Security Analysis of "Redbus" Mobile Application

Purbak Sengupta - 202310899@post.au.dk - AU759326
Kanak Rana - 202310900@post.au.dk - AU759379
August 11, 2024

Abstract

This report carries out a security analysis of the Redbus mobile application developed by Pilani Soft Labs Pvt. Ltd. A threat model has been composed, identifying various potential threats to the application. The security analysis includes software and network security assessments using freely available tools, as well as an attempt to perform a man-in-the-middle(MITM) attack, which successfully exposed sensitive user data. The report further describes the authentication mechanisms and flows within the application, revealing significant weaknesses. Furthermore, the application's handling of sensitive data, permissions and the presence of multiple trackers are analyzed, highlighting critical vulnerabilities that compromise user privacy and security.

I INTRODUCTION

The RedBus app, developed by Pilani Soft Labs Pvt. Ltd., offers a convenient platform for booking bus tickets across India. As digitization continues to grow, ensuring the security and privacy of users' personal data has become very important. This report provides a comprehensive security analysis of the RedBus app, highlighting potential threats and vulnerabilities. Using freely available software tools, we assess both software and network security, including an attempt to perform man-in-the-middle attack to uncover any exposure of sensitive data.
Given that most people book their travel tickets through digital devices now a days and hence it is important to secure all the user data like payment details or travel booking code in such a way that no external adversary can get hold of them. This report covers the authentication mechanisms, handling of sensitive data and required permissions within the application providing insights into its overall security posture.
Through examining these aspects, it is our end objective to expose all the vulnerabilities and provide some possible recommendations to strengthen the security of RedBus application so that a user can use the app with confidence without bothering about his data getting compromised.

II DESCRIPTION OF THE APPLICATION

RedBus is a popular app for booking bus tickets in India, with 3.6 Cr of users. It is part of India's leading online travel company MakeMyTrip Limited. It allows users to search for routes, book tickets, and check seat availability[1]. Users can create accounts using their phone numbers or Google accounts, providing personal information that the app uses to enhance the booking experience. RedBus App version 22.9.5. is analyzed in this report.

A. Threat Model

Given the sensitivity of personal data handled by RedBus, such as payment information, personal details, and travel history, maintaining a strong security posture is critical. Potential threats include:

- Internal Threats: Employees or insiders with access to sensitive systems might misuse or leak data for personal gain or due to negligence, leading to data breaches or unauthorized data access.
- Cyber-crimes: Hackers and cyber-criminals may target the platform to steal payment card information, personal details, or hijack accounts for fraudulent activities. This can lead to financial losses and damage to the company's reputation.
- Competitors: Rival companies might engage in espionage to gain market insights or acquire sensitive business information, such as customer data or proprietary algorithms, to undermine RedBus's competitive advantage.
- External Threats: Organized cybercrime groups and state-sponsored attackers may target RedBus for large-scale attacks, exploiting vulnerabilities in the system to cause disruption or extort the company. These threats necessitate robust security measures to protect user data and ensure the integrity and confidentiality of the services provided by RedBus.

B. Relevant Security Properties

To protect its users, the RedBus app must adhere to several security properties:

- Confidentiality: Ensure that user data, including personal and payment details, are secure and cannot be accessed by unauthorized parties.
- Integrity: Prevent unauthorized alterations of data, ensuring users' booking information remains accurate and unaltered.
- Authenticity: Confirm that only legitimate users can access their accounts and booking information. It would be dangerous for an adversary to claim someone's identity and getting access to the data.

- Non-repudiation: Guarantee that actions taken by users, such as bookings, are recorded and verifiable. Users of the application must be assured that the sender receives proof of delivery and the recipient receives proof of the sender's identity, ensuring that neither party can later deny having handled the information.

- Reliability: The application must ensure its reliability. Ensure that the app and its services are consistently available and perform as expected. Users need to trust how the app handles their data. When users record important information, they should be confident that the data stored in the database matches exactly what they entered.

- Privacy: Protect users' personal and financial information from being exposed or misused.

- Anonymity: It's important to minimize the collection of personally identifiable information and avoid associating it with specific transactions unless absolutely necessary. This way, the application can protect users' privacy and ensure that their personal details are only used when it is essential.

- Availability: Maintain service availability to prevent user dissatisfaction and potential loss of business. However, this is not directly correlated to security.

C. Attack Surfaces

Potential attack surfaces for RedBus include:

- User Devices: If an unauthorized person gains physical access to a user's device, they could easily open the RedBus app without needing to enter a password. This situation could lead to a serious breach of user data and personal information. Therefore, it's crucial to implement additional security measures such as biometric authentication, two-factor authentication and automatic logout after a period of inactivity to protect the app and user data even if the device falls into the wrong hands.

- Network Security: Man-in-the-Middle (MITM) attacks pose a significant threat by intercepting sensitive information transmitted over public or insecure networks[2]. This means that any data you send or receive could be captured and potentially misused by attackers. To safeguard against this, it's essential to use strong encryption methods, secure communication protocols and implement additional security measures such as VPNs or secure Wi-Fi connections to ensure your data remains private and protected while being transmitted[3].

- Third-party Integrations: Integrating with third-party services can introduce vulnerabilities if those services are not secure. This means that any weak points in the third-party systems could be exploited by attackers, potentially compromising the security of the application.

- Backend Servers: Attacks on backend infrastructure, including databases that store sensitive user information, pose significant risks to data security. If adversaries gain access to these servers, they could steal, alter, or delete critical data, leading to severe privacy breaches and potential harm to users.

III ANALYSIS

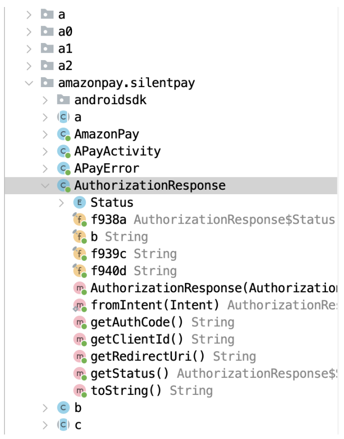
This part covers a thorough analysis of the "Redbus" Mobile application. It is further divided into four parts:

- Software security
- Network security
- Authentication
- Privacy

All these parts are very essential in terms of security. We have used some pre-existing software tools to investigate and discover potential security vulnerabilities.

A. Software Security

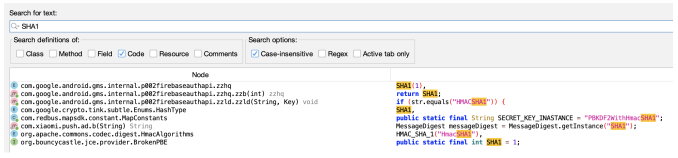
To conduct the software security analysis we did static analysis of the "Redbus apk v22.9.5" first. We used APKtool[4] and JADX[5] for decompilation of the apk into JAVA code. The decompiled codes were highly obfuscated for both cases(see fig 1). obfuscation simply means turning



```
> a
> a0
> a1
> a2
v amazonpay.silentpay
  > androidsdk
  > a
  > AmazonPay
  > APayActivity
  > APayError
  v AuthorizationResponse
    > Status
      f938a AuthorizationResponse$Status
      b String
      f939c String
      f940d String
      AuthorizationResponse(Authorizatio
      fromIntent(Intent) AuthorizationRe
      getAuthCode() String
      getClientId() String
      getRedirectUri() String
      getStatus() AuthorizationResponse$
      toString() String
  > b
  > c
```

Figure 1: Obfuscation level of the decompiled code

the application code into some modified version that makes it difficult to understand and hard to follow. This way, code structure can be hidden and attackers cannot simply perform any kind of attack on the application as it successfully hides informations that might open up ways for the attackers to perform an attack. First of all, we conducted a thorough searching through different software security related keywords, such as AES, RSA, Key, Password, MD5, SHA1, SHA256 etc. We conducted the search in two ways: through terminal using "grep -r" command[6] and using JADX search option. After this, we used the deobfusca-



Search for text:
SHA1

Search definitions of: Class [✓ Code] Method Field Resource Comments
Search options: [✓ Case-insensitive] Regex Active tab only

Node

```
com.google.android.gms.internal.p002firebaseauthapi.zzhq                       SHA1(1),
com.google.android.gms.internal.p002firebaseauthapi.zzhq.zzb(int) zzhq         return SHA1;
com.google.android.gms.internal.p002firebaseauthapi.zzld.zzld(String, Key) void if (str.equals("HMACSHA1")) {
com.google.crypto.tink.subtle.Enums.HashType                                   SHA1,
com.redbus.mapsdk.constant.MapConstants                                        public static final String SECRET_KEY_INASTANCE = "PBKDF2WithHmacSHA1";
com.xiaomi.push.ad.b(String) String                                            MessageDigest messageDigest = MessageDigest.getInstance("SHA1");
org.apache.commons.codec.digest.HmacAlgorithms                                 HMAC_SHA_1("HmacSHA1"),
org.bouncycastle.jce.provider.BrokenPBE                                        public static final int SHA1 = 1;
```

Figure 2: Extensive keywords searching in JADX

tion tool in JADX, but it was asking to increase the heap memory and restart the GUI to conduct the deobfuscation. So we opened the the "jadx-gui" in a text editor and modified the 'DEFAULT_JVM_OPTS' line by adding '-Xmx12g' which set the heap size to 12 GB[7]. Then we reopened the JADX-GUI and conducted the deobfuscation. Now, we looked into the code again and found out some known structure and recongnized some folder and class names. However, it was still very unclear and the extensive keywords searching also did not reveal any kind of potential security threats but we atleast got to know that the keywords are definitely being used in the codes without getting any clear context of their usage.
We moved forward to conduct static analysis and dynamic analysis and for that we used MobSF[8]. Mobile Security Framework(MobSF) is a security assessment framework to conduct static and dynamic analysis of mobile application binaries such as APKs. THe overall score of Redbus was 42/100 and it was graded as B (see fig 3) which indicates

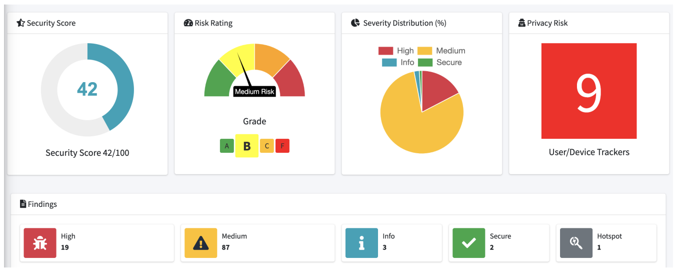
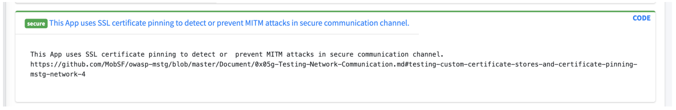


Figure 3: Result of static analysis

moderate to poor level of security. The risk rating suggests that the application has several vulnerabilities that could potentially be exploited. It is not highly vulnerable, but there is a substantial amount of risk that needs to be mitigated.
There were 19 **high** security vulnerabilities among which padding oracle attack, phising attack, usage of weak encryption algorithm threats were present. Also, there were 87 **medium** level security vulnerabilities among which insecure Random Number Generation(RNG) usage in the code, usage of MD5 and SHA1 and handcoded secrets were notable threats. SHA1 is used by SSL certificate authorities to sign certificate and is a previous state-of-the-art hash function. However, it has collision issues, thus latest SHA256 should be used instead of SHA1 and MD5[25]. SHA256 generates a hash value of 256 bit instead of 160 bit which improves the security[9]. 9 trackers were also detected which leaves major security concerns as it can collect and send user data to third parties [10], potentially leading to privacy breaches.
From the static analysis it was found that the application uses SSL certificate pinning (see fig 4). It is a process of

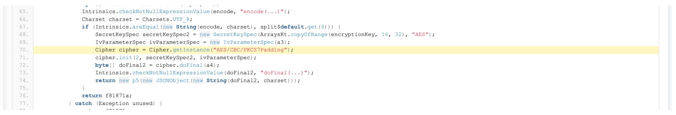


secure This App uses SSL certificate pinning to detect or prevent MITM attacks in secure communication channel. CODE

```
This App uses SSL certificate pinning to detect or  prevent MITM attacks in secure communication channel.
https://github.com/MobSF/owasp-mstg/blob/master/Document/0x05g-Testing-Network-Communication.md#testing-custom-certificate-stores-and-certificate-pinning-mstg-network-4
```

Figure 4: App uses SSL certificate pinning

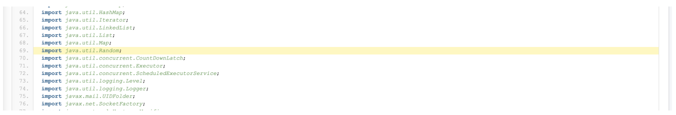
linking the back-end server with a specific pre-configured list of certificates or public keys, rather than accepting any certificate issued by a trusted certificate authority [11]. This technique is used to identify man-in-the-middle (MITM) attacks in secure communication channels to prevent any kind of MITM attack attempts. Thus, it should be able to prevent the MITM attack attempt we will make afterwards.
The application uses the encryption mode CBC with PKCS7 padding(see fig 5). This configuration is vulnerable to padding oracle attacks[12]. A padding oracle attack is a type of cryptographic attack where an attacker exploits the padding validation of encrypted messages to decrypt data or forge cryptographic signatures. In addition,



```java
        Intrinsics.checkNotNullExpressionValue(encode, "encode(...)");
        Charset charset = Charsets.UTF_8;
        if (Intrinsics.areEqual(new String(encode, charset), split$default.get(4))) {
            SecretKeySpec secretKeySpec2 = new SecretKeySpec(ArraysKt.copyOfRange(encryptionKey, 16, 32), "AES");
            IvParameterSpec ivParameterSpec = new IvParameterSpec(a3);
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS7Padding");
            cipher.init(2, secretKeySpec2, ivParameterSpec);
            byte[] doFinal2 = cipher.doFinal(a4);
            Intrinsics.checkNotNullExpressionValue(doFinal2, "doFinal(...)");
            return new p5(new JSONObject(new String(doFinal2, charset)));
        }
        return f81871a;
    } catch (Exception unused) {
```

Figure 5: Usage of CBC encryption mode with PKCS7 padding

the app uses 'java. util. Random'' to generate random numbers (see fig 6). This class is based on the system clock and therefore could be reproduced by an attacker, if they know when it was generated. The Random class is not a cryptographically secure random number generator (RNG) and should never be used in security-critical applications. Instead, the SecureRandom class generates its



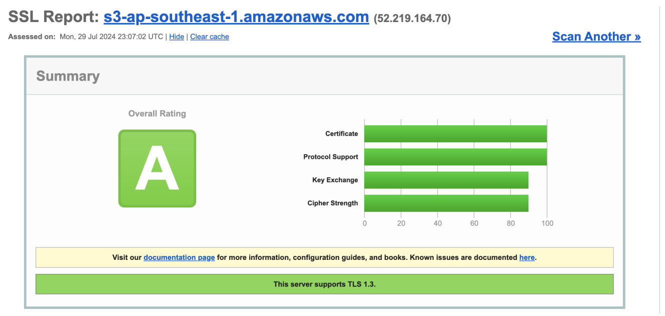
```java
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Random;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.Executor;
import java.util.concurrent.ScheduledExecutorService;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.mail.UIDFolder;
import javax.net.SocketFactory;
```

Figure 6: Insecure RNG usage

seed using random data provided by the operating system and is built to be cryptographically strong, making it appropriate for security-sensitive tasks [13], therefore should be used. .
Next, we conducted dynamic analysis of the APK where it passed TLS/SSL Security Tester. However, no further notable vulnerabilities were found.

B. Network Security

We successfully found the hostname using Wireshark[14] to which the application was communicating. To do that, we first installed an android emulator named Genymotion [15] (tried Android Studio as well). It provided a large set of Android devices with different Android versions. We used Samsung Galaxy S6 with Android 9.0(Pie) API 28 as it was compatible with our system. The hostnames detected analyzing the traffic in Wireshark was: "s3-ap-southeast-1.amazonaws.com". At first, Wireshark was only showing the IP address of the host with which the application was communicating. We then turned on the "Resolve Network Addresses" tool of Wireshark which immediately gave us the hostname associated with the IP[16].
Next, we moved forward to analyze the TLS-configuration on the server side for the hostname. The analysis was done using "Qualys SSL Labs"[17](see fig 7). "s3-ap-



SSL Report: s3-ap-southeast-1.amazonaws.com (52.219.164.70)
Assessed on: Mon, 29 Jul 2024 23:07:02 UTC | Hide | Clear cache
Scan Another »

Summary

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

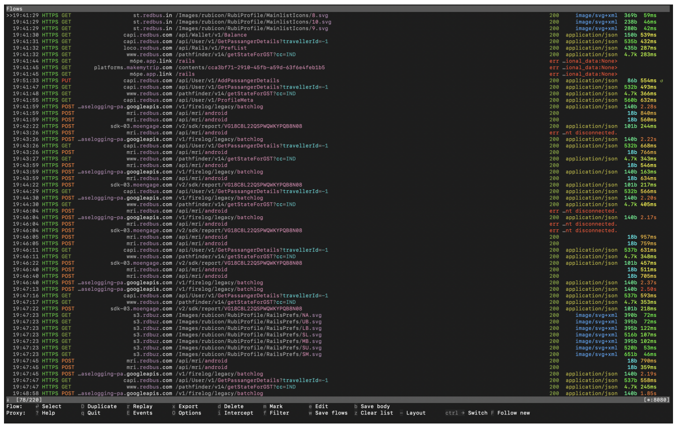
This server supports TLS 1.3.

Figure 7: Anlysis of the TLS-configuration

southeast-1.amazonaws.com" is configured to use TLS 1.3 and TLS 1.2. For any modern application or system, TLS 1.3 should be used whenever possible, with TLS 1.2 as a fallback[18]. TLS 1.3 is the most secure and efficient, with modern features and better performance, whereas TLS 1.2 has also strong security, but less efficient than TLS 1.3. However, both are more secure than its previous protocols such as TLS 1.0, TLS 1.1, SSL 2.0 and SSL 3.0. From fig 7 it is clear that the rating is quite high(A) and the reason is the server is using TLS 1.3.

Man In The Middle Attack(MITM)

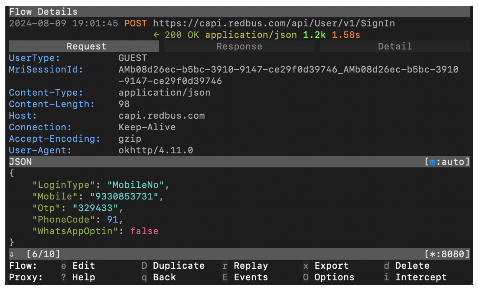
After the analysis of software and network security, we further moved on to attempt a man-in-the-middle attack(MITM)[2] mainly focusing on intercepting messages between the application and server. Now, we already discussed that the app uses SSL certificate pinning(from static analysis) so it should be able to prevent this MITM attack attempt.
At first, we tried to patch the Redbus apk using apk-mitm[19]. We downloaded mitmproxy into the host machine and pushed the mitmproxy CA certificate into the Genymotion emulator as we could not download it directly via web from *"http://mitm.it/"*. Samsung Galaxy S6 emulator with Android version 9.0 API 28 was used again to run the patched apk. Then the proxy settings of the emulator was configured to make sure mitmproxy and emulator run on the same network IP address and proxy port 8080. However, we faced some issued here. We could see the traffic at first, but the application was constantly crashing. We tried to figure out the problem, however we could not actually managed to get any proper conclusion so we tried to patch it again and retry the whole process. The result was the same that is the application was constantly crashing. We assumed that it is due to SSL pinning and the attempt is failed however we tried for last time without patching the apk. We expected the same result again as the use of apk-mitm is to bypass the SSL pinning and as we already knew from the starting that the application has SSL certificate pinning implemented, we expected this attempt to fail. Surprisingly, it worked and we intercepted the traffic successfully(see fig 8).



```
Flows
>>19:41:29 HTTPS GET                st.redbus.in /Images/rubicon/RubiProfile/MainlistIcons/8.svg          200     image/svg+xml  369b  59ms
  19:41:29 HTTPS GET                st.redbus.in /Images/rubicon/RubiProfile/MainlistIcons/10.svg         200     image/svg+xml  238b  46ms
  19:41:29 HTTPS GET                st.redbus.in /Images/rubicon/RubiProfile/MainlistIcons/9.svg          200     image/svg+xml  280b  42ms
  19:41:30 HTTPS GET              capi.redbus.com /api/Wallet/v1/Balance                                  200  application/json  150b 539ms
  19:41:31 HTTPS GET              capi.redbus.com /api/User/v1/GetPassangerDetails?travellerId=-1         200  application/json  535b 432ms
  19:41:32 HTTPS GET              loco.redbus.com /api/Rails/v1/PrefList                                  200  application/json  435b 287ms
  19:41:32 HTTPS GET               www.redbus.com /pathfinder/v14/getStateForGST?cc=IND                   200  application/json  4.7k 283ms
  19:41:44 HTTPS GET                m6pe.app.link /rails                                                  err …ional_data:None>
  19:41:45 HTTPS GET       platforms.makemytrip.com /contents/cca3bf71-2910-45fb-a59d-63f6e4feb1b5        err …ional_data:None>
  19:41:45 HTTPS GET                m6pe.app.link /rails                                                  err …ional_data:None>
  19:51:33 HTTPS PUT              capi.redbus.com /api/User/v1/AddPassangerDetails                        200  application/json   86b 554ms ↺
  19:41:47 HTTPS GET              capi.redbus.com /api/User/v1/GetPassangerDetails?travellerId=-1         200  application/json  532b 493ms
  19:41:48 HTTPS GET               www.redbus.com /pathfinder/v14/getStateForGST?cc=IND                   200  application/json  4.7k 366ms
  19:41:55 HTTPS GET              capi.redbus.com /api/User/v1/ProfileMeta                                200  application/json  560b 632ms
  19:41:59 HTTPS POST …aselogging-pa.googleapis.com /v1/firelog/legacy/batchlog                         200  application/json  140b 2.28s
  19:41:59 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 840ms
  19:41:59 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 560ms
  19:42:22 HTTPS POST         sdk-03.moengage.com /v2/sdk/report/VG18C8L22QSPWQWKYPQB8N08                 200  application/json  101b 244ms
  19:43:26 HTTPS POST              mri.redbus.com /api/mri/android                                        err …nt disconnected.
  19:43:26 HTTPS POST …aselogging-pa.googleapis.com /v1/firelog/legacy/batchlog                         200  application/json  140b 2.22s
  19:43:26 HTTPS GET              capi.redbus.com /api/User/v1/GetPassangerDetails?travellerId=-1         200  application/json  532b 668ms
  19:43:26 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 766ms
  19:43:27 HTTPS GET               www.redbus.com /pathfinder/v14/getStateForGST?cc=IND                   200  application/json  4.7k 343ms
  19:43:59 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 546ms
  19:43:59 HTTPS POST …aselogging-pa.googleapis.com /v1/firelog/legacy/batchlog                         200  application/json  140b 163ms
  19:43:59 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 634ms
  19:44:22 HTTPS POST         sdk-03.moengage.com /v2/sdk/report/VG18C8L22QSPWQWKYPQB8N08                 200  application/json  101b 217ms
  19:44:29 HTTPS GET              capi.redbus.com /api/User/v1/GetPassangerDetails?travellerId=-1         200  application/json  532b 566ms
  19:44:30 HTTPS POST …aselogging-pa.googleapis.com /v1/firelog/legacy/batchlog                         200  application/json  140b 2.20s
  19:44:30 HTTPS GET               www.redbus.com /pathfinder/v14/getStateForGST?cc=IND                   200  application/json  4.7k 405ms
  19:46:04 HTTPS POST              mri.redbus.com /api/mri/android                                        err …nt disconnected.
  19:46:04 HTTPS POST …aselogging-pa.googleapis.com /v1/firelog/legacy/batchlog                         200  application/json  140b 2.17s
  19:46:04 HTTPS POST              mri.redbus.com /api/mri/android                                        err …nt disconnected.
  19:46:04 HTTPS POST         sdk-03.moengage.com /v2/sdk/report/VG18C8L22QSPWQWKYPQB8N08                 err …nt disconnected.
  19:46:05 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 957ms
  19:46:05 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 759ms
  19:46:11 HTTPS GET              capi.redbus.com /api/User/v1/GetPassangerDetails?travellerId=-1         200  application/json  537b 631ms
  19:46:11 HTTPS GET               www.redbus.com /pathfinder/v14/getStateForGST?cc=IND                   200  application/json  4.7k 348ms
  19:46:22 HTTPS POST         sdk-03.moengage.com /v2/sdk/report/VG18C8L22QSPWQWKYPQB8N08                 200  application/json  101b 457ms
  19:46:40 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 511ms
  19:46:40 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 705ms
  19:46:40 HTTPS POST …aselogging-pa.googleapis.com /v1/firelog/legacy/batchlog                         200  application/json  140b 2.37s
  19:47:13 HTTPS POST …aselogging-pa.googleapis.com /v1/firelog/legacy/batchlog                         200  application/json  140b 2.50s
  19:47:16 HTTPS GET              capi.redbus.com /api/User/v1/GetPassangerDetails?travellerId=-1         200  application/json  537b 593ms
  19:47:17 HTTPS GET               www.redbus.com /pathfinder/v14/getStateForGST?cc=IND                   200  application/json  4.7k 353ms
  19:47:22 HTTPS POST         sdk-03.moengage.com /v2/sdk/report/VG18C8L22QSPWQWKYPQB8N08                 200  application/json  101b 218ms
  19:47:23 HTTPS GET                s3.rdbuz.com /Images/rubicon/RubiProfile/RailsPrefs/NA.svg            200     image/svg+xml  390b  72ms
  19:47:23 HTTPS GET                s3.rdbuz.com /Images/rubicon/RubiProfile/RailsPrefs/UB.svg            200     image/svg+xml  395b  72ms
  19:47:23 HTTPS GET                s3.rdbuz.com /Images/rubicon/RubiProfile/RailsPrefs/LB.svg            200     image/svg+xml  395b 122ms
  19:47:23 HTTPS GET                s3.rdbuz.com /Images/rubicon/RubiProfile/RailsPrefs/SL.svg            200     image/svg+xml  516b 107ms
  19:47:23 HTTPS GET                s3.rdbuz.com /Images/rubicon/RubiProfile/RailsPrefs/MB.svg            200     image/svg+xml  395b 102ms
  19:47:23 HTTPS GET                s3.rdbuz.com /Images/rubicon/RubiProfile/RailsPrefs/SU.svg            200     image/svg+xml  520b  53ms
  19:47:23 HTTPS GET                s3.rdbuz.com /Images/rubicon/RubiProfile/RailsPrefs/SM.svg            200     image/svg+xml  651b  46ms
  19:47:45 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 790ms
  19:47:45 HTTPS POST              mri.redbus.com /api/mri/android                                        200                     18b 359ms
  19:47:45 HTTPS POST …aselogging-pa.googleapis.com /v1/firelog/legacy/batchlog                         200  application/json  140b 2.19s
  19:47:47 HTTPS GET              capi.redbus.com /api/User/v1/GetPassangerDetails?travellerId=-1         200  application/json  537b 558ms
  19:47:47 HTTPS GET               www.redbus.com /pathfinder/v14/getStateForGST?cc=IND                   200  application/json  4.7k 245ms
  19:48:58 HTTPS POST …aselogging-pa.googleapis.com /v1/firelog/legacy/batchlog                         200  application/json  140b 1.85s
⇩  [78/220]                                                                                                                    [*:8080]
Flow:   ⏎ Select    D Duplicate  r Replay   x Export   d Delete     m Mark    e Edit        b Save body
Proxy:  ? Help      q Quit       E Events   O Options  i Intercept  f Filter  w Save flows  z Clear list  - Layout   ctrl → Switch F Follow new
```

Figure 8: Interception of data

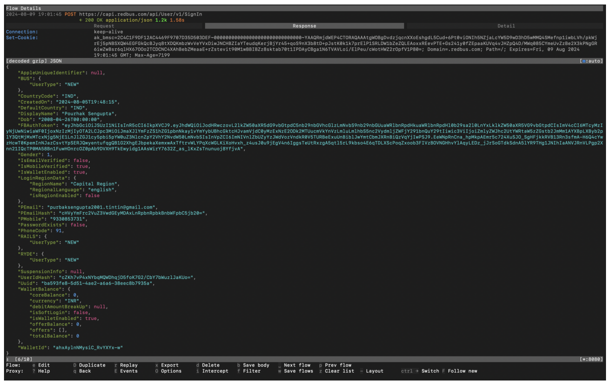
It seems that the application might not have implemented the SSL pinning successfully or properly in practice. We performed some activities within the application to generate traffic. First thing we did was to login to the system with phone number. The API was called and we logged into the system. We intercepted the traffic of this activity where the POST request is exposing my phone number and the OTP which I used for logging in(see fig 9). This is a major security vulnerability as my login credentials(phone number) now can be mishandled by an adversary. What



```
Flow Details
2024-08-09 19:01:45 POST https://capi.redbus.com/api/User/v1/SignIn
                         ← 200 OK application/json 1.2k 1.58s
        Request                  Response                  Detail
UserType:            GUEST
MriSessionId:        AMb08d26ec-b5bc-3910-9147-ce29f0d39746_AMb08d26ec-b5bc-3910
                     -9147-ce29f0d39746
Content-Type:        application/json
Content-Length:      98
Host:                capi.redbus.com
Connection:          Keep-Alive
Accept-Encoding:     gzip
User-Agent:          okhttp/4.11.0
JSON                                                                   [m:auto]
{
    "LoginType": "MobileNo",
    "Mobile": "9330853731",
    "Otp": "329433",
    "PhoneCode": 91,
    "WhatsAppOptin": false
}
⇩  [6/10]                                                              [*:8080]
Flow:    e Edit        D Duplicate   r Replay    x Export    d Delete
Proxy:   ? Help        q Back        E Events    O Options   i Intercept
```

Figure 9: POST request exposing credentials

more surprising was the response of this POST request where all of my details are exposed including email id, name, date of birth, wallet balance, wallet id, FBAuthToken etc(see fig 10). However, as we used the application

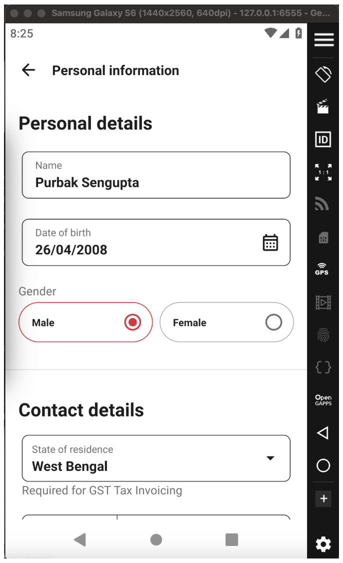


```
Flow Details
2024-08-09 19:01:45 POST https://capi.redbus.com/api/User/v1/SignIn
                         ← 200 OK application/json 1.2k 1.58s
                    Request                          Response                          Detail
Connection:               keep-alive
Set-Cookie:               ak_bmsc=2C4C1F9DF12AC4469F9707D35D503DEF~000000000000000000000000000000~YAAQRmjdWEP4CTORAQAAAtgWOBgDvdzjqcnXXoEshgdL5Cud+6Pt0viONIh5NZjaLcYW5D9wD3hD5wMMQ4SMefnp1iwbLVh/pkWj
                          rEj5pNBSXQW4EGFGkQcBJyq8tXDQKmbzWvVeYVxDiwJNCHBZIaYTeudqKerjBjYr45+qoS9nX3b8tD+pJstK0k1k7prElP1SRLDW1bZeZQLEAoxxREevPTE+Gs241y0fZEpaaKUVq4vJHZpQ4D/MWq085CYmeUvZzBe2X3kPNgGR
                          6iwZwBsr6qlHX67OOo2TCDCNC4XAhBebZMeaaE+rZstevit90M1wBBIBZz8sktab70t1IPDAyCBga1N6TVAVLoi/ElPeu/cWotHWZ2rOpfV1P80=; Domain=.redbus.com; Path=/; Expires=Fri, 09 Aug 2024
                          19:01:45 GMT; Max-Age=7199
[decoded gzip] JSON                                                                                                               [m:auto]
{
    "AppleUniqueIdentifier": null,
    "BUS": {
        "UserType": "NEW"
    },
    "CountryCode": "IND",
    "CreatedOn": "2024-08-05T19:48:15",
    "DefaultCountry": "IND",
    "DisplayName": "Pourhak Sengupta",
    "Dob": "2008-04-26T00:00:00",
    "FBAuthToken": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJodHRwczovL2lkZW50aXR5dG9vbGtpdC5nb29nbGVhcGlzLmNvbS9nb29nbGUuaWRlbnRpdHkuaWRlbnRpdHl0b29sa2l0LnYxLklkZW50aXR5VG9vbGtpdCIsImV4cCI6MTcyMzI
yNjUwNiwiaWF0IjoxNzIzMjIyOTA2LCJpc3MiOiJmaXJlYmFzZS1hZG1pbnNkay1vYmYybUBhcGktcHJvamVjdC0yMzExNzE2ODk2MTUucmVkYnVzLmluLmlhbS5nc2VydmljZWFjY291bnQuY29tIiwic3ViIjoiZmlyZWJhc2UtYWRtaW5zZGstb2JmMm1AYXBpLXByb2p
lY3QtMjMxMTcxNjg5NjE1LnJlZGJ1cy5pbi5pYW0uZ3NlcnZpY2VhY2NvdW50LmNvbSIsInVpZCI6ImNIVnlZbUZyYzJWdVozVndkR0V5TURBeExuUnBiblJwYmtCbmJXRnBiQzVqYjIwPSJ9.EeWNpRnCna_hpMkpAEmr5c724XuSJO_SgHFjkkRVB13Rn3sfmA-H6Q4cYw
zHcwT0KpemInNJazCsvtYpSERJQwyentufqgQB1G2XhgEJbpekaXemxwAxTftrvWLYPqXcWGLKiXoHvxh_z4usJ0u9jEgV4n6IggsTeUtRxzgA5qtl5rL9kbso4E6qTDLXScPoqZxoob3FIVzBOVNGHhvYlAqyLEDz_jJrSoGTdkSdnA5lYR9THg1JNIhIaANVJRnVLPgp2X
nn21IQcTP0MA58Bn1FuwHOnrcOZ0pAb9DVXH9TkEwyidg1AAsWirY7632Z_as_lKxZsTnunuoj8YfjvA",
    "Gender": 1,
    "IsEmailVerified": false,
    "IsMobileVerified": true,
    "IsWalletEnabled": true,
    "LoginRegionData": {
        "RegionName": "Capital Region",
        "RegionalLanguage": "english",
        "isRegionEnabled": false
    },
    "PEmail": "purbaksengupta2001.tintin@gmail.com",
    "PEmailHash": "cHVyYmFrc2VuZ3VwdGEyMDAxLnRpbnRpbkBnbWFpbC5jb20=",
    "PMobile": "9330853731",
    "PasswordExists": false,
    "PhoneCode": 91,
    "RAILS": {
        "UserType": "NEW"
    },
    "RYDE": {
        "UserType": "NEW"
    },
    "SuspensionInfo": null,
    "UserIdHash": "cZKh7vP4xNYbqMQWDhqjD5foK7G2/CbY7bWuzlJaKUo=",
    "Uuid": "ba593fe8-5d51-4ae2-a6a6-38eec8b7935a",
    "WalletBalance": {
        "coreBalance": 0,
        "currency": "INR",
        "debitAmountBreakUp": null,
        "isSoftLogin": false,
        "isWalletEnabled": true,
        "offerBalance": 0,
        "offers": [],
        "totalBalance": 0
    },
    "WalletId": "ahxAylnNMysiC_RvYXYx-w"
}
↓ [6/10]                                                                                                                         [*:8080]
Flow:   e Edit      D Duplicate  r Replay  x Export   d Delete     b Save body  _ Next flow   p Prev flow
Proxy:  ? Help      q Back       E Events  O Options  i Intercept  f Filter     w Save flows  z Clear list  - Layout    ctrl → Switch F Follow new
```

Figure 10: Response of POST request

before, some details are not exact such as my name is showing Pourhak instead of Purbak because we changed that manually before. We can manipulate that again by making a PUT request and eventually we did that after this.

We tried to change my name that was set wrong already to see how the application reacts. We modified the PUT request and replayed it to see what happens. As expected, the modification showed in the application as well(see fig 11). Next we moved further and tampered some more de-



Samsung Galaxy S6 (1440x2560, 640dpi) - 127.0.0.1:6555 - Ge...

8:25

Personal information

Personal details

Name
Purbak Sengupta

Date of birth
26/04/2008

Gender

[✓ Male] Female

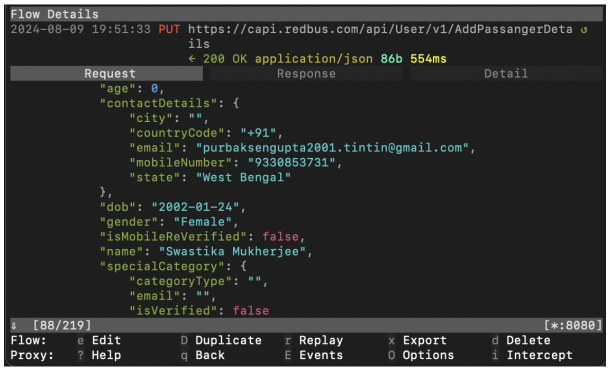
Contact details

State of residence
West Bengal

Required for GST Tax Invoicing

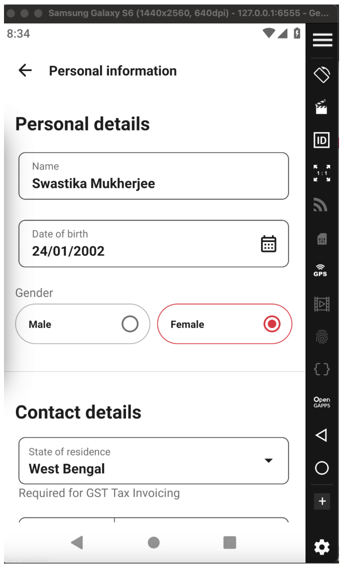
Figure 11: Modifying the name only

tails such as name, date of birth and gender and then replayed the PUT request again.(see fig 12). This was again successful and the changes reflected in the application as shown in fig 13. Now, we got to know the FBAuthToken in the response of the POST request(see fig 10) which is supposed to be acting as a session token which means it should be deleted when user logs out of the account. However, when we logged out, no such DELETE request



```
Flow Details
2024-08-09 19:51:33 PUT https://capi.redbus.com/api/User/v1/AddPassangerDeta ↺
ils
                       ← 200 OK application/json 86b 554ms
          Request                       Response                       Detail
            "age": 0,
            "contactDetails": {
                "city": "",
                "countryCode": "+91",
                "email": "purbaksengupta2001.tintin@gmail.com",
                "mobileNumber": "9330853731",
                "state": "West Bengal"
            },
            "dob": "2002-01-24",
            "gender": "Female",
            "isMobileReVerified": false,
            "name": "Swastika Mukherjee",
            "specialCategory": {
                "categoryType": "",
                "email": "",
                "isVerified": false
↓ [88/219]                                                          [*:8080]
Flow:   e Edit     D Duplicate  r Replay  x Export   d Delete
Proxy:  ? Help     q Back       E Events  O Options  i Intercept
```

Figure 12: PUT request to tamper with the details



Samsung Galaxy S6 (1440x2560, 640dpi) - 127.0.0.1:6555 - Ge...

8:34

Personal information

Personal details

Name
Swastika Mukherjee

Date of birth
24/01/2002

Gender

Male [✓ Female]

Contact details

State of residence
West Bengal

Required for GST Tax Invoicing

Figure 13: Tampered details shown in the application

was there in the traffic. It means the data created with this session token is still accessible through the API. It is possible that the application might be relying on token expiration rather than explicitly invalidating the token on the server. In this case, the token will simply expire after a certain time and cannot be used again. It is also possible that the application has not implemented a secure logout process that invalidates tokens server-side, which would be a security oversight.

C. Authentication

We are going to discuss about the authentication mechanisms the application has in this part. Being a highly important security feature, it is needed to ensure that no one other that the user can get access to the user's account without his/her consent.

Authentication Mechanisms

Typically, there can be three types of authentication mechanisms to authenticate a valid user:

- Something like password or pattern
- Something like cryptographic keys
- Biometric Authentication such as fingerprint

Some application uses two-factor authentication(2FA)[20] which basically is a combination of above mentioned mechanisms, for example a person has to give his/her password first and then he/she has to go through biometric authentication for successful validation so that they can login to the app. This provides an additional layer of security to the system which makes it even more challenging for an adversary to crack.
Unfortunately, Redbus has no authentication mechanisms implemented in the app which is very risky for an user to use this app. It is very vulnerable to physical attack, which means if an adversary somehow gets access to the user's device, he/she can easily open the app and see the user's details, bookings etc. For security reasons, atleast 2FA should have been implemented. The application is easy to use, however it is not secure to use right now from authentication point of view.
It is to be remembered that, a single password or social account login is also not very secure as there might have been several data breaches where passwords might have been leaked. Also, people often use same password for a long period of time or for several application. Now if an adversary gets to know the password for another application, getting access to Redbus will be very easy. Firstly, Redbus should introduce password authentication every time the user wants to login to the system. The password policy should be strong such as it should be of minimum 10-12 characters long with upper and lower case alphabetical combination, there should be numerical and alphabetical combination, atleast one symbol should be there etc. Introducing mandatory symbol in password can not only eliminate straight dictionary attack[21] but also makes it difficult for an attacker to crack. Lastly, there should a timeout or maximum attempts to login, failing which the user gets notified. The user will get to know then if an attack has happened or not, if not then the user can always recover password("forgot password?") to login again. However, our recommendation is this traditional username and password login is still not enough, the application should atleast implement a biometric authentication like fingerprint after the password phase for a two-factor authentication. This way, the user friendly nature of the app can be retained as well as the app would be more secured.

Authentication Flow

1. Account Creation: The application recommends to sign up with and account to save user's details for further use, however skipping that part is possible. It is possible to use the app without creating an account although details will be needed while booking. User gets two options to sign up: with mobile number or with google account. If the users choose mobile number, they have to give an OTP that has been sent to the given mobile number to sign up, and if they choose google account sign-up option, they might have to authenticate themselves in Google through 2FA if it is already set up in their Google account. However, as already discussed in the previous section, after this the users do not need to verify themselves anymore to open the app. After creating/signing into the account, the users can directly use the app for booking. Personal information is used to facilitate bookings, send transaction updates, customize content, conduct surveys, gather user-generated content and run marketing promotions by Redbus[22]. The data enables better services, individual experiences and research. Data retention is based on necessity and legal requirements.

2. Changing Details: We found two scenarios in this case, if the users sign in with their mobile number, they can add/change every details anytime. On the other hand, if the users sign in with Google account, then they can add/change their details anytime except their email id. This is not at all secured again. As there is no authentication mechanisms implemented in the app while opening(as discussed in the previous section), an adversary can easily get into the users' accounts and change the details anytime so easily. Also, if the users make a payment, especially after their first payment, all the payment details are saved by the application which can be easily visible to the adversary. This is also not very secure from a security point of view. We tried to login to the same account from different devices. Google account was chosen for login and the user was not even notified through an email that there is another device who has now access to the account. This was surprising as we consistently changed some personal information from one device and it reflected on the other device as well. We tried this from different devices, however result was the same. Overall, we found the authentication system of Redbus application very poor.

D. Privacy

We have already mentioned before that privacy is essential to be maintained by the application for user's security purpose. In this part, we are going to discuss about the potential vulnerabilities related to the application's privacy policy.

Sensitive Data

The Redbus application indeed deals with many sensitive information such as an user's personal info like name, phone number, email id etc, card payment details, banking and wallet details, booking references and travelers list linked to the account. It is important that these information are handled with great care and from Redbus's privacy policy it it clear that they are well aware of this. They say: *"The information as detailed below is collected for us to be able to provide the services chosen by you and also to fulfill our legal obligations as well as our obligations towards third parties as per our User Agreement."*[23], which clearly states that the information are shared only with contracted third parties by anonymizing the data to provide services to the user and enhance their experience.

Permissions

When opening the application for the first time, users will be asked for location and notification allowance. However, by default the application has only *"Siri and Search"* permission enabled only for iPhone and that too can be turned off by the users if they want.
There were several permissions categorized as *"dangerous"* in the MobSF report. Rest of the permissions were normal and common for an application development. However, we found the number of dangerous permissions list too long, which include:

- android.permission.ACCESS_COARSE_LOCATION: Access coarse location sources, such as the mobile network database, to determine an approximate phone location, where available. Malicious applications can use this to determine approximately where you are.
- android.permission.ACCESS_FINE_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.AUTHENTICATE_ACCOUNTS: Allows an application to use the account authenticator capabilities of the Account Manager, including creating accounts as well as obtaining and setting their passwords.
- 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.GET_ACCOUNTS: Allows access to the list of accounts in the Accounts Service.
- android.permission.POST_NOTIFICATIONS: Allows an app to post notifications.
- android.permission.READ_CALENDAR: Allows an application to read all of the calendar events stored on your phone. Malicious applications can use this to send your calendar events to other people.
- android.permission.READ_CONTACTS: Allows an application to read all of the contact (address) data stored on your phone. Malicious applications can use this to send your data to other people.
- android.permission.READ_EXTERNAL_STORAGE: Allows an application to read from external storage.
- android.permission.READ_PHONE_STATE: Allows the application to access the phone features of the device. An application with this permission can determine the phone number and serial number of this phone, whether a call is active, the number that call is connected to and so on.
- android.permission.READ_SMS: Allows application to read SMS messages stored on your phone or SIM card. Malicious applications may read your confidential messages.
- android.permission.RECEIVE_SMS: Allows application to receive and process SMS messages. Malicious applications may monitor your messages or delete them without showing them to you.
- android.permission.RECORD_AUDIO: Allows application to access the audio record path.
- android.permission.SYSTEM_ALERT_WINDOW: Allows an application to show system-alert windows. Malicious applications can take over the entire screen of the phone.
- android.permission.USE_CREDENTIALS: Allows an application to request authentication tokens.
- android.permission.WRITE_CALENDAR: Allows an application to add or change the events on your calendar, which may send emails to guests. Malicious applications can use this to erase or modify your calendar events or to send emails to guests.
- android.permission.WRITE_EXTERNAL_STORAGE: Allows an application to write to external storage.

Total tally of *"dangerous"* listed permissions is 17 which is quite alarming. The interesting part is, some dangerous permissions are related to authentication mechanisms like "android.permission.AUTHENTICATE_ACCOUNTS", however as already discussed before, after login there is no authentication mechanisms implemented further in the application which is quite contradictory.
The application has coarse and fine location permissions however, exact location is not needed to uniquely identify an user. Only a few data points are needed for that purpose. Even a coarse data set provides little anonymity[24]. Redbus shares data with contracted third parties. They have clearly mentioned with whom they share users' data with. The list consists of : 1. Service Providers and suppliers, 2. Companies In The Same Group, 3. Business Partners and Third-Party Vendors and 4. Disclosure of Information Under Certain Circumstances. They say:

"In the interests of improving personalization and service efficiency, we may, under controlled and secure circumstances, share your Personal Information with our affiliate or associate entities." and also *"We may also share certain filtered Personal Information to our corporate affiliates or business partners who may contact the customers to offer certain products or services."*[22].

So, it is clear that they only share the data to enhance the performance as well as for business related works and only share limited anonymized information that should be shared under specific circumstances. However, the users are not aware of this while downloading and using the application. They are not notified to accept any kind of terms and conditions or to read the privacy policy before using the application. Redbus should implement more clarification on the user side providing them with a crystal-clear picture of the application's privacy before they fill in their personal information into the application.

Trackers

From MobSF report, we already knew the number of trackers and later we used Exodus Privacy to look more into the trackers available in APK. A tracker is a small piece of software that records users' activities on the system. Trackers send information about the activities they have recorded to a tracking server, owned by the company that made the tracker[10]. It basically looks for users' usage patterns. Based on the static analysis conducted by Exodus Privacy and MobSF, the app includes nine such trackers.

- io.branch: Branch is a mobile linking and attribution platform. It provides solutions for deep linking, user attribution, and measurement across devices and channels. The Redbus app uses Branch to enhance user acquisition, engagement, and retention by optimizing its marketing campaigns. Branch usually collects data such as device fingerprint ID, identity ID, hardware ID, brand, model, screen DPI, screen height, and local IP addresses to understand user interactions and target audiences better.

- com.facebook.appevents: Facebook App Events is a tool within the Facebook SDK that allows apps to measure the effectiveness of their ads and understand user engagement. The Redbus app uses Facebook App Events to track user actions, optimize ad targeting, and measure the results of their ad campaigns. This service collects data such as device information, app usage patterns, and events triggered within the app.

- com.facebook.login: Facebook Login is a feature within the Facebook SDK that allows users to log into third-party apps using their Facebook credentials. This integration allows Redbus to access basic profile information such as name, email, and profile picture, which helps streamline the registration process and personalize the user experience. Additionally, Facebook Login enables Redbus to offer social features and integrate with the user's Facebook network, enhancing the app's functionality and user engagement.

- com.facebook.share: Facebook Share is a feature within the Facebook SDK that allows users to share content from third-party apps to their Facebook timeline, pages, groups, or stories. The Redbus app uses Facebook Share to enable users to share their travel experiences, ticket bookings, and other relevant information directly to their Facebook account. This integration helps increase user engagement and visibility by allowing users to effortlessly share content with their Facebook friends and followers. The shared content can include text, images, links, and other media, facilitating a seamless and interactive sharing experience within the Redbus app.

- com.google.android.gms.analytics: Google Analytics for mobile apps is a service provided by Google that allows app developers to measure user interactions and engagement with their apps. The Redbus app uses Google Analytics to collect data on how users interact with the app, such as which features are used most frequently, user demographics, session duration, and other behavioral metrics. Google Analytics ensures that the data is anonymized, protecting user privacy while providing valuable insights to the app developers.

- com.google.firebase.crashlytics: Firebase Crashlytics is a real-time crash reporting tool that helps app developers track, prioritize, and fix stability issues in their apps. The Redbus app uses Firebase Crashlytics to monitor crashes and errors that users encounter while using the app. When a crash occurs, Crashlytics collects and sends detailed diagnostic information, such as stack traces, device metadata, and user actions leading up to the crash, to the developers. This information allows the Redbus development team to identify the root cause of crashes quickly, prioritize them based on impact, and release timely fixes.

- com.google.firebase.analytics.FirebaseAnalytics: Firebase Analytics is an app measurement solution that provides insights on app usage and user engagement. This data helps Redbus understand user behavior, evaluate features and improve user experience. Firebase Analytics also enables personalized marketing and targeted notifications by segmenting users based on behavior. Also, the data is anonymized to protect user privacy while delivering valuable insights into app performance and engagement.

- com.google.android.gms.tagmanager: Google Tag Manager is a tag management system that allows developers to easily update measurement codes and related code fragments collectively known as tags on their mobile applications. The Redbus app uses Google Tag Manager to manage and deploy analytics and marketing tags without having to modify the app code. This helps in tracking user interactions and measuring the effectiveness of marketing campaigns. It also ensures efficient tag management, allowing for quick updates and implementations.

- com.moengage: MoEngage is a customer engagement platform that helps businesses understand and interact with their users across various channels. The Redbus app uses MoEngage to analyze user behavior and deliver personalized communication. This includes sending targeted notifications, in-app messages, and email campaigns. MoEngage collects data such as user interactions, preferences, and usage patterns to provide insights that help in optimizing engagement strategies and improving user retention. The platform ensures that the communication is relevant and timely, enhancing the overall user experience.

IV CONCLUSION

This project report outlined a thorough security analysis of the Redbus v22.9.5 mobile application. We identified a threat model and described the security properties and attack surface. Application apk was decompiled and the obfuscated structure was analyzed. A static and dynamic analysis was conducted through MobSF where network security was analyzed to have TLS 1.3 and TLS 1.2 which are secure. However, MobSF static analysis report suggested a moderate to low level of security assessment due to having large number of high and medium security vulnerabilities as well as trackers. We thoroughly searched for security related keywords as well but it was not clear and difficult to find any security breaches from manual searching and vulnerabilities related to MobSF report. The static analysis suggested that the application has SSL certificate pinning, however our MITM attack attempt was successful which was unexpected. It suggests that the SSL pinning of the application was either not properly implemented/handled or was weak. The MITM attack exposed multiple personal information which is a severe security vulnerability. Hence, the confidentiality of the user is compromised as an adversary can see all the personal details by eavesdropping to the traffic between client and server. The integrity of the user's data is also compromised as it can be tampered by an adversary. Lastly, authenticity of the user is also compromised, as the login credentials are exposed in the MITM attack, so an adversary now have all the details to login to the system. All these are major security vulnerabilities because an adversary can easily look for a specific user's calendar and journey, booking id, wallet details, payment details etc. There is no track of deleting the session token which compromises reliability as there is no surety how the data is being handled. Authentication mechanism of the application is very poor. After creating or login to the account, there is no need to authenticate again as there is no password or biometric authentication implemented. This makes much easier for physical attacks to happen. Anyone who has access to the device other than the user can open the application and see all the details anytime they want. The user does not even get notified of a new login to the same account from another device(if that happens).
The company's privacy policy is well documented and provides a clear picture about how the user's data is being handled and to whom are these data being shared with. However, it is not fully transparent as the user does not get to know these information while downloading or using the application. The user has to go through Redbus's privacy policy website under "Know about redBus" option in the application. Most of the users do not read these long documents themselves. There was a possibility if these documents were shown while starting the application for the first time, users might read. However, this is also not the case as users get to use the application straight after login or even without login/creating account. Eventually, users have the impression of having the ownership of their own data, they only get to know that their data are being shared with contracted third parties by reading those documents.
Overall, Redbus has several major security vulnerabilities and our ultimate assessment of the security level of this application is low. The vulnerable issues need to be fixed as soon as possible with upcoming updates to gain users' trust and reliability as well as from their security and privacy point of view. Currently, using and booking tickets through this application is not secure and it is recommended not to use.

References

[1] https://www.redbus.in/info/aboutus

[2] https://www.techtarget.com/iotagenda/definition/man-in-the-middle-attack-MitM

[3] https://www.strongdm.com/blog/man-in-the-middle-attack-prevention

[4] APKTool

[5] JADX

[6] https://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/

[7] GitHub, Out of memory problems

[8] MobSF, GitHub

[9] https://www.geeksforgeeks.org/difference-between-sha1-and-sha256/

[10] https://brave.com/glossary/tracker/

[11] https://www.ssl.com/blogs/what-is-certificate-pinning/

[12] https://robertheaton.com/2013/07/29/padding-oracle-attack/

[13] "Random vs Secure Random numbers in Java", https://www.geeksforgeeks.org/random-vs-secure-random-numbers-java/

[14] Wireshark

[15] Genymotion

[16] "Resolved Addresses", Wireshark

[17] Qualys SSL Labs

[18] https://www.cloudflare.com/en-gb/learning/ssl/why-use-tls-1.3/

[19] https://github.com/shroudedcode/apk-mitm

[20] "What is two-factor authentication?", Microsoft

[21] "What is a Dictionary Attack?", Kaspersky

[22] Redbus Privacy Policy

[23] "TYPE OF INFORMATION WE COLLECT AND ITS LEGAL BASIS", Redbus Privacy Policy

[24] Montjoye, Yves-Alexandre & Hidalgo, Cesar & Verleysen, Michel & Blondel, Vincent. (2013). Unique in the Crowd: The Privacy Bounds of Human Mobility. Scientific reports. 3. 1376. 10.1038/srep01376.

[25] S Kumar, "What is the MD5 Algorithm?", Geeksforgeeks