

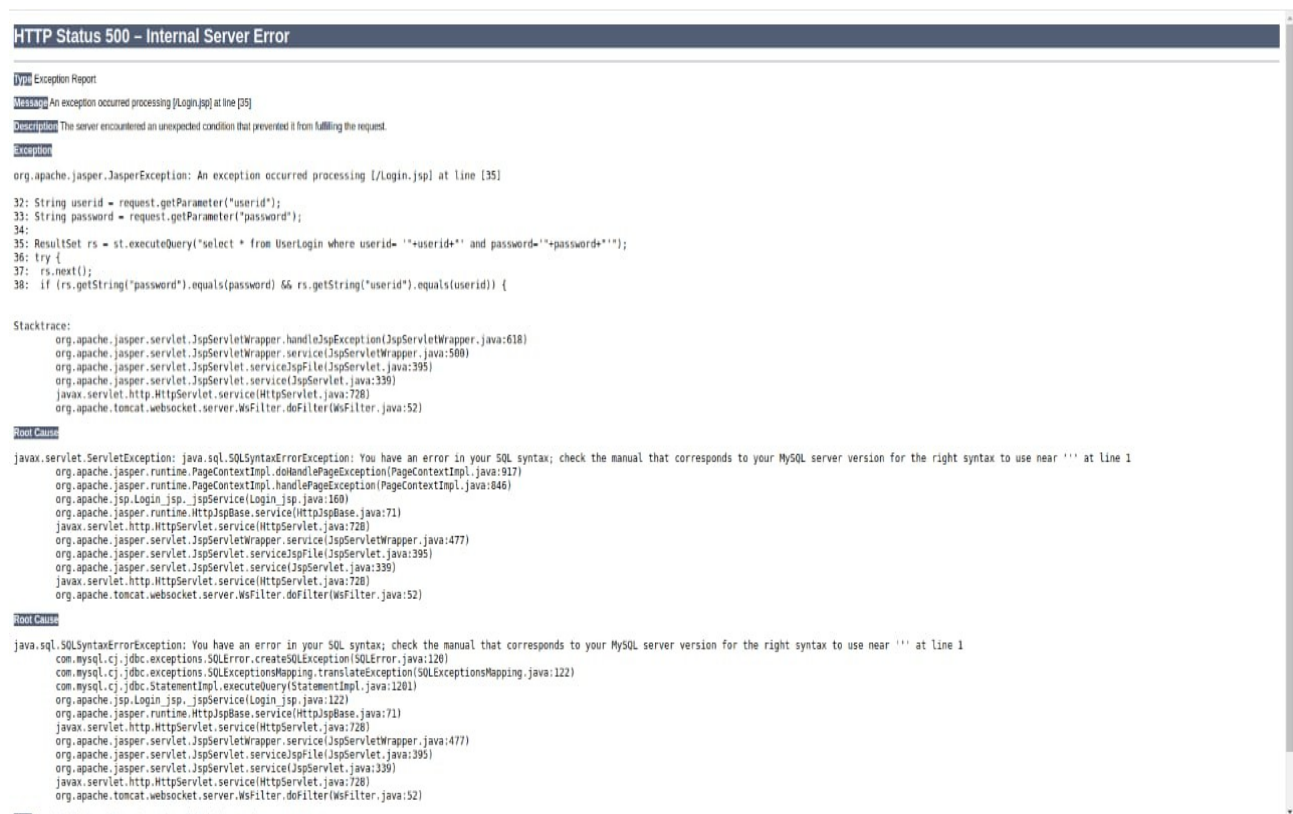
SUBJECT

Report after conducted Manual Inspection, Thread Modelling ,Source code Review in “**TRAIN BOOKING**” application.

VULNERABILITIES

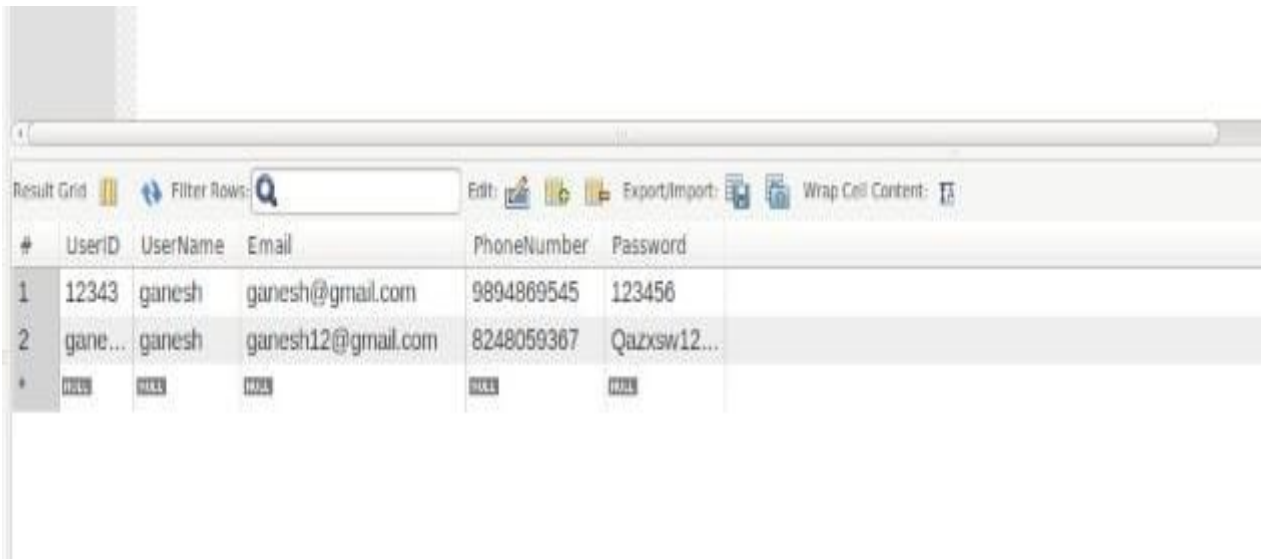
We realized that some of the vulnerabilities in the application that leads to attack the sensitive data.

- At first ,we found some Unhandled exceptions in their code that leads to their code are visible to the user. It may be the easier path to attack and steal information.Paticularly it shows the table name and column names in the database.



- In the Login page ,there are so many possibilities for Brute Force attack it causes the attacker easily misuse and does identity theft. It is a simple yet reliable tactic for gaining unauthorized access to individual accounts and organizations' systems and networks.

- We came to know from discussion that the data stored in the database are not properly encrypt ,they stored the sensitive data of the user in the native format. It may leads to Cyberattack.



#	UserID	UserName	Email	PhoneNumber	Password
1	12343	ganesh	ganesh@gmail.com	9894869545	123456
2	gane...	ganesh	ganesh12@gmail.com	8248059367	Qazxsw12...
*	NULL	NULL	NULL	NULL	NULL

- This application lack of Login Monitoring , When it comes to exploitation of cybersecurity, insufficient logging and monitoring have been the major cause of Third persons Involvement.
- In the application,when the third party tries to access the login page they fails to warn the user via alerts. It will diminish reliability of the respective application.
- On the other hand, there is a one more possible way to the attacker to access the credentials of the user by the Developers, because they used few of the usual terms acting as admin, user etc.
- The following code lets for huge chances for SQL Injection attacks. It will alter the SQL queries by injecting malicious code .

```
Activities Google Chrome Tue 17:33
github.com/hadanDevs/TrainBooking/blob/main/Login.jsp
31 <%
32 String userid = request.getParameter("userid");
33 String password = request.getParameter("password");
34
35 ResultSet rs = st.executeQuery("select * from UserLogin where userid= '"+userid+"' and password='"+password+"'");
36 try {
37     rs.next();
38     if (rs.getString("password").equals(password) && rs.getString("userid").equals(userid)) {
39         %>
40         <div class="center">
41             <span style="font-family: wingdings; font-size: 200%;">&#9989;</span>
42         </div>
43         <div class="center">
44             <h1> Hii... Successfully SignIn</h1>
45         </div>
46     <%
47
48     } else {
49         out.println("Invalid password or username.");
50     }
51 } catch (Exception e) {
52     e.printStackTrace();
53 }
```

RECOMMENDATIONS

- From the overview of the application, if java exceptions are not handled, programs may crash or request may fail. We suggest them to handle those unhandled exceptions.
- They are failed to encrypt the user data stored in database , we recommend them to use hashing which is irreversible rather than encryption to defend against Cyber-attacks including Malware and Ransomware. When we using encryption we can easily decrypt the user data.
- We also prefer for strong passwords should be enforced. Password hashes should be stored encrypted and salted. User account should be locked after three or four attempts to resist against unwanted access.
- We also suggest to use Parameterized Query to defend against SQL Injection attacks.

CONCLUSION

Security Testing attempts should ensure the confidentiality, integrity, and availability of the application. Our intention is to identify the risks in the application and measure its potential vulnerabilities so that the threats can be encountered . Till our discussison, they are assured to rectify the specified vulnerabilities.