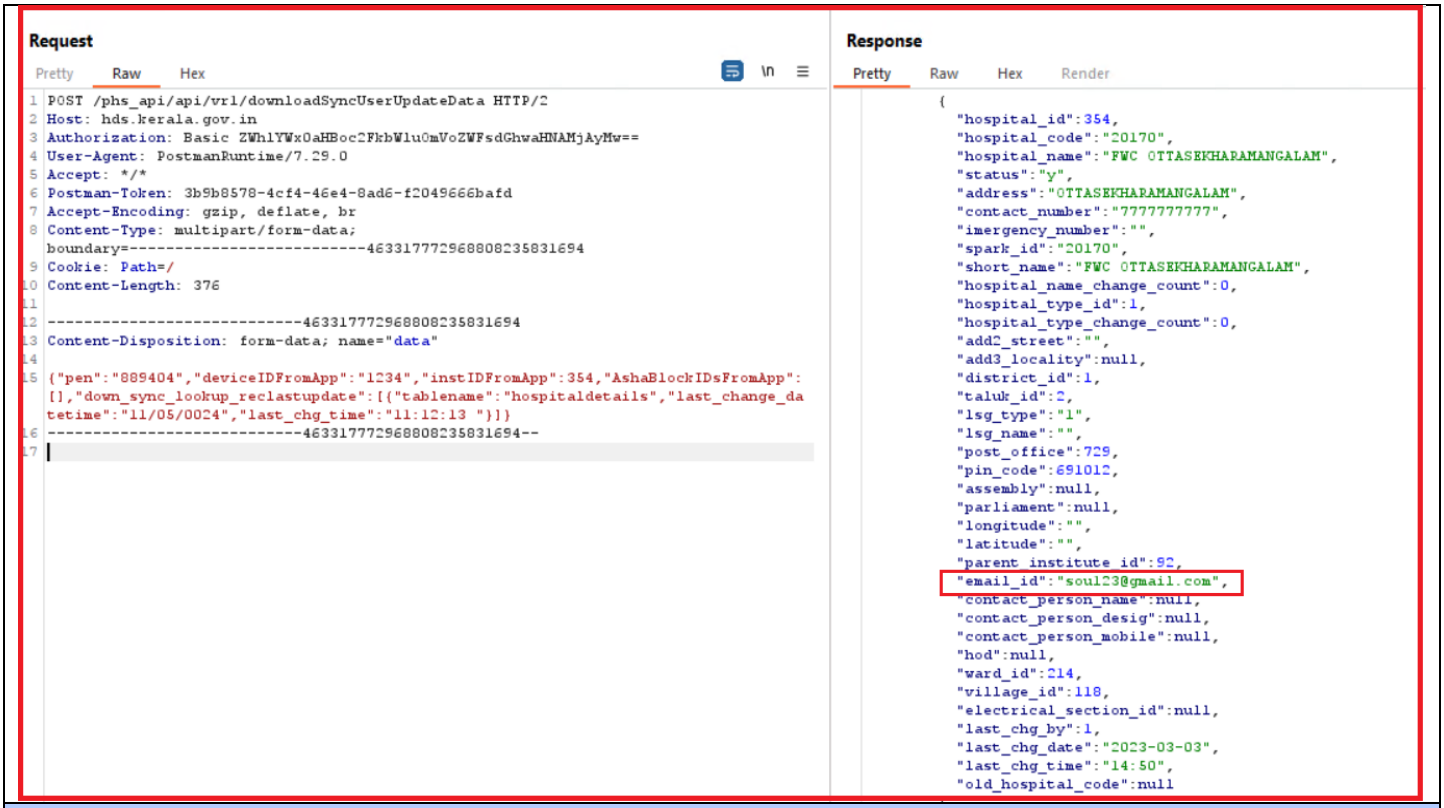
Exposing email addresses can compromise user privacy, potentially leading to unsolicited emails, phishing attempts, or targeted attacks.

Recommendation

- Data Minimization: Only collect and store email addresses when necessary. Avoid displaying or sharing email IDs publicly unless it's essential for the application's functionality.
- Encryption and Hashing: Encrypt or hash email addresses when stored in databases or transmitted across networks to prevent unauthorized access in case of a breach.

Proof of Concept (POC)

Instance 1: The following email address was disclosed in the response:
Soul23@gmail.com



Reference

https://cwe.mitre.org/data/definitions/200.html

https://capec.mitre.org/data/definitions/37.html

4.4.9 Cross Origin Resource Sharing

Affected URL	• **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncMasData** • **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncMasUpdateData** • **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserData** • **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserUpdateData** • **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncAshabiData** • **https://hds.kerala.gov.in/phs_api/api/vr1/uploadSyncPHData** • **https://hds.kerala.gov.in/phs_api/api/vr1/generateOTP** • **https://hds.kerala.gov.in/phs_api/api/vr1/verifyOTP** • **https://hds.kerala.gov.in/phs_api/api/vr1/getHospitalTypeId**	
Severity	Medium	
CVSS	Base	4.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-942: Permissive Cross-domain Policy with Untrusted Domains	

Description

Cross-Origin Resource Sharing (CORS) is a security mechanism implemented in web browsers to control access to resources (e.g., fonts, images, scripts) across different origins (domains, protocols, or ports). It allows servers to specify which origins are allowed to access their resources and which HTTP methods are allowed for cross-origin requests.

Impact

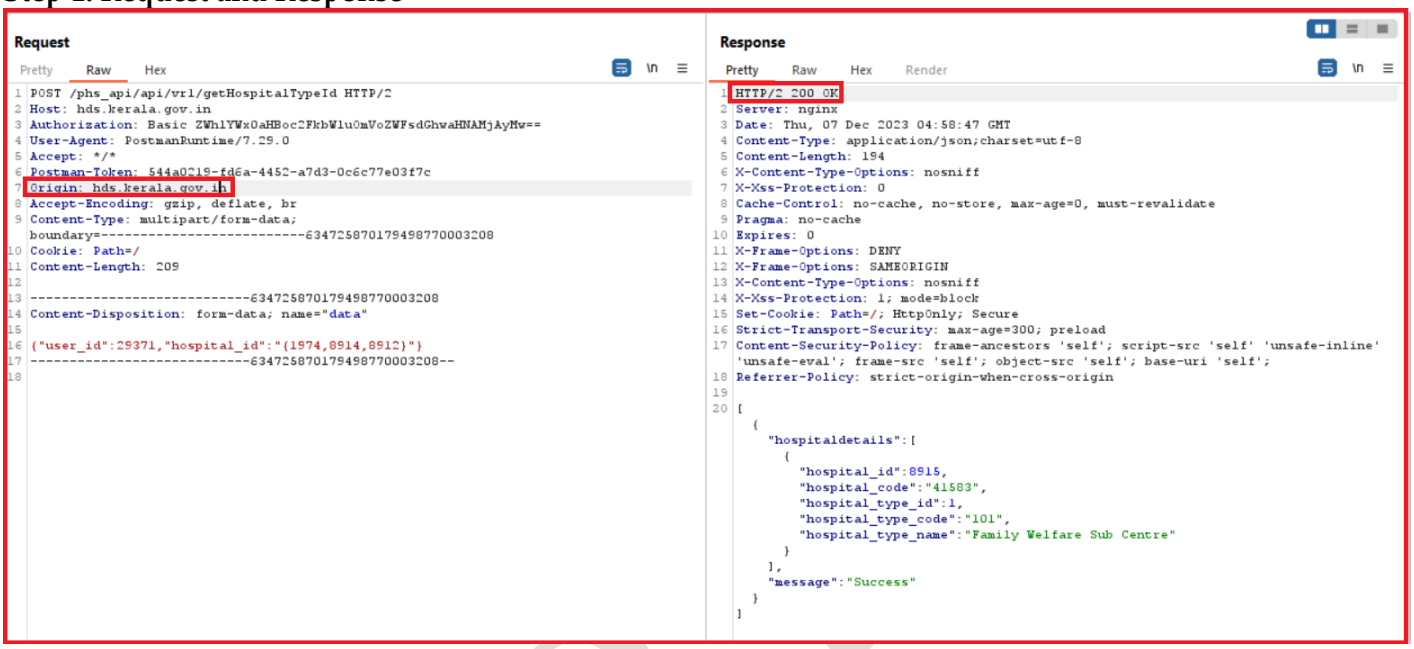
A CORS misconfiguration can leave the application at a high risk of compromises resulting in an impact on the confidentiality and integrity of data by allowing third-party sites to carry out privileged requests through your website's authenticated users such as retrieving user setting information or saved payment card data.

Recommendation

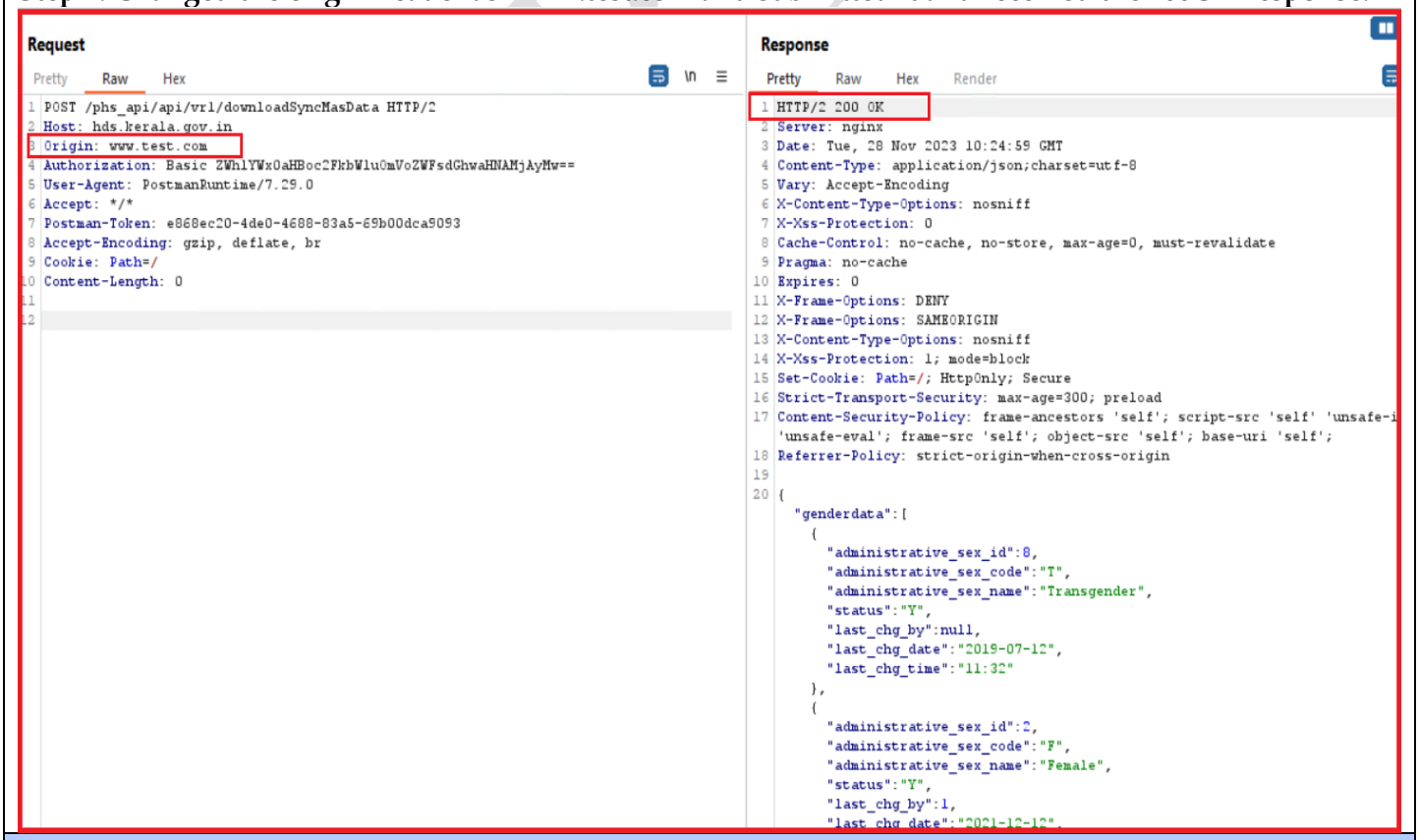
Set the response header to be ,
Access-Control-Allow-Origin: <origin>
Access-Control-Allow-Origin: null

Proof of Concept (POC)

Step 1: Request and Response



Step 2: Changed the origin header as www.test.com and submitted it and received the 200OK response.



Reference

https://portswigger.net/kb/issues/00200600_cross-origin-resource-sharing

4.4.10 Misconfigured Content Security Policy (CSP) Header		
Affected URL	**https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncMasData** **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncMasUpdateData** **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserData** **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserUpdateData** **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncAshabiData** **https://hds.kerala.gov.in/phs_api/api/vr1/uploadSyncPHData** **https://hds.kerala.gov.in/phs_api/api/vr1/generateOTP** **https://hds.kerala.gov.in/phs_api/api/vr1/verifyOTP** **https://hds.kerala.gov.in/phs_api/api/vr1/getHospitalTypeId**	
Severity	**Low**	
CVSS	Base	2.6
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-79 : Failure to Preserve Web Page Structure	

Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. Content Security Policy (CSP) can be implemented by adding a Content- Security-Policy header. The value of this header is a string containing the policy directives describing your Content Security Policy. To implement CSP, you should define lists of allowed origins for the all of the types of resources that your site utilizes. For example, if you have a simple site that needs to load scripts, stylesheets, and images hosted locally, as well as from the jQuery library from their CDN, the CSP header could look like the following:
Content-Security-Policy:
default-src 'self';
script-src 'self' https://code.jquery.com;

Impact

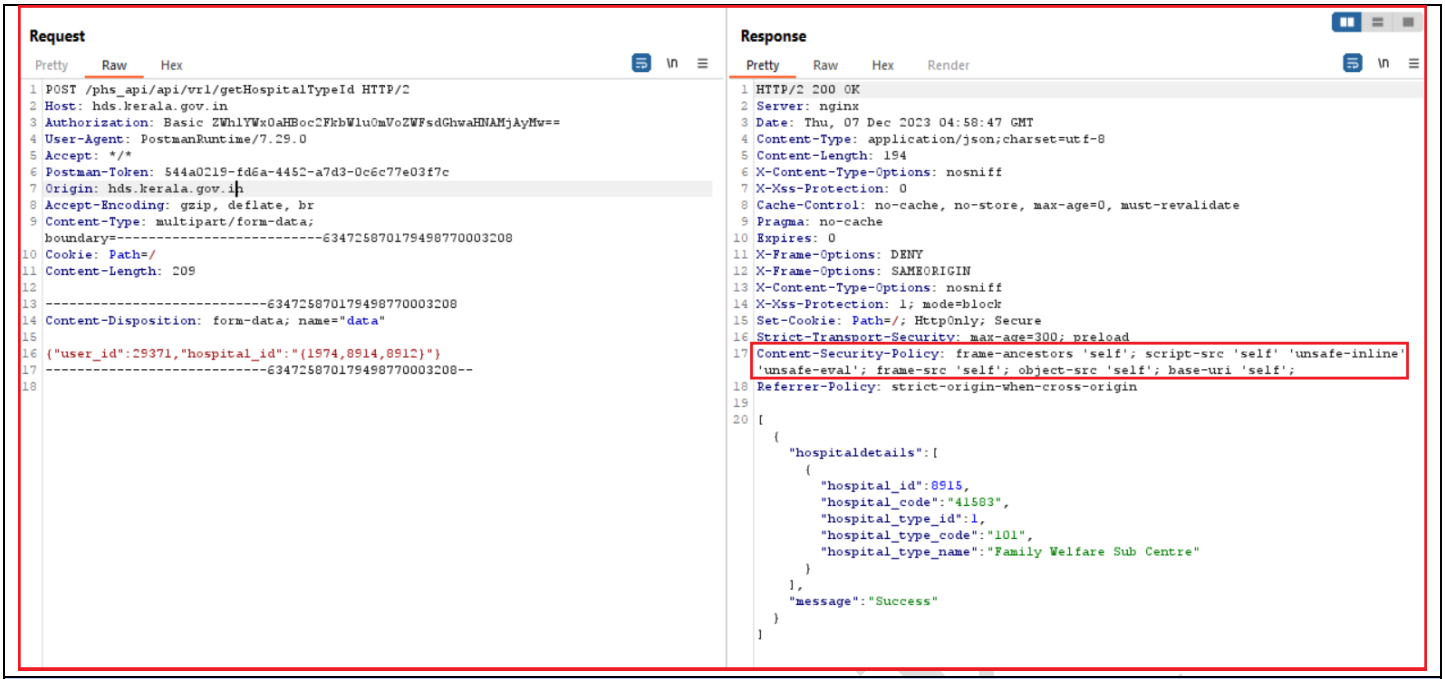
Using this vulnerability, an attacker can:-

- Use this vulnerability to perform cross-site scripting.
- Perform clickjacking on the end users.
- Perform code injection attacks.

Recommendation

It's recommended to implement Content Security Policy (CSP) into your web application. Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control resources the user agent is allowed to load for that page.

Proof of Concept (POC)



Reference
https://cwe.mitre.org/data/definitions/1004.html

4.4.11 Multiple X-Frame-Options Header		
Affected URL	**https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserUpdateData**	
Severity	**Low**	
CVSS	Base	2.6
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-693: Protection Mechanism Failure	

Description

Multiple frameworks or middleware within an application might set this header independently, causing it to appear multiple times in the response.

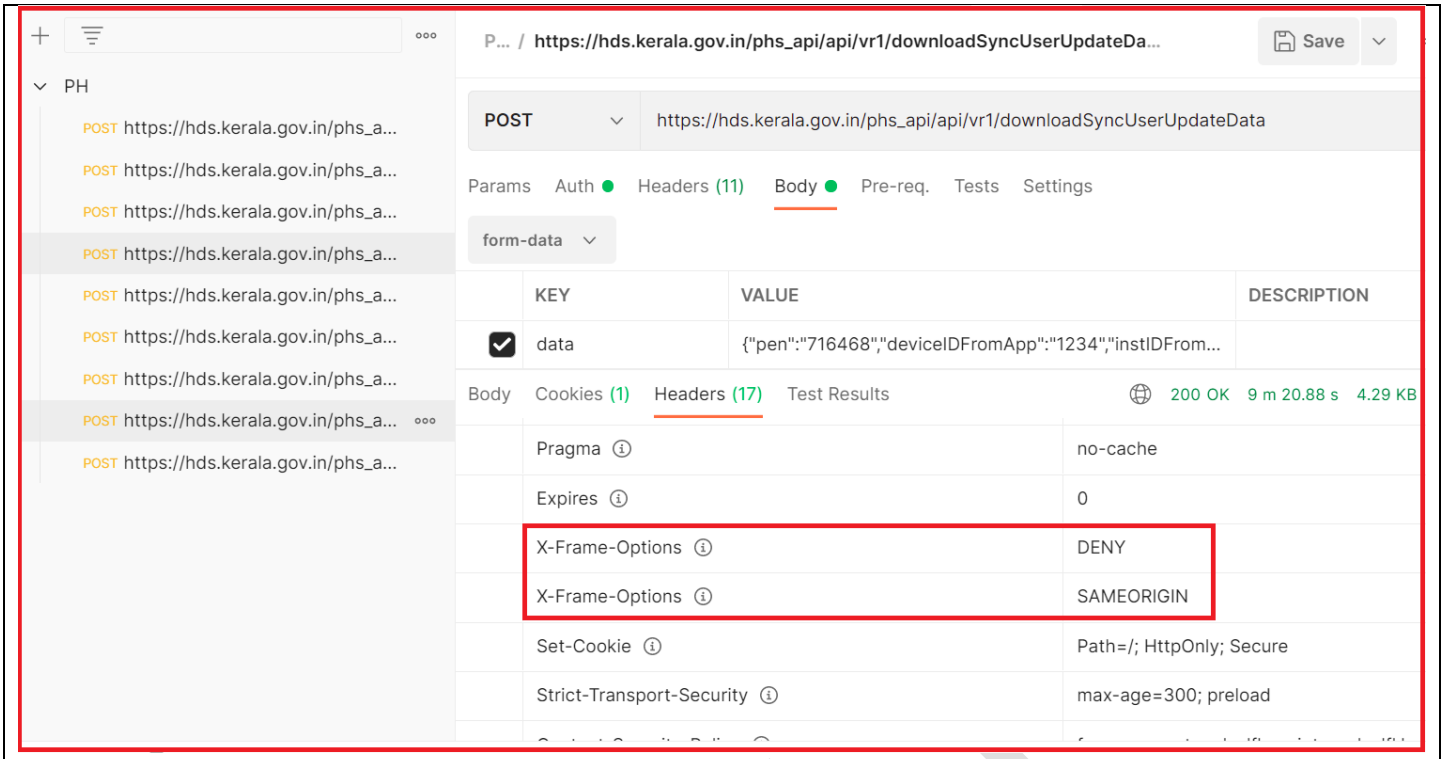
Impact

When multiple X-Frame-Options headers are present, conflicting directives or policies may arise, leading to ambiguity about which directive to follow.

Recommendation

- **Consolidate Header Settings:** Ensure that only one X-Frame-Options header is present in the response by consolidating header settings within the application or its server configurations.
- **Centralized Configuration:** Centralize the configuration for setting the X-Frame-Options header to prevent conflicting or duplicate directives from different parts of the application.
- **Header Order and Priority**: Check the order in which headers are set or delivered. Ensure that the correct header with the appropriate directive (e.g., DENY or SAMEORIGIN) takes precedence.

Proof of Concept (POC)



Reference
https://nvd.nist.gov/vuln/detail/CVE-2021-35237

4.4.12 Multiple XSS Options Header		
Affected URL	**https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserUpdateData**	
Severity	Low	
CVSS	Base	2.6
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-693: Protection Mechanism Failure	

Description

Different components or frameworks within an application might independently set this header, causing it to appear multiple times in the response.

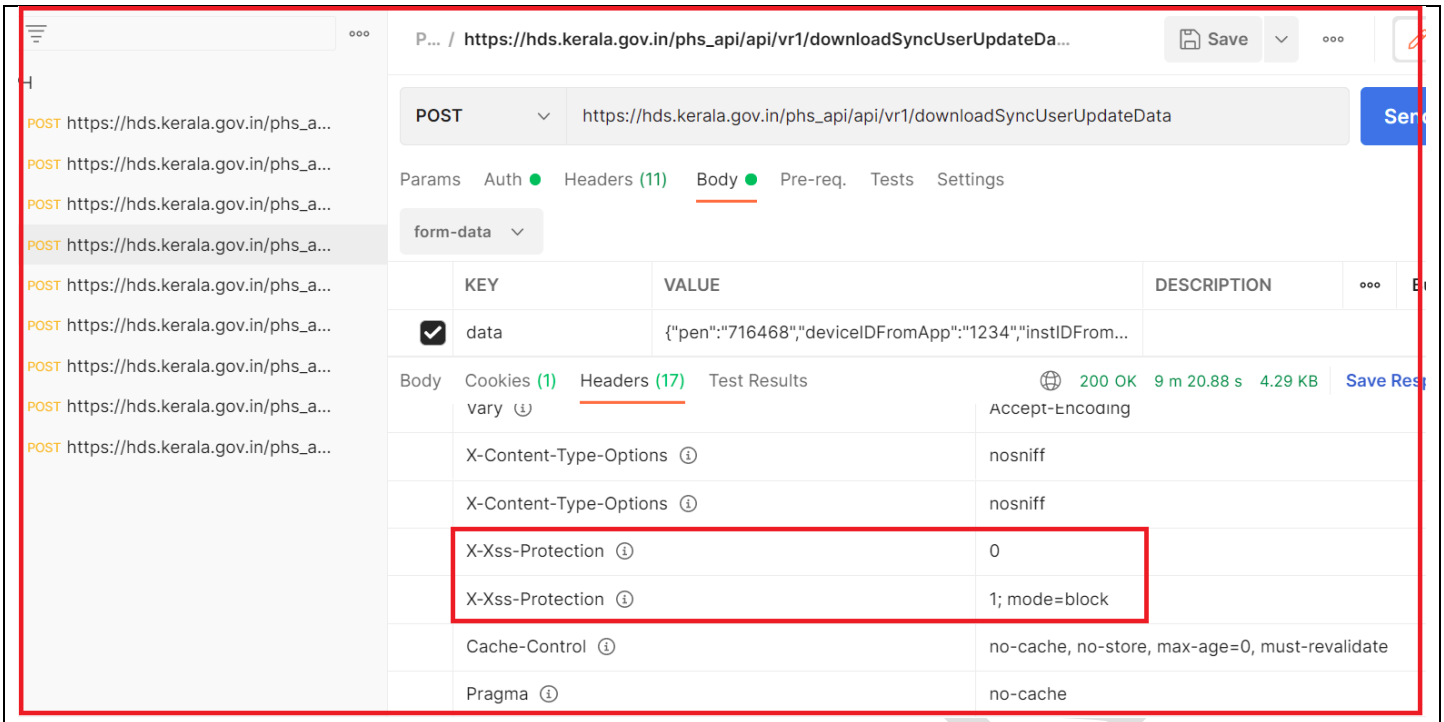
Impact

When multiple X-XSS-Protection headers are present, conflicting directives or policies may cause ambiguity about which directive to follow.

Recommendation

- **Centralized Configuration:** Centralize the configuration for setting the X-XSS-Protection header to prevent conflicting or duplicate directives from different parts of the application.
- **Code Review and Testing**: Review the code base thoroughly and test the application to identify and rectify any instances where the X-XSS-Protection header is being set redundantly.
- **Header Order and Priority**: Check the order in which headers are set or delivered. Ensure that the correct header with the appropriate value (e.g., X-XSS-Protection: 1; mode=block) takes precedence.

Proof of Concept (POC)



Reference
https://cwe.mitre.org/data/definitions/693.html

4.4.13 OPTIONS method is enabled		
Affected URL	**https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncMasData** **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncMasUpdateData** **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserData** **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserUpdateData** **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncAshabiData** **https://hds.kerala.gov.in/phs_api/api/vr1/uploadSyncPHData** **https://hds.kerala.gov.in/phs_api/api/vr1/generateOTP** **https://hds.kerala.gov.in/phs_api/api/vr1/verifyOTP** **https://hds.kerala.gov.in/phs_api/api/vr1/getHospitalTypeId**	
Severity	Medium	
CVSS	Base	4.6
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-650: Trusting HTTP Permission Methods on the Server Side	
Description		
The server supports the OPTIONS HTTP method. The OPTIONS method is used to determine what other methods the server supports for a given URI/resource.		
Impact		
The OPTIONS method exposes sensitive information that can help a malicious user to prepare more advanced attacks.		
Recommendation		
It is recommended to disable OPTIONS method in Apache. To do this, add the following code in the main httpd.conf file and restart apache.		

RewriteEngine On
RewriteCond %{REQUEST_METHOD} !^(GET|POST|HEAD)
RewriteRule .* - [R=405,L]

Proof of Concept (POC)

Reference

https://cwe.mitre.org/data/definitions/650.html

4.4.14 Improper Input Validation

App Name	**Ehealth PHS**	
Severity	**Medium**	
CVSS	Base	4.6
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-942: Permissive Cross-domain Policy with Untrusted Domains	

Description

The application is vulnerable to Improper input validation vulnerability. Improper input validation or unchecked user input vulnerability is caused when the application fails to validate or incorrectly validates input that can affect the control flow or data flow of the application. An input validation attack occurs when an attacker deliberately enters malicious input with the intention of confusing an application and causing it to carry out some unplanned action. Malicious input can include code, scripts and commands, which if not validated correctly can be used to exploit vulnerabilities. The most common input validation attacks include Buffer Overflow, XSS attacks, SQL injection etc. Input Validation should not be used as the primary method of preventing XSS, SQL Injection and other attacks but can significantly contribute to reducing their impact if implemented properly.

Impact

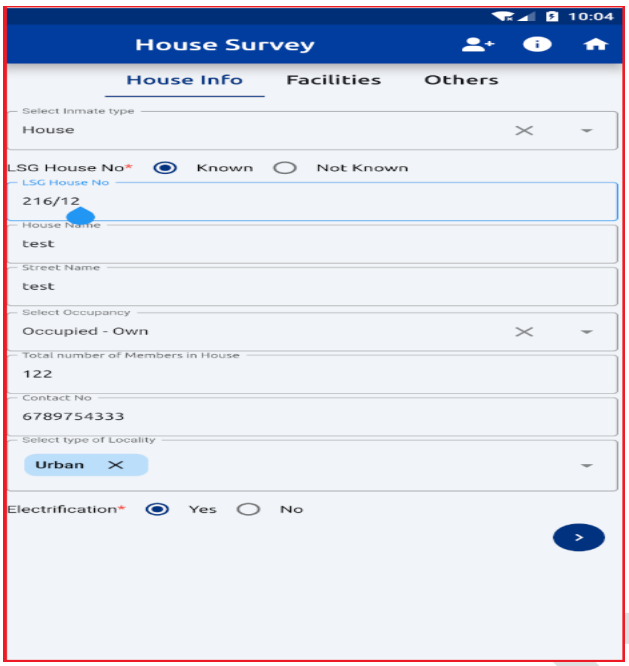
Improper input validation can lead to injection attacks, where malicious code or commands are inserted into input fields. Malicious or malformed input can cause data corruption, leading to inaccurate or inconsistent information stored in databases or processed by applications.

Recommendation

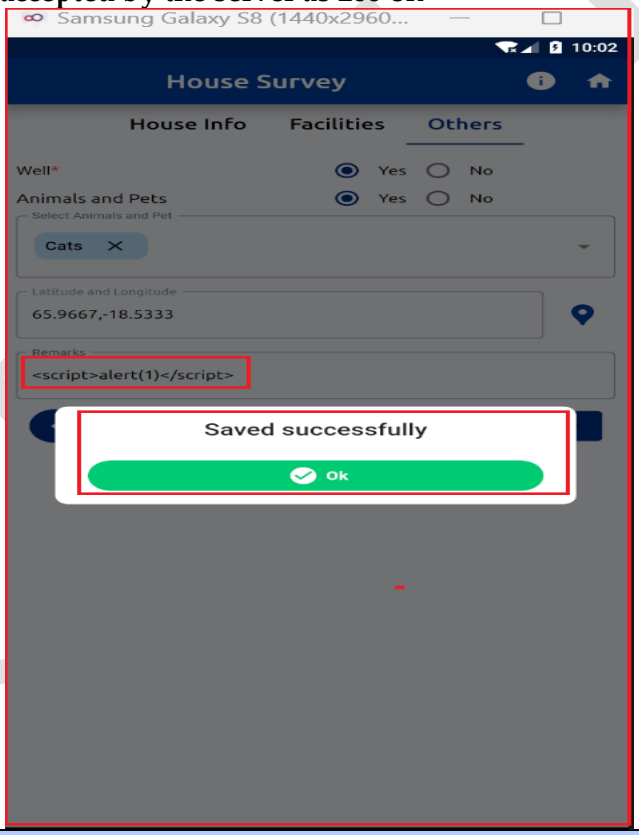
- Validation against JSON Schema and XML Schema (XSD) for input in these formats.
- Type conversion (e.g. Integer.parseInt() in Java, int() in Python) with strict exception handling
- Minimum and maximum value range check for numerical parameters and dates, minimum and maximum length check for strings.
- Array of allowed values for small sets of string parameters (e.g. days of week).
- Regular expressions for any other structured data covering the whole input string (^...$) and not using "any character" wildcard (such as . or \S).

Proof of Concept (POC)

Step 1: After logging with URL, Insert a payload into the parameter "Remarks"



Step 2: The value inserted is accepted by the server as 200 ok



Reference
https://cwe.mitre.org/data/definitions/20.html

4.5 Client Code Quality

4.5.1 SQL Injection

Affected URL	• **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncMasData** • **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncMasUpdateData** • **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserData** • **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncUserUpdateData** • **https://hds.kerala.gov.in/phs_api/api/vr1/downloadSyncAshabiData** • **https://hds.kerala.gov.in/phs_api/api/vr1/uploadSyncPHData** • **https://hds.kerala.gov.in/phs_api/api/vr1/generateOTP** • **https://hds.kerala.gov.in/phs_api/api/vr1/verifyOTP** • **https://hds.kerala.gov.in/phs_api/api/vr1/getHospitalTypeId**	
Severity	**High**	
CVSS	Base	7.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-20: Improper Input Validation	

Description

This script is possibly vulnerable to SQL Injection attacks. SQL injection is a vulnerability that allows an attacker to alter backend SQL statements by manipulating the user input. An SQL injection occurs when web applications accept user input that is directly placed into a SQL statement and doesn't properly filter out dangerous characters. This is one of the most common application layer attacks currently being used on the Internet. Despite the fact that it is relatively easy to protect against, there is a large number of web applications vulnerable.

Impact

An attacker can execute arbitrary SQL statements on the vulnerable system. This can compromise the integrity of your database and/or expose sensitive information. Depending on the back-end database in use, SQL injection vulnerabilities lead to varying levels of data/system access for the attacker. It is possible to not only manipulate existing queries, but to UNION in arbitrary data, use sub selects, or append additional queries. In some cases, it is possible to read in or write out to files, or to execute shell commands on the underlying operating system. Certain SQL Servers such as Microsoft SQL Server contain stored and extended procedures (database server functions). If an attacker can obtain access to these procedures it is possible to compromise the entire machine.

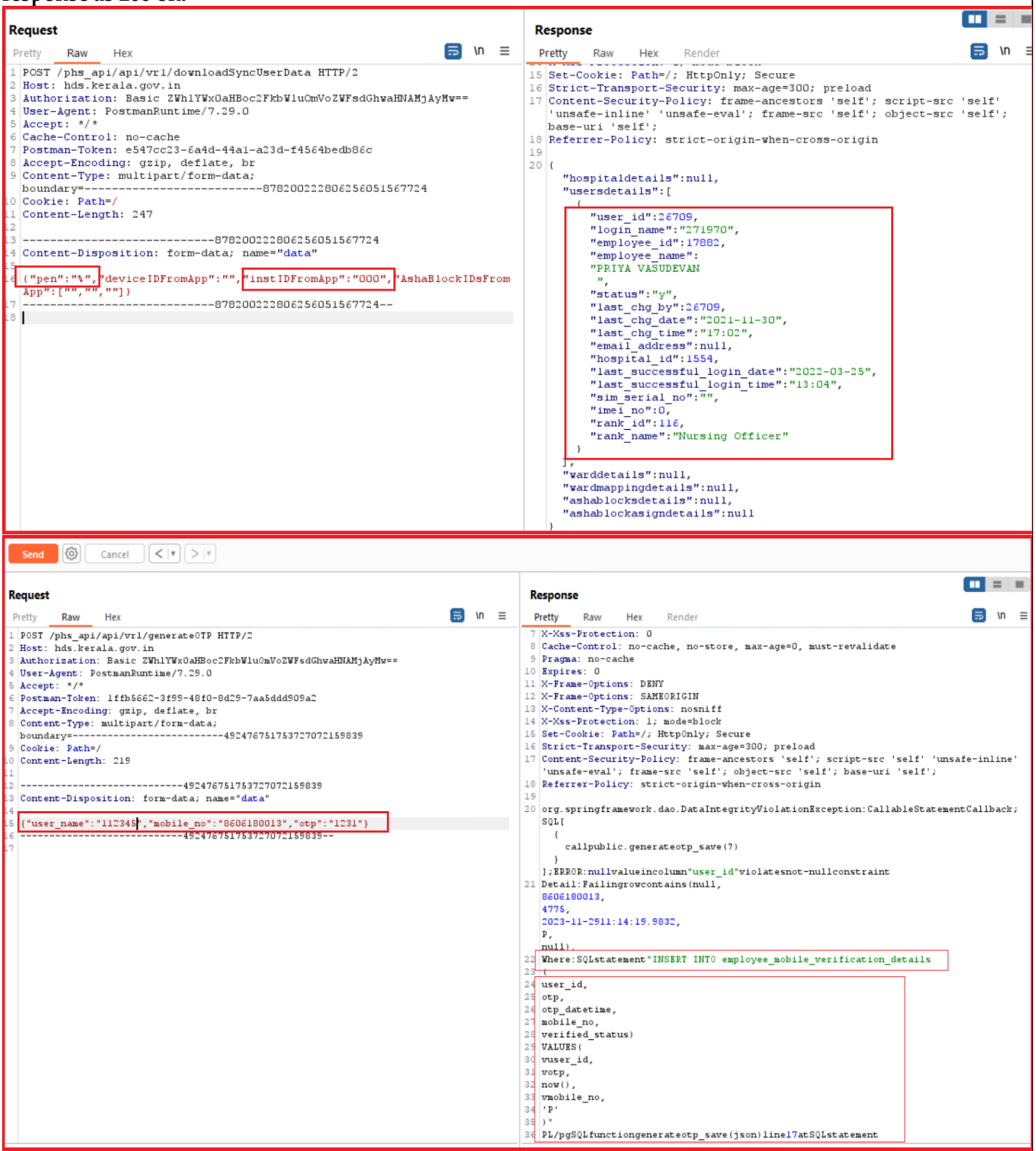
Recommendation

1. Use of prepared statements (Parameterized queries): Parameterized queries force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied. It ensures that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker.
2. Stored Procedures: They require the developer to define the SQL code first, and then pass in the parameters after. The difference between prepared statements and stored procedures is that the SQL code for a stored procedure is defined and stored in the database itself, and then called from the application.
3. Escaping All User Supplied Input: Each DBMS supports one or more character escaping schemes specific to certain kinds of queries. If all users supplied input are escaped using the proper escaping scheme for the database using, the DBMS will not confuse that input with SQL code written by the developer, thus avoiding any possible SQL injection vulnerabilities.
4. Least Privilege: To reduce the potential damage of a successful SQL injection attack, minimize the privileges assigned to every database account in the environment. Do not assign DBA or admin type access rights to the application accounts.

5. White List Input Validation: Input validation can be used to detect unauthorized input before it is passed to the SQL query.

Proof of Concept (POC)

Step 1: The request and response After adding a payload in the parameter "pen". The server accept the response as 200 ok.



Reference

https://owasp.org/www-community/attacks/SQL_Injection

4.5.2 Cross Site Scripting(Reflected)		
Affected URL	**https://hds.kerala.gov.in/phs_api/api/vr1/uploadSyncPHData**	
Severity	**High**	
CVSS	Base	7.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CVE-2023-30777- XSS Vulnerability	

Description

The script is vulnerable to Cross Site Scripting (XSS) attacks. Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of JavaScript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies orsession tokens retained by the browser.

Impact

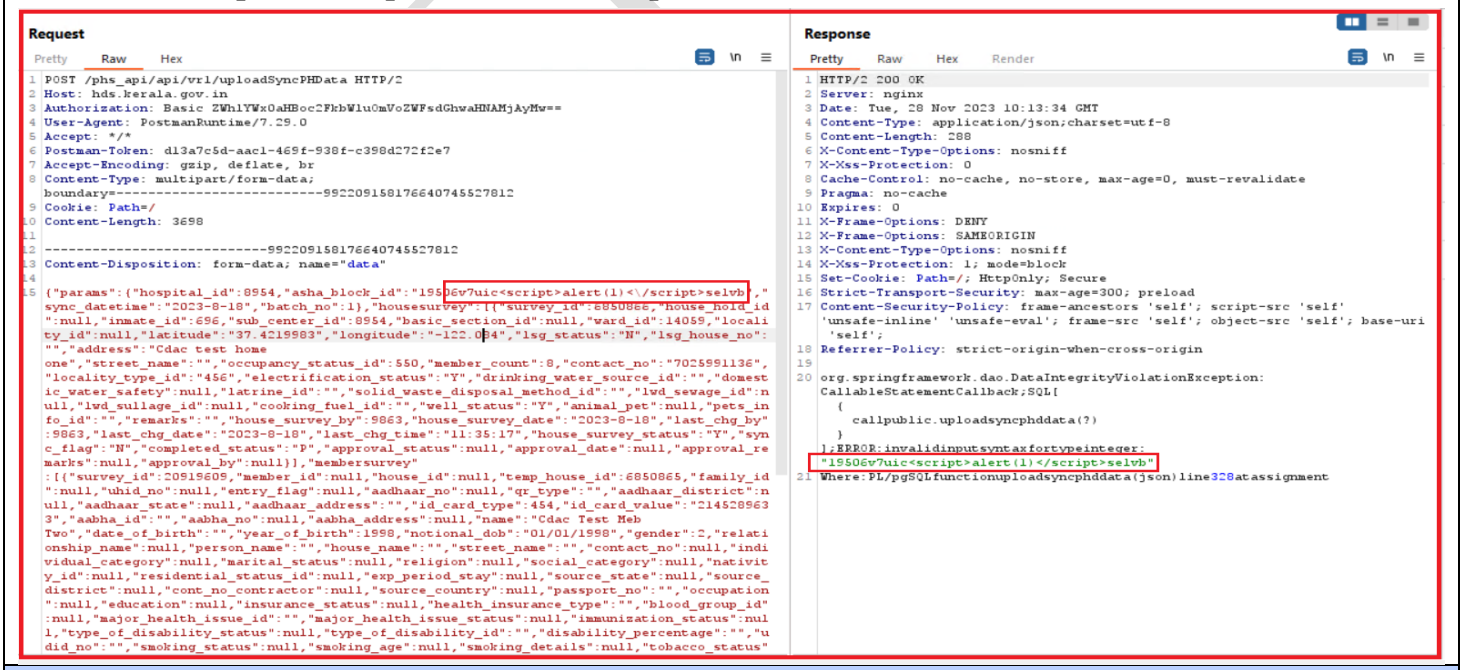
Malicious users can inject JavaScript, VBScript, ActiveX, HTML or Flash into a vulnerable application to fool a user in order to gather data from them. An attacker can steal the session cookie and take over the account, impersonating the user. It is also possible to modify the content of the page presented to the user.

Recommendation

- Never Insert Untrusted Data except in Allowed Locations
- HTML Escape before Inserting Untrusted Data into HTML Element Content
- Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes
- JavaScript Escape before Inserting Untrusted Data into JavaScript Data Value
- CSS Escape and Strictly Validate Before Inserting Untrusted Data into HTML Style Property Value
- URL Escape before Inserting Untrusted Data into HTML URL Parameter Values
- Sanitize HTML Mark-up with a Library Designed for the Job
- Implement Content Security Policy

Proof of Concept (POC)

Step 1: In the request a Script payload <script>alert (1)</script>was injected in the asha block ID, Server accepts the response and the script is reflected.



Reference

https://owasp.org/www-community/attacks/xss/

4.5.3 App uses SQLite Database and execute raw SQL query		
Affected File	**e/e/a/r.java**	
Severity	**Medium**	
CVSS	Base	4.8
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	

Description

When an app uses a SQLite database and executes raw SQL queries, it means that the app directly interacts with the database by sending SQL commands without employing a higher-level abstraction or ORM (Object-Relational Mapping) framework.

Impact

Improper handling or construction of raw SQL queries can lead to SQL injection vulnerabilities. Attackers could exploit this vulnerability to manipulate or access unauthorized data, compromise the database, or execute malicious SQL commands.

Recommendation

- Validate and sanitize user inputs to prevent the injection of malicious code into SQL queries. Validate input data types and use whitelisting approaches wherever possible.
- Consider using Object-Relational Mapping (ORM) libraries or database abstractions provided by Android frameworks (such as Room Persistence Library) to abstract away direct SQL queries. These frameworks handle many security concerns and reduce the risk of SQL injection.

Proof of Concept (POC)

r.java

```
1.  package e.e.a;
2.
3.  import android.content.Context;
4.  import android.database.Cursor;
5.  import android.database.DatabaseErrorHandler;
6.  import android.database.SQLException;
7.  import android.database.sqlite.SQLiteCantOpenDatabaseException;
8.  import android.database.sqlite.SQLiteCursorDriver;
9.  import android.database.sqlite.SQLiteDatabase;
10. import android.database.sqlite.SQLiteQuery;
11. import android.util.Log;
12. import g.a.c.a.k;
13. import java.io.File;
14. import java.util.ArrayList;
15. import java.util.Arrays;
16. import java.util.HashMap;
17. import java.util.List;
18. import java.util.Map;
19. /* JADX INFO: Access modifiers changed from: package-private */
20. /* loaded from: classes.dex */
21. public class r {
```

Reference

https://developer.android.com/privacy-and-security/risks/sql-injection

4.6 Insufficient Cryptography

4.6.1 App can be run in the rooted Environment

App Name	**E-Health PHS**	
Severity	**Medium**	
CVSS	Base	5.2
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-250: Execution with Unnecessary Privileges	

Description

When an app can be run in a rooted environment, it means the app is compatible with devices that have undergone rooting a process that grants users elevated privileges, allowing them to access and modify system files and settings that are typically restricted.

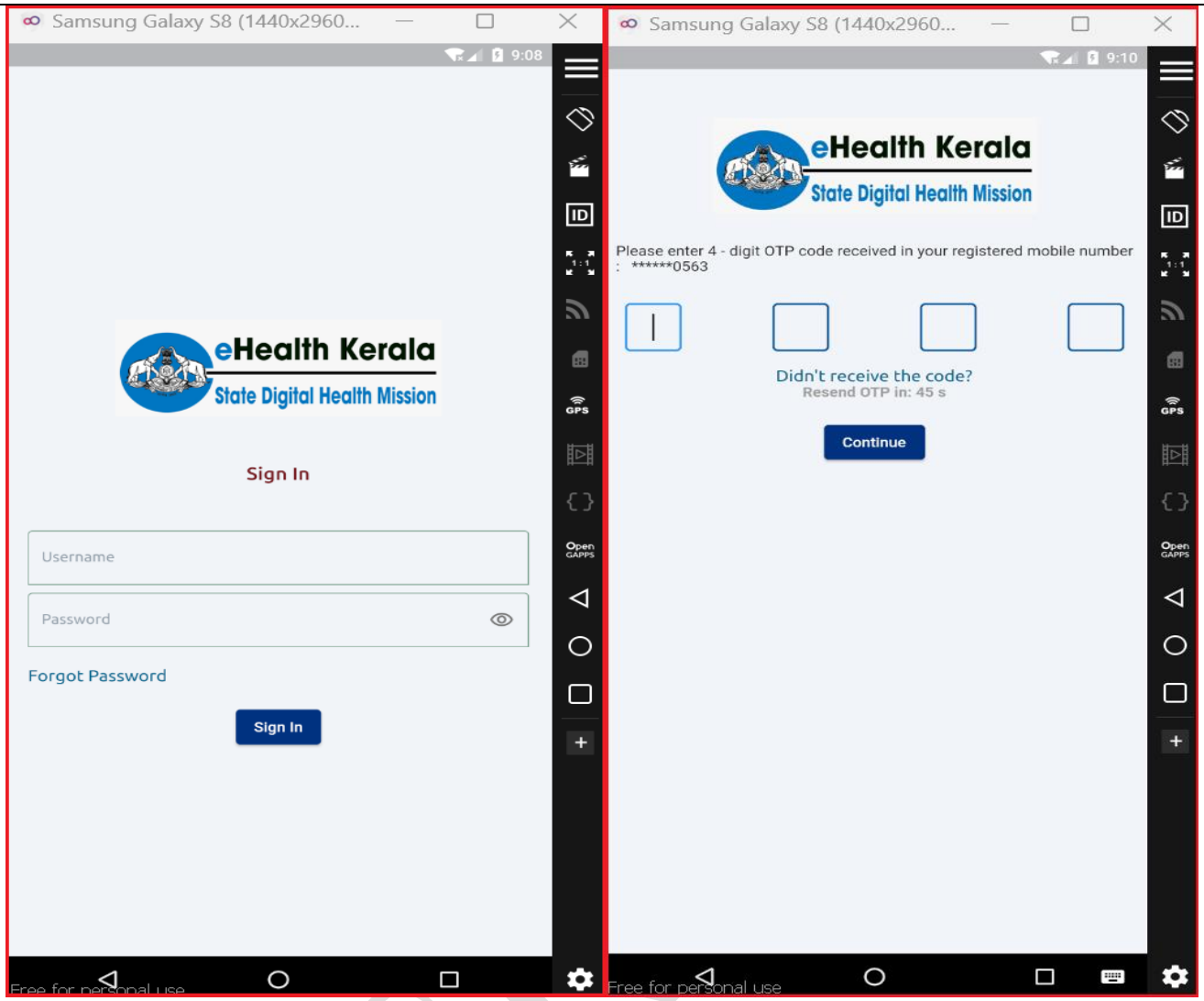
Impact

Running an app in a rooted environment poses higher security risks due to the increased access and permissions. It can potentially bypass security measures designed to protect the device and its data.

Recommendation

- **Security Checks:** Implement security checks within the app to detect if it's running on a rooted device. This can help prevent sensitive operations or data access if the device is rooted.
- **Limit Root Access:** Design the app to function properly without requiring root access whenever possible. Avoid relying on root permissions for normal app operations.
- **Security Hardening:** Apply additional security measures within the app to mitigate the risks associated with running in a rooted environment, such as encryption of sensitive data and validating inputs rigorously.

Proof of Concept (POC)



Reference
https://www.indusface.com/learning/how-to-implement-root-detection-in-android-applications/

4.6.2 SHARED LIBRARY BINARY ANALYSIS

Affected	• **lib/arm64-v8a/libapp.so** • **lib/arm64-v8a/libflutter.so** • **lib/armeabi-v7a/libapp.so** • **lib/armeabi-v7a/libflutter.so** • **lib/x86_64/libapp.so** • **lib/x86_64/libflutter.so** • **lib/arm64-v8a/libapp.so** • **lib/arm64-v8a/libflutter.so** • **lib/armeabi-v7a/libapp.so** • **lib/armeabi-v7a/libflutter.so**	
Severity	**Medium**	
CVSS	Base	4.6
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	

CVE	CWE-649: Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking

Description

These are collections of precompiled code and resources that multiple applications can use, offering functionalities such as APIs, frameworks, or utilities.

Impact

Shared libraries might contain vulnerabilities or weaknesses that could be exploited by multiple applications using them, leading to security breaches or data exposure.

Recommendation

Conduct regular security audits and analysis of shared libraries to identify potential vulnerabilities, outdated dependencies, or insecure coding practices.

Proof of Concept (POC)

lib/armeabi-v7a/libflutter.so	**True** info The binary has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	**True** info This binary has a stack canary value added to the stack so that it will be overwritten by a stack buffer that overflows the return address. This allows detection of overflows by verifying the integrity of the canary before function return.	**None** info The binary does not have run-time search path or RPATH set.	**None** info The binary does not have RUNPATH set.	**False** info The binary does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions. This check is not applicable for Dart/Flutter libraries.	**True** info Symbols are stripped.

Reference

https://cwe.mitre.org/data/definitions/649.html

4.6.3 The App uses an insecure Random Number Generator

Affected	• **h/w/a.java** • **h/w/b.java** • **h/w/d/a.java**	
Severity	**Medium**	
CVSS	Base	4.6
	(AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L)	
CVE	CWE-330: Use of Insufficiently Random Values	

Description

An insecure random number generator (RNG) in an application refers to the usage of a flawed or weak algorithm for generating random numbers. These insecure RNGs often produce predictable or easily guessable sequences, which can compromise the security of various functionalities relying on randomness, such as encryption, authentication, or session management.

Impact

Insecure RNGs may produce patterns or sequences that can be predicted or replicated, enabling attackers to guess or infer generated numbers.

Recommendation

Implement secure random number generation algorithms specifically designed for security-sensitive applications. Use libraries or functions known for cryptographic strength and unpredictability.

Proof of Concept (POC)

a.java

```
1.  package h.w;
2.
3.  import java.util.Random;
4.  /* loaded from: classes.dex */
5.  public abstract class a extends c {
6.      @Override // h.w.c
7.      public int b() {
8.          return c().nextInt();
9.      }
10.
11.     public abstract Random c();
12. }
```

Reference

https://owasp.org/www-community/vulnerabilities/Insecure_Randomness

5. General Recommendation

Operating system

- Keep operating system up to date with the latest service pack and patches to protect against common attacks.
- Remove all unneeded users and groups.
- Disable unwanted services and protocols
- Configure access controls for all protected files, directories, devices.
- Restrict access of operating system source files, configuration files, and their directories to authorized administrators.
- Use Strong Password for user accounts.
- Antivirus for scanning/detecting/cleaning must be installed on the computer system and it should be regularly updated.
- Regularly review the server log files (Access/Error/Security logs) for knowing any attacks and intrusions, preferably daily. Logs with HTTP error code 403 (Forbidden) and 404 (Not found) may be malicious attempts. Any actual incident may be reported to cert.ksitm@kerala.gov.in.
- Take regular back up of application and db.

Web Server

- Install the latest version of the web server software along with the latest security patches.
- Install only the required features of the application server and remove default features not being used.
- Remove all sample files scripts, manuals and executable code from the web server application root directory.
- Remove all files that are not part of the website.
- Remove the web server banner information so that web server and operating system type and version are not reported.
- Make sure that a dedicated User account with limited privileges should be used for the Web Server Processes.
- All default user names and IIS/apache pages (like admin, default.aspx, index.aspx...etc) should be renamed.

- The configuration files of the web server process should be readable by web server process but not writable.
- Consider security implications before selecting programs, scripts and plug-ins for the web server.
- Third party free modules available should not be used without proper checking and verification of their functionality and security.
- Use the correct CHMOD for each folder and file-Setting files or folders to a CHMOD of 777 or 707 is only necessary when a script needs to write to that file or directory.
- Use web application firewall for monitoring/ filtering the request in order to prevent hacking attempts.

API Best Practices

❖Authentication and Authorization

- Use strong authentication mechanisms such as API keys, OAuth, or JWT (JSON Web Tokens)
- If using tokens like JWT, set appropriate expiration times and implement secure token management practices to prevent token misuse or replay attacks.
- Implement granular access control to limit API access based on user roles and permissions.
- Always validate user credentials and tokens before granting access to sensitive data.

❖API Gateway and Firewall

- Employ an API gateway for centralized security enforcement, monitoring and management.
- Implement web application firewall (WAF) to protect against common web threats.

❖Data Protection and Secure Communication

- Encrypt sensitive data using appropriate encryption algorithms and key management.
- Apply data masking techniques to hide sensitive information in logs and responses.
- Use secure communication protocols to prevent eavesdropping and man in-the-middle attacks.
- Employ secure headers and practices to prevent information leakage.

❖ Input Validation and Sanitization

- All user inputs should be validated and sanitized to prevent injection attacks (e.g., SQL injection, XSS) and parameter manipulation.

❖ Output Encoding

- Encode output to protect against HTML/JavaScript injection (XSS) and other data manipulation attacks.

❖ Rate Limiting and Throttling

- Implement rate limiting and throttling mechanisms to prevent abuse of the API and DDoS attacks by limiting the number of requests from a single client within a specific time frame.

❖ Error Handling and Logging

- Ensure proper error handling to avoid exposing sensitive information.
- Implement comprehensive logging for monitoring and auditing purposes.

❖ CORS (Cross-Origin Resource Sharing)

- Configure CORS properly to restrict which domains can access the API from the client-side, thereby preventing unauthorized cross-origin requests.

❖ Secure Storage of Secrets

- Store API keys, credentials, and sensitive data securely using encryption and access controls

Mobile Application Best Practices

- **Secure Authentication and Authorization:**
  - Use strong authentication mechanisms like biometrics, multi-factor authentication (MFA), or OAuth.
  - Implement proper session management to prevent unauthorized access.
- **Data Encryption and Secure Storage:**
  - Encrypt sensitive data both in transit and at rest.
  - Leverage secure storage mechanisms provided by the mobile platform.
- **Input Validation and Sanitization:**
  - Validate and sanitize all user inputs to prevent injection attacks (SQL injection, XSS, etc.).
- **Secure Communication:**
  - Use secure communication protocols (HTTPS) for all network interactions.
  - Implement certificate pinning to prevent man-in-the-middle attacks.
- **Code Protection and Obfuscation:**
  - Apply code obfuscation techniques to deter reverse engineering.
  - Use tools and methods to detect and prevent tampering with the app's code.
- **Secure APIs and Backend:**
  - Ensure API endpoints are properly secured with authentication and authorization mechanism
  - Regularly update and patch server-side software and libraries.
- **App Permissions and Least Privilege:**
  - Request only necessary permissions and explain to users why each permission is required.
  - Implement the principle of least privilege to limit access to sensitive resources.
- **Regular Security Testing and Audits:**
  - Conduct regular security assessments, including penetration testing and code reviews.
  - Use automated security testing tools to identify vulnerabilities.
- **Compliance with Standards and Regulations:**
  - Adhere to industry standards (like OWASP Mobile Top 10) and comply with relevant data protection regulations (GDPR, HIPAA, etc.)

6. Conclusion

During the vulnerability assessment, it is observed that the APK has some High/Medium severity vulnerabilities. Hence it is advised to fix the vulnerabilities in the APK as per the recommendations given above and the corrected APK has to be submitted to CERT-K for a subsequent round of auditing.