Web Application Penetration Testing Report

Report For :

Organization Name : Gujarat Informatics Ltd.

Report Date : August 31st 2024

Report By :

Organization Name : Xiarch Solutions Pvt Ltd

Report Date : August 31st 2024

Address : 403-404, Tower A , Spaze Edge, Sohna Road, Gurugram, Haryana 122018

Tel : +91-9667916333

Email : info@xiarch.com

Web : www.xiarch.com

Disclaimer

All the information contained in this document is confidential to said company, disclosure and use of any information contained in this document by photographic, electronic or any other means, in whole or part, for any reason other than security enhancement is strictly prohibited without written consent of auditee organization.

Whilst all due care and diligence has been taken in the preparation of this document it is not impossible that document of this nature may contain errors or omissions because of a misunderstanding of Clients requirements. Any recommendations are made in good faith as guidelines to assist the client in evaluation and must not be construed as warranties of any kind. Findings in this report are based on various tests conducted using manual techniques and third-party tools and Xiarch Solutions Pvt Ltd has put its best efforts to eliminate all the false positives reported by these tools.

Xiarch Solutions Pvt Ltd shall assume no liability for any changes, omissions, or errors in this document. Xiarch Solutions Pvt Ltd shall not be liable for any damages financial or otherwise arising out of use/misuse of this report by any general member of public

.

1. Document Control

This document serves as a comprehensive report of the findings and recommendations resulting from the web application security assessment performed on Gujarat Panchayat Service Selection Board(Gujarat Informatics Ltd.) by Xiarch Solutions Private Limited. This report is intended for Management, and is to be used as a guide for improving the security posture of the web application.

1.1 Document Version Control

Version	Report	Person	Action	Date
1.0	Level 01	Manish Gupta	Audit	31/08/2024
1.1	Level 01	Nitin Panwar	Review	31/08/2024
1.2	Level 01	Ronak Bal	Final Approval	31/08/2024

1.2 Document Distribution List

Name	Designation	Organization
Manish Gupta	Associate InfoSec Consultant	Xiarch Solutions Private Limited
Nitin Panwar	InfoSec Consultant	Xiarch Solutions Private Limited
Ronak Bal	Senior InfoSec Consultant	Xiarch Solutions Private Limited

2. Introduction

2.1 Summary

This report outlines the results of a comprehensive web application security assessment conducted for the Gujarat Panchayat Service Selection Board web application. The purpose of this assessment was to identify any potential security vulnerabilities and to provide recommendations for their mitigation.

The assessment was conducted using a combination of manual and automated testing techniques, including vulnerability scanning, penetration testing, and code review. The scope of the assessment covered the application's front-end and back-end components, as well as any supporting infrastructure.

The assessment was conducted in accordance with industry-standard security best practices and guidelines, including the Open Web Application Security Project (OWASP) Top 10 list of web application security risks.

The results of the assessment identified a number of security vulnerabilities, which have been categorized according to their severity and impact on the application's security posture. Recommendations for mitigating these vulnerabilities have been provided, along with a roadmap for addressing any remaining security issues.

The findings and recommendations contained in this report are intended to assist the development team in improving the overall security of the web application, and to help ensure that it meets the necessary security standards and compliance requirements.

2.2 Assessment Objective:

- Identify and assess security flaws in mobile application according to industry principal security standards like OWASP Web Security Project Top 10 and SANS 25 etc.
- Provide recommendations for mitigation of risk(s) emerged during the identified vulnerabilities.

3. Scope

3.1 Scope Details

The security assessment was carried out in the pre-production environment and it included the following scope:

Application Name	**Panchayat Web Application**
URL	https://panchayat.gipl.in/ https://panchayat.gipl.in/workflow
User IDs and Access	Username- DDO_AMR, Panchayat , Password: *********
Testing Environment Details	Testing Environment
Application Description	**Panchayat** Web application

For Employee	
Modules	**Sub module**
Login	N/A
Dashboard	N/A
Edit Details	N/A
Logout	N/A

Out of Scope	
Module	**Sub module**
N/A	N/A

Assumptions:

- No changes were made during assessment by project team, all artefacts/documents shared with us are latest as on shared date and are unaltered.
- This vulnerability assessment exercise did not include the testing of vulnerabilities leading to Denial of service (DoS) and similar attacks.

- Appropriate backup and rollback plan are made prior to implementing the recommendation on the system.
- Vulnerabilities identified were as on the data scan conducted and as per the scan policies (non-intrusive) & plugins selected. There may be vulnerabilities which may exist & not assessed since their exploits may lead to system downtime. Also, the vulnerabilities identified after the scan date may also not form part of this report.

3.2 Post Assessment Clean-up

Any test accounts, which were created for the purpose of this assessment, should be disabled or removed, as appropriate, together with any associated content.

4. Terms , Clarity & Legends

This section describes the format in which the identified vulnerabilities are reported in the later section of the report. "Vulnerability Table" shown below is used to provide the details of the vulnerability, its impact and the recommendations.

4.1 Vulnerability Table

Observation ID & Title :			
CVSS Risk Rating :		**Status :**	
Observation Details :			
Risk/Impact :			
Recommendations :			
Affected URL & Parameter :			
CVSS Vector :			

Title of the Vulnerability	-	A short title that describes the vulnerability
Risk Level	-	It describes the risk level. The title bar of each vulnerability table is colour coded for quick identifications of the severity level of the vulnerabilities
Description	-	It provides a brief description of the vulnerability.
Impact	-	Describes the probable impact if the vulnerability is successfully exploited.
Recommendations	-	Provide the recommendations to fix the vulnerability.
Affects	-	Provide the information where vulnerability is present.
Status	-	Provides the information whether the vulnerability is closed or not.

4.2 Findings Ranking System

In order to prioritize the assessment results, each finding was categorized based on severity classifications. Final analysis of the risk or impact to the application will require an internal evaluation. Xiarch Labs has developed classifications using the severity nomenclature for ranking the issues identified within the various severity categories.

4.2.1 Severity Categories

Based on Xiarch Lab analysis of the particular finding and assets affected, a finding will fall into one of the following severity level categories:

Critical Vulnerabilities require an immediate response through mitigating controls, direct remediation or a combination thereof. Exploitation of critical severity vulnerabilities results in privileged access to the target system, application or sensitive data and enables further access to other hosts or data stores within the environment. In general, a critical severity ranking is warranted when the issue has a direct impact on regulatory or compliance controls imposed on the environment, accesses personally identifiable information (PII) or financial data or could cause significant reputational or financial harm.

High Findings with a high severity ranking require immediate evaluation and subsequent resolution. Exploitation of high severity vulnerabilities leads directly to an attacker gaining privileged, administrative-level access to the system, application or sensitive data. However, it does not enable further access to other hosts or data stores within the environment. If left unmitigated, high severity vulnerabilities can pose an elevated threat that could affect business continuity or cause significant financial loss.

Medium A finding with a medium severity ranking requires review and resolution within a short time. From a technical perspective, vulnerabilities that warrant a medium severity ranking can lead directly to an attacker gaining non-privileged or user-level access to the system, application or sensitive data. Findings that can cause a denial-of-service (DoS) condition on the host, service or application are also classified as medium risk. Alternately, the vulnerability may provide a way for attackers to gain elevated levels of privilege. From a less technical perspective, observations with this ranking are significant, but they do not pose as much of a threat as high or critical severity exposures.

Low Low severity findings should be evaluated for review and resolution once the remediation efforts for critical, high and medium severity issues are complete. From a technical perspective, vulnerabilities that warrant a low severity ranking may leak information to unauthorized or anonymous users used to launch a more targeted attack against the environment.

Informational An informational finding presents no direct threat to the confidentiality, integrity or availability of the data or systems supporting the environment. These issues pose an inherently low threat to the organization and any proposed resolution should be considered as an addition to the information security procedures already in place.

5. Assessment Methodology

A penetration testing methodology for web apps typically includes the following steps:

Probe: Involves gathering information about the target system, such as the target's IP address, server type, operating system, and other details that can help identify vulnerabilities.

Enumeration: In this phase, the tester tries to identify any open ports, services or applications that may be running on the target system. This can be done using various tools such as port, network or web application scanners.

Vulnerability Scanning: Once the tester has identified the services running on the target system, they can use vulnerability scanners to identify any known vulnerabilities or weaknesses that can be exploited.

Exploitation: After identifying vulnerabilities, the tester attempts to exploit them to gain unauthorized access to the target system or its data. This can be done using various tools and techniques such as SQL injection, cross-site scripting or brute force attacks.

After Exploitation: After gaining access to the system, the tester attempts to maintain access and escalate privileges to gain more control over the target system.

Reporting: Finally, the tester documents his findings and provides the client with recommendations for remediation. This report usually contains a detailed description of the vulnerabilities found, their severity, and recommended steps to mitigate them.

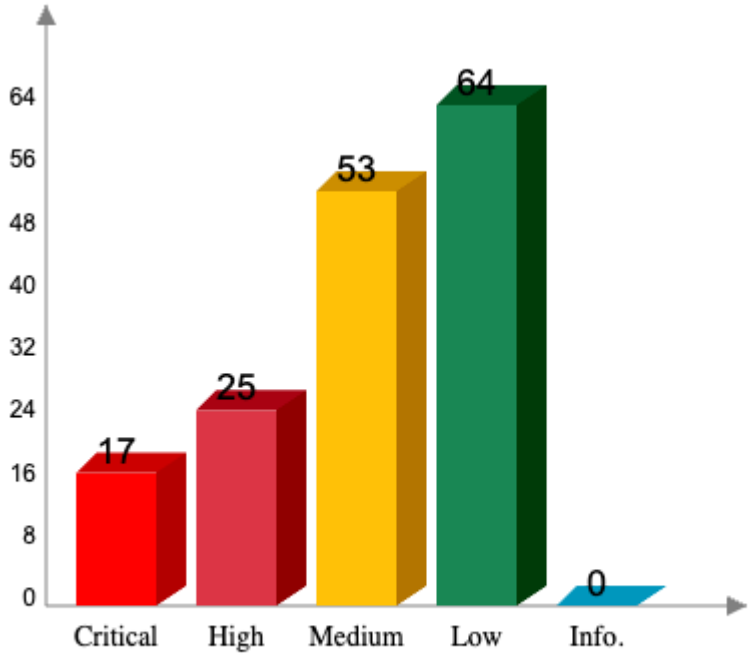
It is important to note that this methodology may vary depending on the specific target being tested and the objectives of the test.

6. Executive Summary

Xiarch conducted a comprehensive security assessment of **Gujarat Informatics Ltd.** in order to determine existing vulnerabilities and establish the current level of security risk associated with the environment and the technologies in use. This assessment harnessed penetration testing and vulnerability assessment techniques to provide **Gujarat Informatics Ltd.** management with an understanding of the risks and security posture of their corporate environment.

6.1 Application Health

Severity Stats



Bar Chart - Severity Stats

7. Assessment Findings Overview

The table below provides a summary of the assessment findings categorized by group and ranked by severity. The table provides an overview of all of the findings from the assessment and allows the remediation team to focus efforts on the areas of highest severity as determined by Xiarch Labs. Click the individual link below to go directly to that finding.

S. No.	Vulnerability Title	Severity	Status
1	Improper Access Control	Critical	Open
2	Default credentials	Critical	Not Applicable
3	Privilege Escalation	Critical	Not Applicable
4	NoSQL injection	Critical	Not Applicable
5	Command Injection	Critical	Not Applicable
6	XML External Entity Injection (XXE)	Critical	Not Applicable
7	Apache Struts RCE	Critical	Not Applicable
8	Apache Spark RCE	Critical	Not Applicable
9	PHPMyadmin RCE	Critical	Not Applicable
10	Shellshock bash RCE	Critical	Not Applicable
11	Pickle Code Execution	Critical	Not Applicable
12	Log4j RCE	Critical	Not Applicable
13	Blind SQL Injection	Critical	Not Applicable
14	In-Band SQL Injection	Critical	Not Applicable
15	Error Based SQL Injection	Critical	Not Applicable
16	UNION Based SQL Injection	Critical	Not Applicable
17	Time-Based SQL Injection	Critical	Not Applicable
18	Insecure Direct Object References	High	Not Applicable
19	Reflected Cross Site Scripting	High	Not Applicable
20	Stored Cross Site Scripting	High	Not Applicable
21	LDAP Injection	High	Not Applicable
22	XPath Injection	High	Not Applicable
23	Local File Inclusion	High	Not Applicable
24	Remote File Inclusion	High	Not Applicable
25	Sensitive information sent via unencrypted channels	High	Not Applicable
26	DOM based Cross Site Scripting	High	Not Applicable

27	Session Replay/Token Reuse	High	Not Applicable
28	No Rate Limit - Account Bruteforce	High	Open
29	Broken Authentication	High	Open
30	Application-Level Denial-of-Service (DoS)	High	Not Applicable
31	Server-Side Request Forgery (SSRF)	High	Not Applicable
32	Second Factor Authentication (2FA) Bypass	High	Not Applicable
33	Hardcoded Password - Non-Privileged User	High	Not Applicable
34	Token Leakage via Host Header Poisoning	High	Not Applicable
35	OAuth Misconfiguration - Account Takeover	High	Not Applicable
36	Cryptographic Flaw	High	Not Applicable
37	Authentication Bypass	High	Not Applicable
38	Information Disclosure	High	Not Applicable
39	Unencrypted Communication	High	Not Applicable
40	F5 Big IP Exploitation	High	Not Applicable
41	Play Session Injection	High	Not Applicable
42	JS Prototype Pollution	High	Not Applicable
43	Review Old, Backup and Unreferenced Files for Sensitive Information	Medium	Not Applicable
44	Bypassing authentication schema	Medium	Not Applicable
45	Improper Cache Control	Medium	Open
46	Weak password change or reset functionalities	Medium	Not Applicable
47	Directory traversal/file include	Medium	Not Applicable
48	Bypassing Session Management Schema	Medium	Not Applicable
49	Session Fixation	Medium	Not Applicable
50	Cross Site Request Forgery	Medium	Not Applicable
51	HTTP Parameter pollution	Medium	Not Applicable
52	ORM Injection	Medium	Not Applicable
53	XML Injection	Medium	Not Applicable
54	SSI Injection	Medium	Not Applicable
55	Buffer overflow	Medium	Not Applicable
56	HTTP Splitting/Smuggling	Medium	Not Applicable
57	Improper Error handling	Medium	Open
58	Padding Oracle	Medium	Not Applicable
59	Business Logic Data Validation	Medium	Not Applicable

60	Number of Times a Function Can be Used Limits	Medium	Not Applicable
61	Upload of Malicious Files	Medium	Not Applicable
62	Vulnerable JavaScript Library	Medium	Open
63	WebSockets hijacking	Medium	Not Applicable
64	Local Storage/Session Storage	Medium	Not Applicable
65	Race Condition	Medium	Not Applicable
66	OAuth Misconfiguration	Medium	Not Applicable
67	Path Traversal	Medium	Not Applicable
68	Weak 2FA Implementation	Medium	Not Applicable
69	Token is Not Invalidated After New Token is Requested	Medium	Not Applicable
70	Failure to Invalidate Session on Logout	Medium	Not Applicable
71	Missing Security Header	Medium	Open
72	Mail Server Misconfiguration	Medium	Not Applicable
73	Server-Side Credentials Storage - Plaintext	Medium	Not Applicable
74	SQL Possible	Medium	Open
75	EXIF Geolocation Data Not Stripped From Uploaded Images	Medium	Not Applicable
76	Failure to Invalidate Session - Password Change	Medium	Not Applicable
77	Web Application Firewall (WAF) Bypass	Medium	Not Applicable
78	HTTP Response Manipulation	Medium	Not Applicable
79	TLS Fallback SCSV Support	Medium	Open
80	Broken Access Control	Medium	Not Applicable
81	Information Disclosure through XMLRPC	Medium	Not Applicable
82	Nginx Rate Filtering Shaping Overflow	Medium	Not Applicable
83	JWT Token Exploit Possible	Medium	Not Applicable
84	GraphQL Injection	Medium	Not Applicable
85	PHP Object Injection	Medium	Not Applicable
86	PHP Include And Post Exploitation	Medium	Not Applicable
87	JSON Web Encryption	Medium	Not Applicable
88	Length Extension Attack	Medium	Not Applicable
89	OAuth2: State Fixation	Medium	Not Applicable
90	SAML: Comment Injection	Medium	Not Applicable
91	Development Configuration File Leakage	Medium	Not Applicable

92	Null Byte Injection	Medium	Not Applicable
93	XQUERY Injection	Medium	Not Applicable
94	Origin Bypass	Medium	Not Applicable
95	Sensitive Path Disclosure	Medium	Not Applicable
96	Banner Grabbing	Low	Open
97	Lucky13	Low	Open
98	No Session Timeout	Low	Open
99	Fingerprint Web Application Framework	Low	Not Applicable
100	Fingerprint Web Application	Low	Not Applicable
101	File Extensions Handling for Sensitive Information	Low	Not Applicable
102	Dangerous Method Enabled	Low	Open
103	HTTP Strict Transport Security	Low	Open
104	RIA cross domain policy	Low	Not Applicable
105	Account Enumeration and Guessable User Account	Low	Not Applicable
106	Weak or unenforced username policy	Low	Not Applicable
107	Credentials Transported over an Encrypted Channel	Low	Not Applicable
108	Weak lock out mechanism	Low	Open
109	AutoComplete Enabled	Low	Open
110	Weak password policy	Low	Open
111	Weak security question/answer	Low	Not Applicable
112	Weaker authentication in alternative channel	Low	Not Applicable
113	Missing secure Flag Not Set	Low	Open
114	Missing HTTP only Flag Set	Low	Not Applicable
115	Simultaneous Login Possible	Low	Not Applicable
116	Session puzzling	Low	Not Applicable
117	Improper Session Expiry	Low	Open
118	Cross-Site Scripting	Low	Not Applicable
119	IMAP/SMTP Injection	Low	Not Applicable
120	Insufficient Transport Layer Protection	Low	Not Applicable
121	Ability to Forge Requests	Low	Not Applicable
122	HTML Injection	Low	Not Applicable
123	Client Side URL Redirect	Low	Not Applicable
124	CSS Injection	Low	Not Applicable

125	IFRAME Injection	Low	Not Applicable
126	Cross Origin Resource Sharing	Low	Not Applicable
127	Cross Site Flashing	Low	Not Applicable
128	Clickjacking	Low	Open
129	Cache Poisoning	Low	Not Applicable
130	TLS 1.0 And TLS 1.1 Enabled	Low	Open
131	Sweet32 Attack	Low	Open
132	BREACH Attack	Low	Not Applicable
133	Old Cipher Supported	Low	Not Applicable
134	Improper Input Validation	Low	Open
135	Directory Listing Enabled	Low	Not Applicable
136	Captcha Bypass	Low	Not Applicable
137	No Token/Cookie Validation	Low	Not Applicable
138	Sensitive Token in URL	Low	Not Applicable
139	Token Leakage via Referer	Low	Not Applicable
140	Content Spoofing	Low	Not Applicable
141	Exposed Admin Portal	Low	Not Applicable
142	No Captcha Implementation	Low	Not Applicable
143	Lack of Password Confirmation	Low	Not Applicable
144	Same-Site Scripting	Low	Not Applicable
145	No Secure Integrity Check - Executable Download	Low	Not Applicable
146	Server-Side Request Forgery (SSRF)	Low	Not Applicable
147	Weak Password Reset Implementation	Low	Not Applicable
148	Weak Registration Implementation	Low	Not Applicable
149	Misconfigured DNS - Zone Transfer	Low	Not Applicable
150	Temp Mail Allowed	Low	Not Applicable
151	LOGJAM	Low	Not Applicable
152	POODLE Attack	Low	Not Applicable
153	Source Code Disclosure	Low	Not Applicable
154	HTTPOXY Attack	Low	Not Applicable
155	HEARTBLEED Attack	Low	Not Applicable
156	DNS Rebinding	Low	Not Applicable
157	SOAP Injection	Low	Not Applicable

158	No CSRF/XSRF Tokens	Low	Not Applicable
159	Password Reuse	Low	Not Applicable

8. Assessment Findings Details

Observation ID & Title	8.1 Improper Access Control		
CVSS Risk Rating	Critical	Status	Open
Observation Details			
It is been observed that the account can be open directly without providing the credentials in the website. As per observation, user logged in first in the account and logout the account. Now attacker just opened the account with the dashboard URL of a account which comes after logging in the account.			
Risk			
It is possible to takeover the account without providing the credentials. This can be used to access the account number of times and attacker can change or edit the details of the user.			
Recommendations			
The Following are the recommendations that should be implemented. 1. Continuous Inspection and Testing Access Control 2. Limiting CORS Usage 3. Proper authentication should be done in the application.			
Affected URL & Parameter			
Throughout the Application			
CVSS Vector			
9.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L			

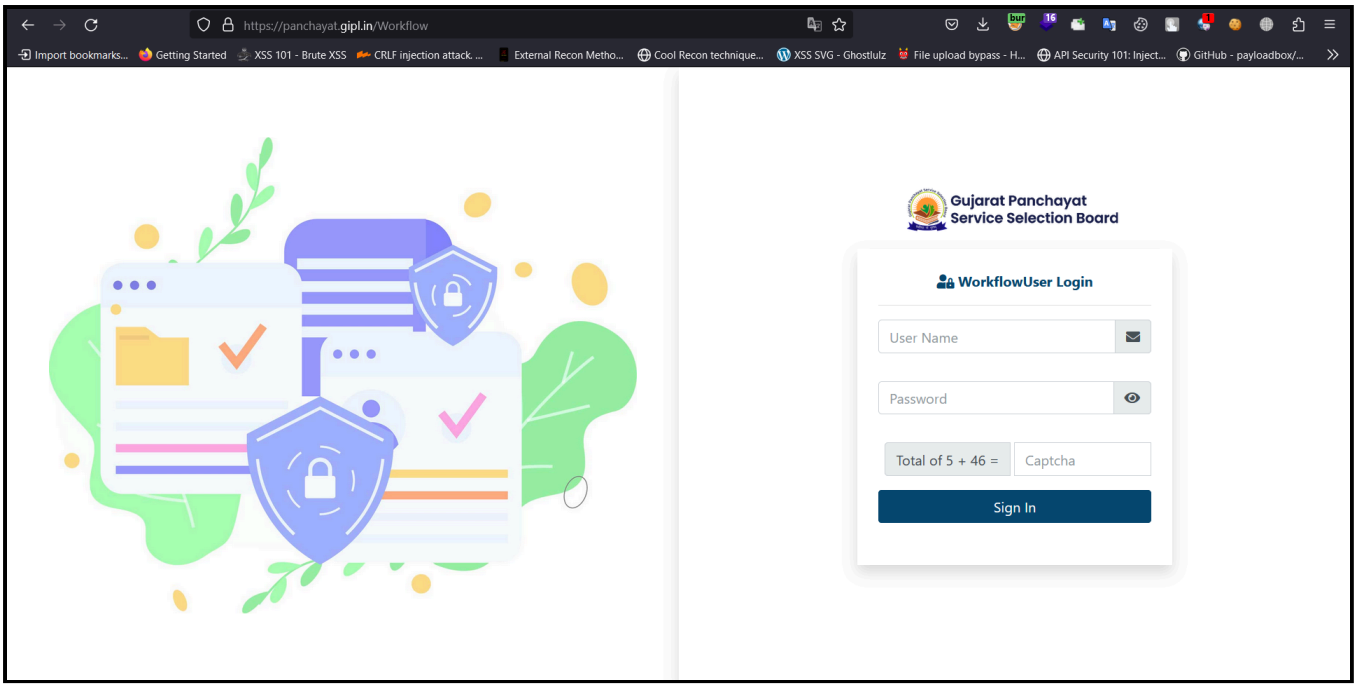


Figure 1 - Go to the mentioned URL

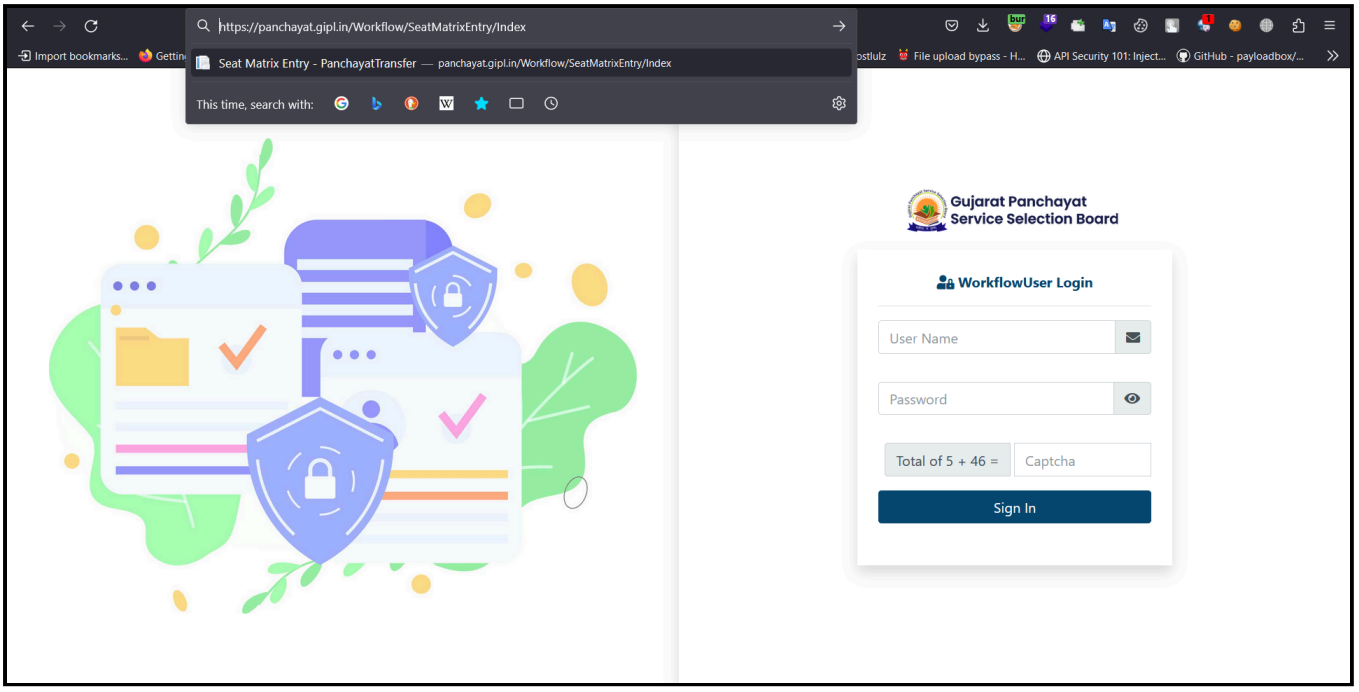


Figure 2 - Paste the URL that comes after Logging in the account.

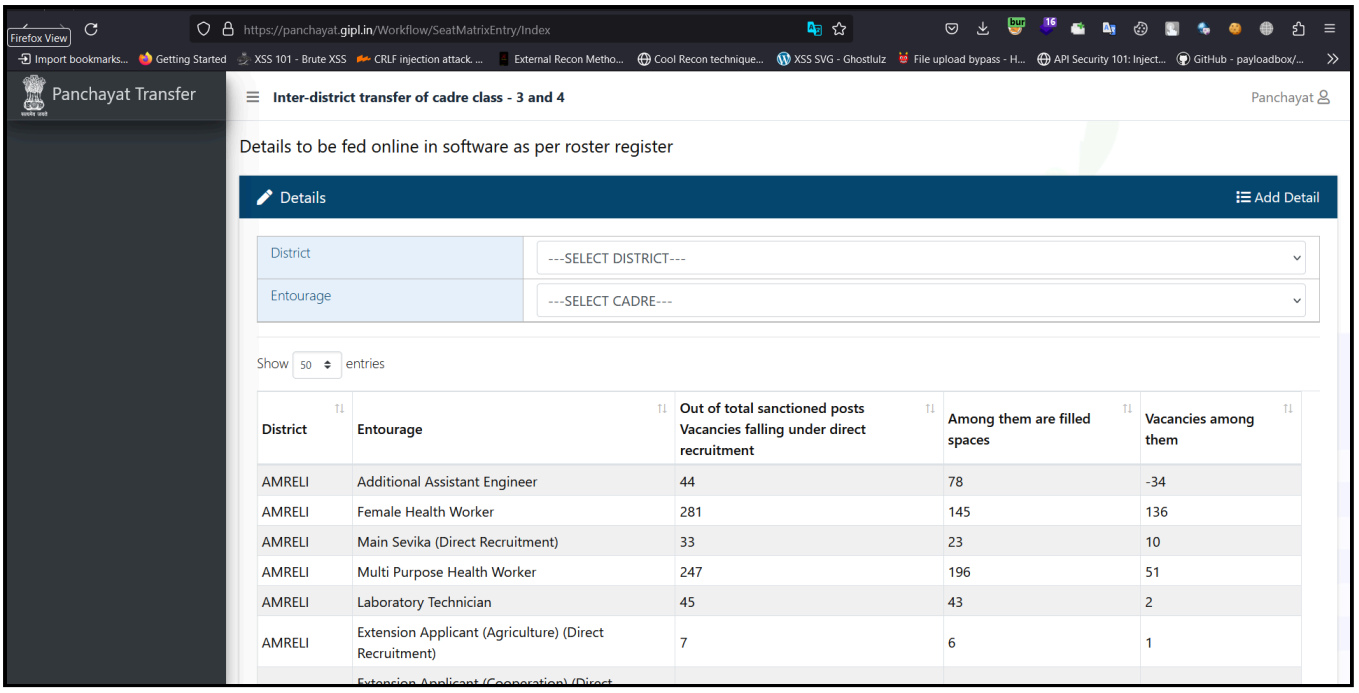


Figure 3 - As you can see that the attacker get the access of the user.

Observation ID & Title	8.2 Default credentials		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

During the testing of the web application, the team was easily able to login to the portal using default credentials. Login through default credentials can be found in many applications. The case might be that the IDs might have been added for testing purposes or an initial setup might've been installed which was later forgot about and later been added to the production server. Usage of common usernames and password leads to a successful brute-force attack. The most common used username and password combinations include "admin: admin", "admin: password", "admin: 12345", etc.

Risk

An attacker might be able to login to the website portal using default credentials, or by using a technique known as brute force attack where an attacker uses automated tools in order to try different login credentials on the login page until he gets successfully logged in. The attacker might be able to view, modify, or even delete critical data if logged in as administrator.

Recommendations

The Following are the recommendations that should be implemented.

1. Implement a strong password policy consisting of a combination of alphanumeric and special characters and a minimum length of 8 characters.
2. Use an anti-brute-force mechanism such as CAPTCHA.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.8 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.3 Privilege Escalation		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

During our testing phase, it came to our notice that the application is vulnerable to privilege escalation which can be very dangerous in any given application. User can specify their own permissions according to their need and there is no need of the administrator to assign the permission for the users. Basically, it's a kind of dangerous attack in order to gain privileged access into a system or in an application. Attacker can design the control flow of the permissions according to their need.

Risk

Using this vulnerability, an attacker can make the site administrator lose access to the server, leak sensitive information causing a data breach or make changes to the data causing data manipulation.

Recommendations

The Following are the recommendations that should be implemented.

Keep critical information on server side.

Encrypt the data sent over the application.

Implicate an MFA rule to change each permission or data with that policy.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.4 NoSQL injection		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

As the team was conducting the testing of the application, we found that the application is vulnerable to NoSQL injection. NoSQL injection is a security weakness in a web application that uses a NoSQL database. NoSQL (Not Only SQL) refers to database systems that use more flexible data formats and do not support Structured Query Language (SQL). They typically store and manage data as key-value pairs, documents, or data graphs.

A NoSQL injection, similar to that of a SQL injection, can allow attackers to bypass authentication, exfiltrate sensitive data, tamper with data on the database, or even compromise the database and the underlying server. Most NoSQL injection vulnerabilities occur because developers accept and process user inputs without properly sanitizing them.

Risk

A NoSQL injection vulnerability, can allow attackers to bypass authentication, exfiltrate sensitive data, tamper with data on the database, or even compromise the database and the underlying server.

Recommendations

The Following are the recommendations that should be implemented.

- Avoid Accepting Raw User Input in Application Code.
- Use the latest version of NoSQL database.
- Use prepared statements (with parameterized queries).
- Use stored procedures.
- Apply input validation and sanitization.
- Apply user-supplied input escaping mechanism.
- Enforcing least privilege policy.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.3 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.5 Command Injection		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

Command injection is an injection technique by which a set of arbitrary commands on the host operating system through a vulnerable application to attack the system. This type of attack is possible when a web application passes unsafe user-supplied data to a system shell. This includes forms, cookies, HTTP headers etc. In this attack, the operating system commands sent by the attacker is executed with the execution privileges of the vulnerable application. There are servers having a vulnerability that can cause Bash code injection vulnerability. This vulnerability can allow an attacker to execute malicious code using a custom environment. This injection vulnerability was found to be in the given web application as we were testing it.

Risk

This vulnerability would lead to the attacker being able to execute commands on the underlying operating system, and gain access to sensitive files and folders being stored on the server side.

Recommendations

The Following are the recommendations that should be implemented.

Avoid system calls and user input to prevent threat actors from inserting characters into the OS command.

Set up input validation to prevent attacks like XSS and SQL Injection.

Create a white list of possible inputs, to ensure the system accepts only pre-approved inputs.

Use execFile() securely—prevent users from gaining control over the name of the program.

You should also map user input to command arguments in a way that ensures user input does not pass as-is into program execution.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.6 XML External Entity Injection (XXE)		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

Upon testing the web application, it was found to be vulnerable to XML External Entity Injection (XXE). XML External Entity attack is an attack against an application that sends an XML input. This attack is countered when a XML input contains a reference to an external entity. And the entity is processed by a weakly configured XML parser. If a server contains weakly configured XML parser, there is a possibility for XML External Entity attack. The XML External Entity attack leads to the disclosure of confidential data, denial of service, server side request forgery and port scanning.

Risk

Depending on the back-end database configuration, its privilege setup and the operating system, an attacker can mount one or more types of attacks such as server side request forgery, denial of service, port forwarding, etc. The attacker can also read, update and delete arbitrary data/tables from the database, and even execute commands on the underlying operating system which might lead to leaking of sensitive information.

Recommendations

It is highly recommended that:-

Upgrading the framework to the latest version.

Ensuring that the inputs are properly validated.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.3 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.7 Apache Struts RCE		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

While conducting the penetration testing of the web application, it came to our notice that there was a Apache Struts vulnerability in the server. Apache Struts is a sophisticated, extensible platform for developing enterprise-grade Java web applications. There is a remote code execution vulnerability in Apache Struts S2-032. This vulnerability exists when Dynamic Method Invocation (DMI) is enabled as expressions passed to the server side are not adequately validated using the method: prefix.

Risk

Remote Code Execution (RCE) vulnerabilities like this can have dire consequences, especially in this case, when it may be possible for an unauthenticated attacker to exploit it. Successfully exploiting a RCE vulnerability could allow the attacker to run arbitrary programs, retrieve source code, or exfiltrate data from the application's database.

Recommendations

The Following are the recommendations that should be implemented.

If Direct Method Invocation (DMI) is enabled, disable it.

Update the Apache Struts server to the latest version to eliminate this vulnerability.

Affected URL & Parameter

N/A

CVSS Vector

9.3 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.8 Apache Spark RCE		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

Apache Spark is a popular open-source distributed computing platform that is often used for big data processing and machine learning. A remote code execution (RCE) vulnerability in Apache Spark could allow an attacker to remotely execute arbitrary code on a server running Spark. This could potentially allow the attacker to gain access to sensitive data or compromise the server.

Risk

The impact of an RCE vulnerability in Apache Spark could be significant, depending on the nature of the code that is executed by the attacker. In the worst case, it could allow the attacker to gain complete control over the server and access sensitive data. This could lead to data breaches, loss of sensitive information, and other security incidents.

Recommendations

To mitigate this type of vulnerability, it is important to keep Apache Spark up to date with the latest security patches.

Implement appropriate network security controls, such as a firewall, to prevent unauthorized access to the server.

Affected URL & Parameter

N/A

CVSS Vector

9.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.9 PHPMyadmin RCE		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

PhpMyAdmin is a free and open-source administration tool for MySQL and MariaDB, providing us with a user-friendly interface. This application has become one of the most popular MySQL administration tools in many hosting services, which provides the functionality to perform Create, Read, Update, and Delete (CRUD) operations on the MySQL database. If the Installed version of PhpMyAdmin is 4.8.1, this could lead to a Remote Code Execution vulnerability

Risk

The impact of a remote code execution attack can vary between simply gaining access to an application and entirely taking it over. Some of the main types of results of an RCE attack include:

1) Access to an application or server: initially, attackers have access to functions in the vulnerable application due to a vulnerability. They can use this to inject and run the underlying web server commands.
2) Privilege escalation: gaining access to the web server and executing commands means that attackers may be able to achieve greater privileges and control over the server. This is particularly dangerous if internal vulnerabilities on the server level are present.
Access to data: using an RCE vulnerability, attackers can gain access to and steal data stored on the server.
3) Denial of service: attackers can disrupt and crash the whole service and other services hosted on the server by executing specific commands.
Ransomware and cryptomining: installing various types of malware is possible if code can be executed on the server. Such malware could be crypto mining or cryptojacking software that uses the server's resources to mine cryptocurrency. The remote code execution vulnerability also opens the door to ransomware and having the whole server taken over and held hostage.

Recommendations

It is recommended to update phpMyAdmin to the latest version.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.10 Shellshock bash RCE		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

Upon conducting the penetration testing of the web application, it came to our notice that the application is vulnerable to Shellshock. Shellshock is a vulnerability that affects all operating systems (Linux and Unix based) that allows an attacker to gain complete access to a victim's computer. This vulnerability targets the bash present in the operating system. Bash is a command language interpreter. The attacker sends malicious environment variable to the bash of the web server deployment. The environment variable controls the output of the processes on the system. This bug allows an attacker to exploit any vulnerable versions of Bash to execute any custom commands and will also allow him to gain unauthorized access to a computer system.

Risk

If this vulnerability exists An attacker would be able to read, update and delete arbitrary data/tables from the database. The attacker would also be able to execute commands on the underlying operating system.

Recommendations

The Following are the recommendations that should be implemented.

Try not to process the user data directly as variables in web/bash code.

Make sure to sanitize the user input and remove all un-needed characters so that developers can disrupt an attack before it takes place.

Use systematic scanning.

Affected URL & Parameter

N/A

CVSS Vector

9.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.11 Pickle Code Execution		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

While testing the web application, it came to our notice that the web application is vulnerable to Pickle code execution. Pickle is a Python module that use for serialization and deserialization. It converts code into bytes stream and reconstructs it later for execution. Converted bytes streams contain opcodes which start executing as soon as we load or reconstruct streams back in which will lead to arbitrary code execution.

Risk

If an application is vulnerable to Pickle code execution, it allows an attackers to create his own bytes streams and use it to get arbitrary code execution or remote code execution

Recommendations

The Following are the recommendations that should be implemented.

Encrypt network connection between machine communicating with pickle data.

Strict file permissions on directory to prevent someone from modifying content.

Try to avoid pickle as for security reasons, for data deserialization we can consider JSON or google protocols buffers.

Affected URL & Parameter

N/A

CVSS Vector

9.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.12 Log4j RCE		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

While testing the web application, it came to notice that this web application is vulnerable to log4j vulnerability. In this vulnerability attacker with permission to modify the logging configuration file can construct a malicious configuration using a JDBC Appended with a data source referencing a JNDI URI which can execute remote code.

Risk

If application is vulnerable to log4j vulnerability, it allow attacker to conduct remote code by exploiting the JNDI lookups feature that is not secured within the login library of log4j. The attackers only require a malicious request with a formatted string to be recognized by the Log4j libraries

Recommendations

The Following are the recommendations that should be implemented.

Upgrading a Vulnerable Version (2.16.0 or higher)

Disabling JDNI Lookups (for Log4J >=2.10).

Using outgoing firewall rules on servers is a good mitigation technique to prevent attackers. If the server can make DNS lookups and attackers scan for vulnerable instances of log4j2 which will trigger the DNS lookup.

Affected URL & Parameter

Throughout the Application

CVSS Vector

10.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.13 Blind SQL Injection		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

While conducting the testing of the web application, it came to our notice that the web application is vulnerable to Blind SQL injection. Blind SQL (Structured Query Language) injection is a type of SQL injection attack that asks the database true or false questions and determines the answer based on the applications response. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.

Risk

If an application is vulnerable to blind SQL injection, it could have an effect on data integrity and confidentiality. Additionally, it may have an impact on the application's authentication and authorization features. Depending on the backend database configuration, access privileges and the web server, an intruder might read, edit, or even delete the information being stored in the database.

Recommendations

The Following are the recommendations that should be implemented.
Use prepared statements (with parameterized queries).

Use stored procedures.

Apply input validation and sanitization.

Apply user-supplied input escaping mechanism.

Enforcing least privilege policy.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.14 In-Band SQL Injection		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

In-band SQL Injection is the most common and easy-to-exploit of SQL Injection attacks. In-band SQL Injection occurs when an attacker is able to use the same communication channel to both launch the attack and gather results. This SQL injection was found in the web application, which can be exploited by attackers and should be remediated as soon as possible.

Risk

If an application is vulnerable to in-band SQL injection, it could have an effect on data integrity and confidentiality. Additionally, it may have an impact on the application's authentication and authorization features. Depending on the backend database configuration, access privileges and the web server, an intruder might read, edit, or even delete the information being stored in the database.

Recommendations

The Following are the recommendations that should be implemented.
Use prepared statements (with parameterized queries).

Use stored procedures.

Apply input validation and sanitization.

Apply user-supplied input escaping mechanism.

Enforcing least privilege policy.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.15 Error Based SQL Injection		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

Upon testing the web application, an error based SQL injection vulnerability was found. SQL injection is a severe vulnerability where data can be extracted remotely with by directly interacting with the database with the help of SQL commands by. Any critical information stored in the database comes under the risk of getting leaked. An error based SQL injection is a type of in-band SQL injection. Using a flaw that causes the code to produce a SQL error rather than the necessary data from the server, one can force data extraction through the database. In many cases, the error generated by the database is enough for the attacker to understand the database entirely.

Risk

If an application is vulnerable to error based SQL injection, it could have an effect on data integrity and confidentiality. Additionally, it may have an impact on the application's authentication and authorization features. Depending on the backend database configuration, access privileges and the web server, an intruder might read, edit, or even delete the information being stored in the database.

Recommendations

The Following are the recommendations that should be implemented.

Use prepared statements (with parameterized queries).

Use stored procedures.

Apply input validation and sanitization.

Apply user-supplied input escaping mechanism.

Enforcing least privilege policy.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.16 UNION Based SQL Injection		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

During the conduction of penetration testing of the web application we found that the application is vulnerable to UNION based SQL injection. The UNION keyword is used to retrieve data from other tables in the database when an application is susceptible to SQL injection and the query results are returned in the application's responses. An SQL injection UNION attack is the result of this.

Risk

If an application is vulnerable to UNION based SQL injection, it could have an effect on data integrity and confidentiality. Additionally, it may have an impact on the application's authentication and authorization features. Depending on the backend database configuration, access privileges and the web server, an intruder might read, edit, or even delete the information being stored in the database.

Recommendations

The Following are the recommendations that should be implemented.

Use prepared statements (with parameterized queries).

Use stored procedures.

Apply input validation and sanitization.

Apply user-supplied input escaping mechanism.

Enforcing least privilege policy.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.1 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.17 Time-Based SQL Injection		
CVSS Risk Rating	Critical	Status	Not Applicable

Observation Details

While testing the web application, it came to our notice that the web application is vulnerable to Time-Based SQL Injection. Time based SQL (Structured Query Language) injection is a type of SQL Injection attack that sends queries to the database, and based on the duration of time after which the response is being generated, the attacker proceeds on to attack the database accordingly.

Risk

If an application is vulnerable to time based SQL injection, it could have an effect on data integrity and confidentiality. Additionally, it may have an impact on the application's authentication and authorization features. Depending on the backend database configuration, access privileges and the web server, an intruder might read, edit, or even delete the information being stored in the database.

Recommendations

The Following are the recommendations that should be implemented.

Use prepared statements (with parameterized queries).

Use stored procedures.

Apply input validation and sanitization.

Apply user-supplied input escaping mechanism.

Enforcing least privilege policy.

Affected URL & Parameter

Throughout the Application

CVSS Vector

9.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.18 Insecure Direct Object References		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

During our testing phase, it came to our notice that this application is vulnerable to Insecure Direct Object References. It is a type of access control vulnerability that arises when an application uses user-supplied input to access objects directly in an internal database but does not check for access control or authentication.

Risk

Using this vulnerability, an attacker can gain unauthorized access to sensitive information of application users, or can manipulate the exposed objects to modify data, access hidden functions, or further escalate privileges or direct access file.

Recommendations

The Following are the recommendations that should be implemented.

Developers should avoid displaying private object references such as keys or file names.

Validation of Parameters should be properly implemented.

Tokens should be generated in such a way that it should only be mapped to the user and should not be public.

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.19 Reflected Cross Site Scripting		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

Upon testing the web application, we came across a type of Cross Site Scripting known as reflected XSS. A web application is vulnerable to reflected cross-site scripting attack when the application passes invalidated input from the clients/end-users. In this attack, an attacker creates and tests a malicious URI and initiates a social engineering step, in which the attacker convinces his victims to execute the malicious URI on their browsers. This step by the user allows the execution of the malicious code on their browser. Usually, an attacker uses Javascript for performing this attack, but other scripting languages are also used. e.g., ActionScript and VBScript.

Risk

An attacker might be able to conduct numerous attacks such as account hijacking, credential theft, data leakage, etc. Through these, the attacker would be able to steal sensitive information about the user, steal credentials, or even fake himself as the host application to the end users and redirect them to the malicious application.
The other attacks include key-logger attack, website defacement and port scan.

Recommendations

The Following are the recommendations that should be implemented.

Ensuring that the inputs are properly validated.

Encoding all input fields.

Ensuring all cookie properties are properly set.

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.20 Stored Cross Site Scripting		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

Cross-site Scripting (XSS) is a client-side code injection attack where an attacker can execute malicious scripts into a website or web application. Stored Cross-Site Scripting affects the web applications that allows users to store data. This action can potentially expose the users to this type of attack. Many web applications fail to filter the stored inputs gathered from the users. As these stores that input in a data store for later use. In this case, after conducting the testing, we found that the given web application was vulnerable to this very injection attack.

Risk

An attacker might be able to conduct numerous attacks such as account hijacking, credential theft, data leakage, etc. Through these, the attacker would be able to steal sensitive information about the user, steal credentials, or even fake himself as the host application to the end users and redirect them to the malicious application.
The other attacks include key-logger attack, website defacement and port scan.

Recommendations

The Following are the recommendations that should be implemented.

Ensuring that the inputs are properly validated.

Encoding all input fields.

Ensuring all cookie properties are properly set.

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.3 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.21 LDAP Injection		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

During the conduction of testing of the web application, LDAP injection vulnerability was found in the application. Lightweight Directory Access Protocol injection is a security exploit used to compromise the authentication process. LDAP directories present in websites using LDAP, stores information as objects. If the directory is used for web application's authentication, the attacker can enter malicious code into the user input field. This action by the attacker will gain him, an unauthorized access to the directory, view and change usernames and passwords. LDAP injection is similar to SQL injection because both injection techniques can be exploited because of unsanitized input.

Risk

If servers fail to sanitize user input, the attacker can modify LDAP statements using a local proxy resulting in the execution of malicious commands such as granting permissions to unauthorized queries and content modification inside the LDAP tree. LDAP injection can be used to access information on users, their roles, their permissions and many more. If this information is released, the application may have catastrophic effects on the server.

Recommendations

The Following are the recommendations that should be implemented.

Use the correct LDAP encoding function for escaping the variables.

Try to use frameworks for protecting the application from LDAP injection.

Implement least privilege to LDAP binding account present in the application.

Use a whitelisted input validation technique to protect the server.

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.22 XPath Injection		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

While testing web application, it came to our notice that the web application is vulnerable to XPath injection. XPath injection is a type of attack where a malicious input can lead to un-authorized access or exposure of sensitive information such as structure and content of XML document. It occurs when user's input is used in the construction of the query string.

Risk

Due to this vulnerability attackers can Read, update and delete arbitrary data/tables from the database or can executing commands on the underlying operating system.

Recommendations

The Following are the recommendations that should be implemented.

The user input needs to be sanitized such as quote(') can be replaced with "'". The validation has to be added both in client and server side.

Proper error pages have to used that do not disclose any information in the time of an error that could benefit the attacker.

We can use parametrized queries in which queries are precompiled and user input is passed as parameters rather than expressions.

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.23 Local File Inclusion		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

While testing the web application, it came to our notice that this application is vulnerable to Local file inclusion. Local file inclusion is the process of including files, that are already locally present on the server, through the exploiting of vulnerable inclusion procedures implemented in the application. This vulnerability occurs, for example, when a page receives, as input, the path to the file that has to be included and this input is not properly sanitized, allowing directory traversal characters to be injected.

Risk

Due to this vulnerability an attacker can trick the web application into exposing or running files on the web server. An LFI attack may lead to something as outputting the contents of the file and it can also lead to Code execution on the web server, Code execution on the client-side such as JavaScript which can lead to other attacks such as cross site scripting (XSS), Denial of Service (DoS) and Sensitive Information Disclosure as well.

Recommendations

The Following are the recommendations that should be implemented.

Please avoid passing user-submitted input to any filesystem/framework API.

Please maintain an allow list of files, that may be included by the page, and then use an identifier (for example the index number) to access to the selected file.

Please make the server send download headers automatically instead of executing files in a specified directory.

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.24 Remote File Inclusion		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

While conducting the testing of the web application, we were able to conduct a Remote File Inclusion (RFI) attack. Here an attacker can cause the web application to include a remote file by exploiting a web application. This vulnerability affects the web application that uses external files or scripts. The consequences of a successful RFI attack include Information Disclosure and Cross-site Scripting (XSS) to Remote Code Execution. A server is said to be prone to a remote file inclusion vulnerability because it failed to properly verify user-supplied input.

Risk

An attacker can include arbitrary remote files containing malicious PHP code and execute it in the context of the web server process. Resulting in the attacker to compromise the application and to gain access to the underlying system.

Recommendations

The Following are the recommendations that should be implemented.

Implement proper input validation and sanitization.

Avoiding passing of user-submitted input to any file-system or framework API.

Implement blacklist; identify and block publicly known attackers and malicious URLs, as well as those that have already tried to infiltrate your site or server.

Implement whitelist; create a source of valid and acceptable file types and text.

Enable code reviewing for identifying vulnerabilities in the code.

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.8 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.25 Sensitive information sent via unencrypted channels		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

While testing web application, it came to our notice that the web application is vulnerable to XML Injection. XML Injection is an attack technique used to manipulate or compromise the logic of an XML application or service. The injection of unintended XML content and/or structures into an XML message can alter the intend logic of the application. XML injection vulnerabilities arise when user input is inserted into a server-side XML document or SOAP message in an unsafe way.

Risk

By exploiting this vulnerability, the attacker can steal user credentials, PINs, Session identifiers, Tokens, Cookies, etc. Personal Identifiable Information can also be stolen if it is sent through unencrypted channels, which could lead to serious repercussions. Data corruption, data breach is also possible as mentioned.

Recommendations

It is highly recommended to use secure channels for transmission of data such as HTTPS in web applications.

Affected URL & Parameter

Throughout the Application

CVSS Vector

7.8 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title		8.26 DOM based Cross Site Scripting	
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

While conducting the test of the web application, we are able to conduct a DOM Based Cross Site Scripting attack. Here attacker attacker can manipulate this data to include XSS content on the web page, for example, malicious JavaScript code.A DOM-based XSS attack is possible if the web application writes data to the Document Object Model without proper sanitization

Risk

This vulnerability allow attackers to Steal another client's cookies or sessions, modify them, steal another client's submitted form information or some sensitive credentials and can modify them as well.

Recommendations

To remediate DOM-based XSS, data must not be dynamically written from any un-trusted source into the HTML document. Security controls must be in place if the functionality requires it.

Sanitize all untrusted data, even if it is only used in client-side scripts.

If you have to use user input on your page, always use it in the text context, never as HTML tags or any other potential code

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.27 Session Replay/Token Reuse		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

During our testing, we have noticed that session replay attack is possible. In a Session Replay Attack, an attacker steals a valid session ID of a user, and reuses it to impersonate an authorized user to perform fraudulent transactions/activities. Web applications that allow reusing old session IDs or session credentials for authorization are also vulnerable to Session Replay Attacks.
Here, as noticed, the session cookie is used to login in the account and there is no expiration time set for that cookie. This can lead to session theft and account stealing of the user. If the account session id is stolen, then attacker can use the cookie and the session id for getting logged into the account. It is very important to maintain the privacy and the confidentiality of the session id's and tokens. It is better to authenticate and validate the cookies on each session the user creates to get in the account.

Risk

In this case, the attacker can conduct session replay attack which can lead to session theft and account stealing of the user. If the account session id is stolen, then attacker can use the cookie and the session id for getting logged into the account. This can be dangerous and the repercussions can be worse if the login credentials of an admin are stolen.

Recommendations

The Following are the recommendations that should be implemented.

Set the expiration time for the session.

Data in encrypted form will be very helpful.

Configure the web application for proper session timeouts.

Affected URL & Parameter

Throughout the Application

CVSS Vector

7.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.28 No Rate Limit - Account Bruteforce		
CVSS Risk Rating	High	Status	Open

Observation Details

Rate limiting is a process to limit requests possible. It is used in web applications in order to prevent brute force attacks. This is necessary to prevent the attackers from logging in accounts by brute forcing credentials or even sending excessive requests to the server.

No rate limit is a flaw that doesn't limit the no. of attempts one makes on a website server to extract data. It is a vulnerability which can prove to be critical when misused by attackers.

Risk

• Identity theft – stealing someone's identity to access their accounts, such as bank accounts or credit cards. This enables the attacker to purchase goods using these details. In addition, information such as social security numbers can be sold for use in other cyber-attacks.
• Loss of data – due to loss of confidentiality if data is stolen which could destroy company reputation. Additionally, there may be reputational damage caused by a leak of sensitive customer information that leads to public distrust and dissatisfaction with the business.
• Downtime – this refers to system outages where websites or computer networks cannot be accessed due to a cyber-attack. This is costly to the business in terms of lost revenue, customer satisfaction as well as loss of image.

Recommendations

The Following are the recommendations that should be implemented.
• Never use information that can be found online (like names of family members).
• Have as many characters as possible.
• Combine letters, numbers, and symbols.
• To protect your organization from brute force password hacking, enforce the use of strong passwords.
• Be different for each user account.
• Avoid common patterns.
• Lockout policy
• Progressive delays
• Authorization should be properly validated from the server side.

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

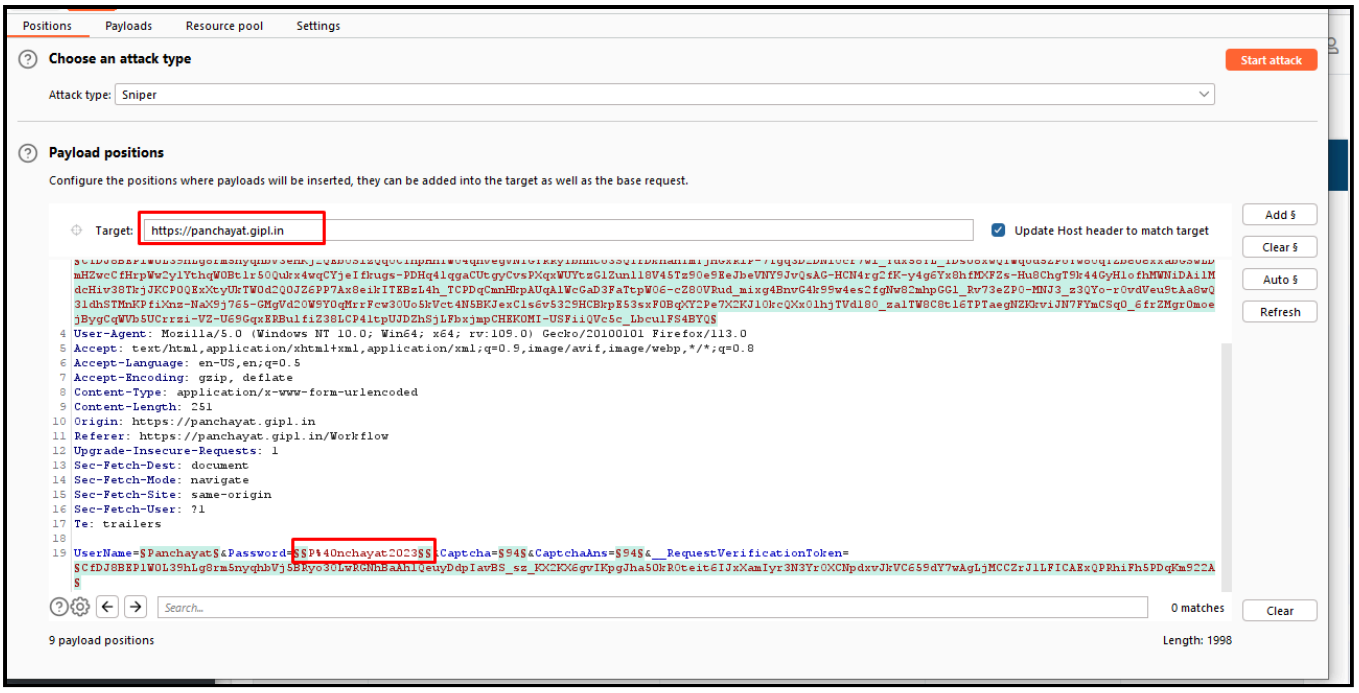


Figure 4 - Intercept the login request > Send the request to Intruder. Select the password as a attack vector.

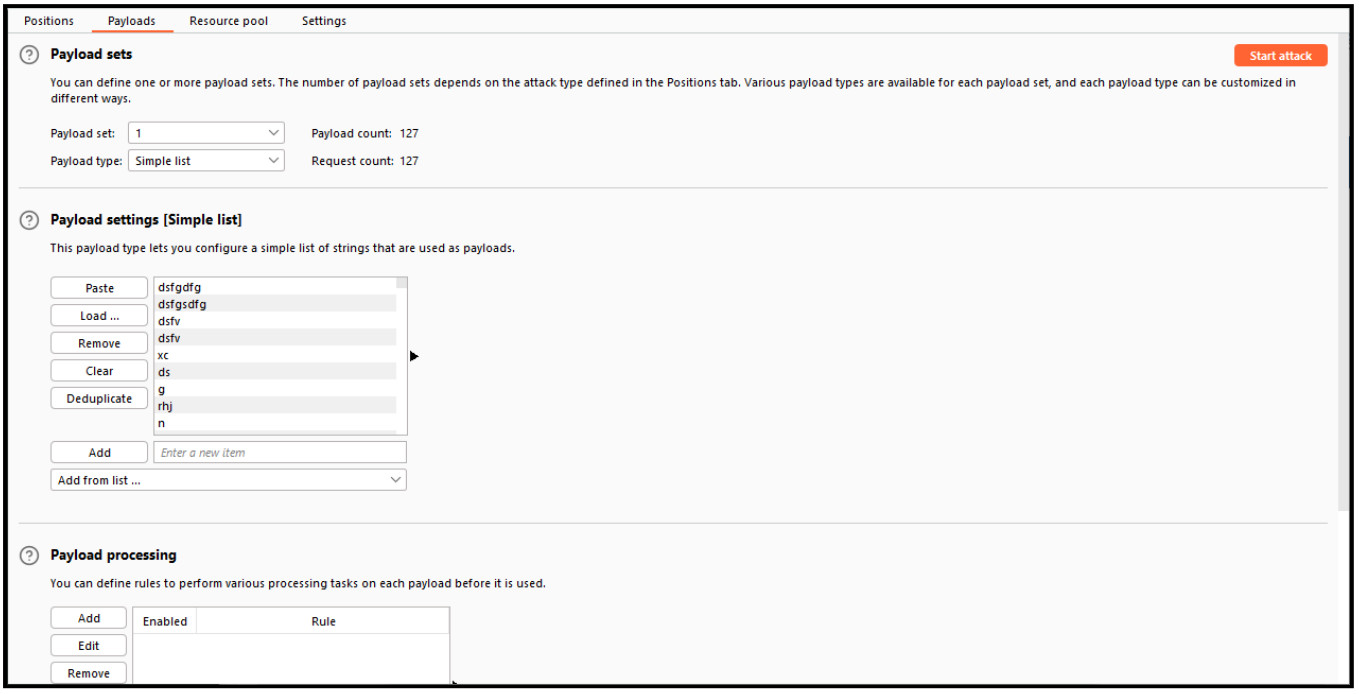


Figure 5 - Select simple list > Upload the wordlist for the attack.

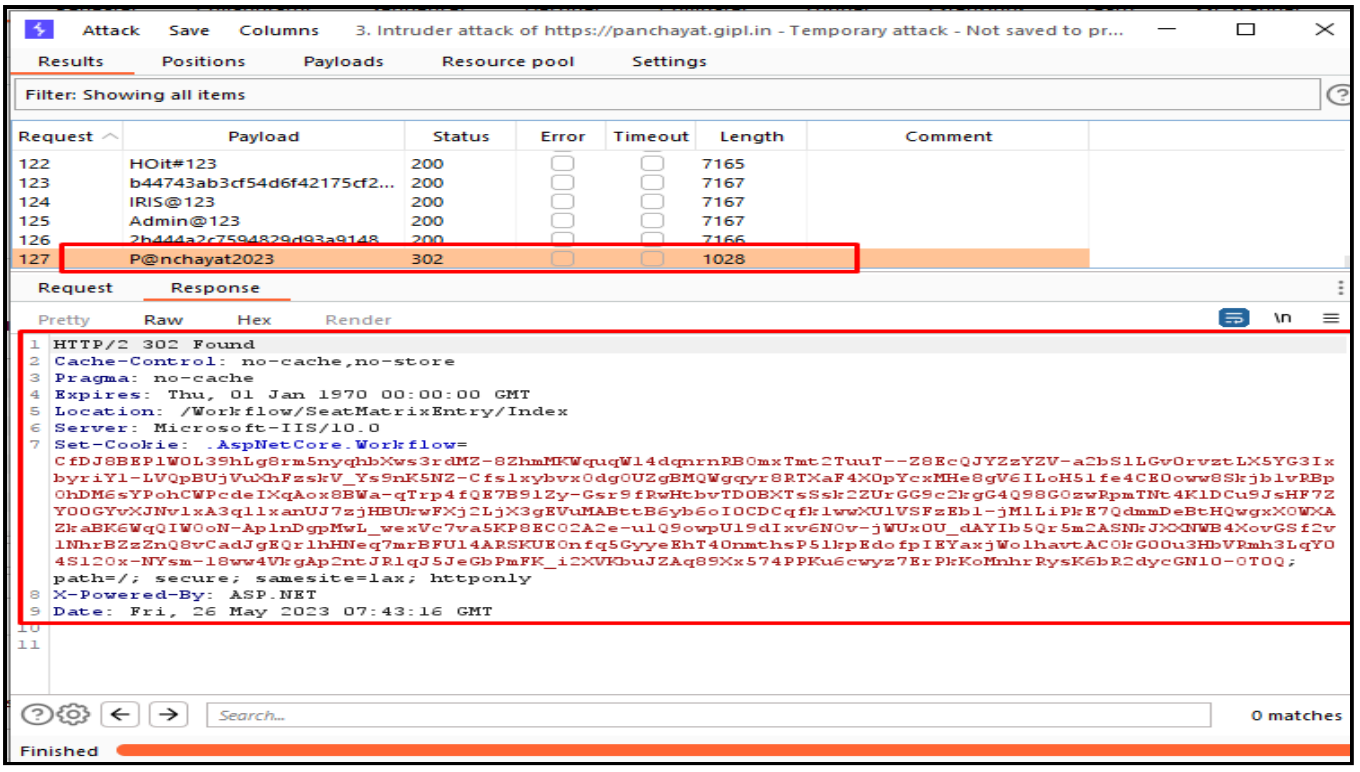


Figure 6 - Start the attack > you will get the password as response of Status 302 Found.

Observation ID & Title	8.29 Broken Authentication		
CVSS Risk Rating	High	Status	Open

Observation Details

The application does not properly invalidate a user's session on the server after the user initiates logout. User sessions remain active on the server, and any requests submitted including the user's session identifier will execute successfully, as though the user had made those requests.

Risk

An attacker can use previous used or available session token to change anything in the user account without logging in the user account.

Recommendations

The Following are the recommendations that should be implemented.

The user's HTTP session should be terminated on the server immediately after a logout action is performed. It is important to note that simply deleting the cookie from the browser will not terminate the server session. The session must be invalidated at the server, using the HTTP container's intrinsic session abandonment mechanism.

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

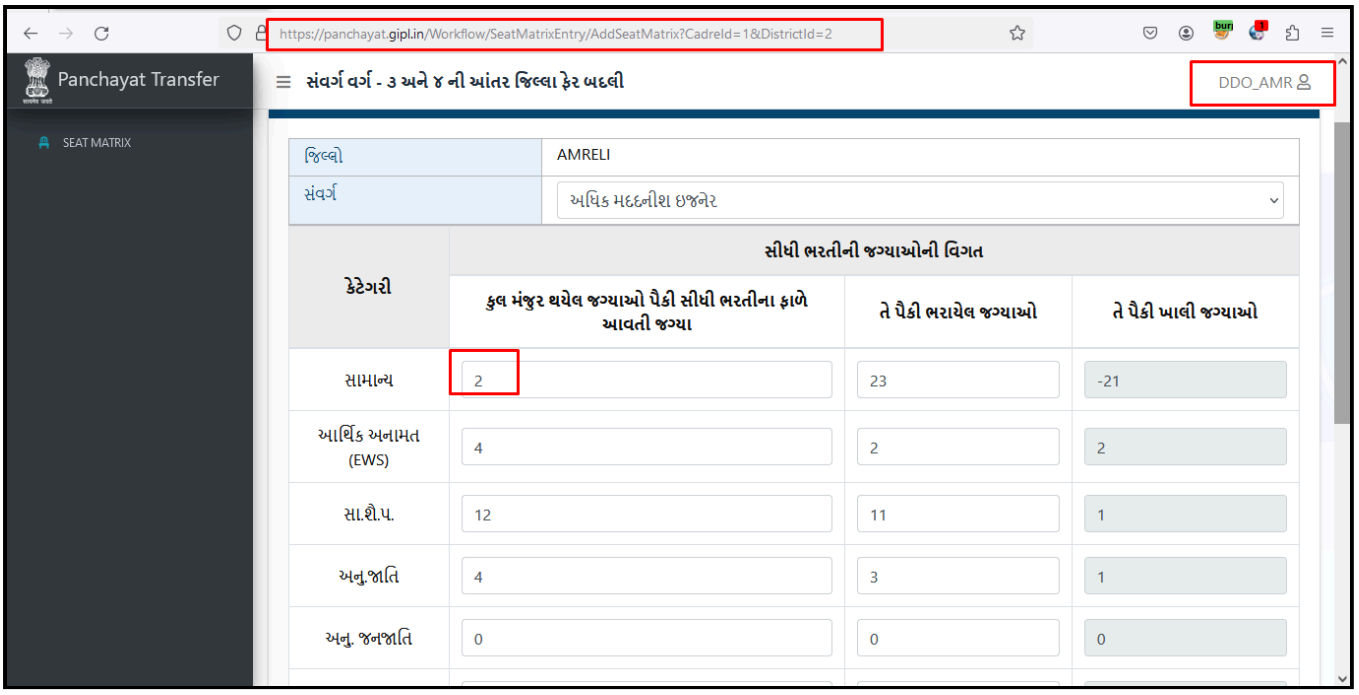


Figure 7 - Intercept the request by clicking on save in the Edit Details.

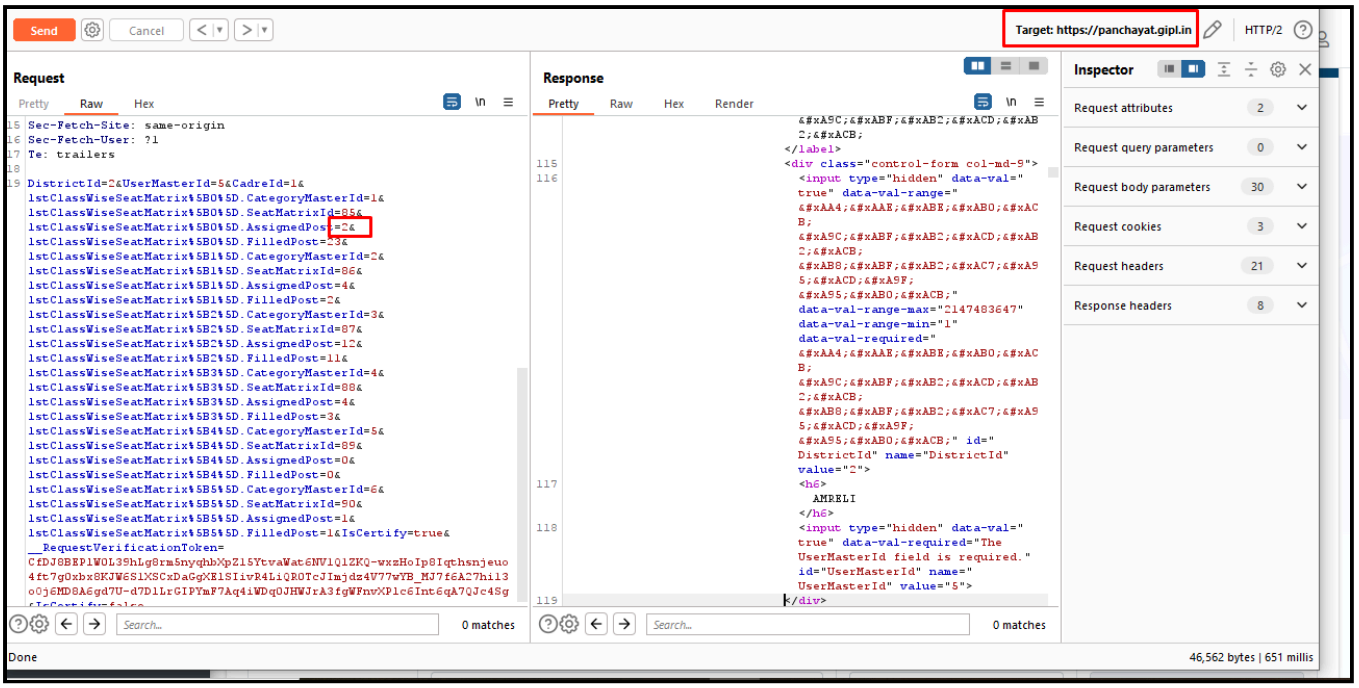


Figure 8 - Send the request to Repeater > Logout the account.

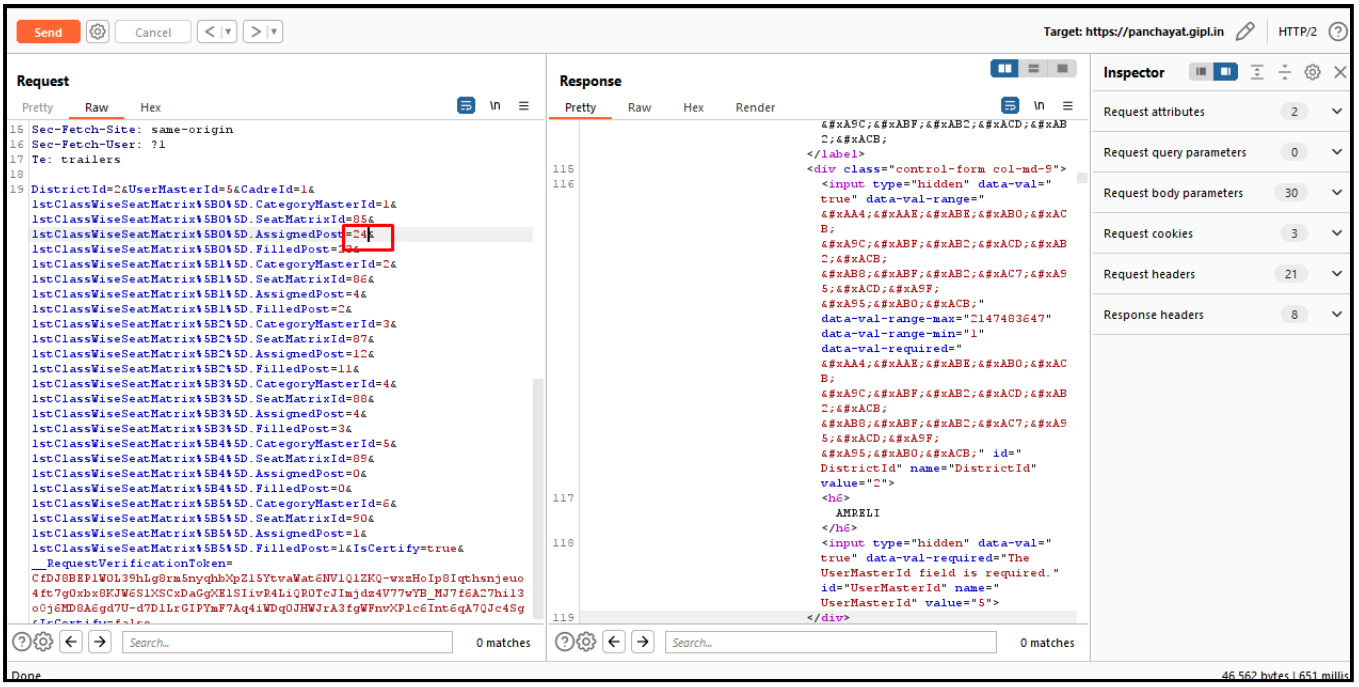


Figure 9 - Now in the repeater, change the value from 2 to 24. Click on Send.

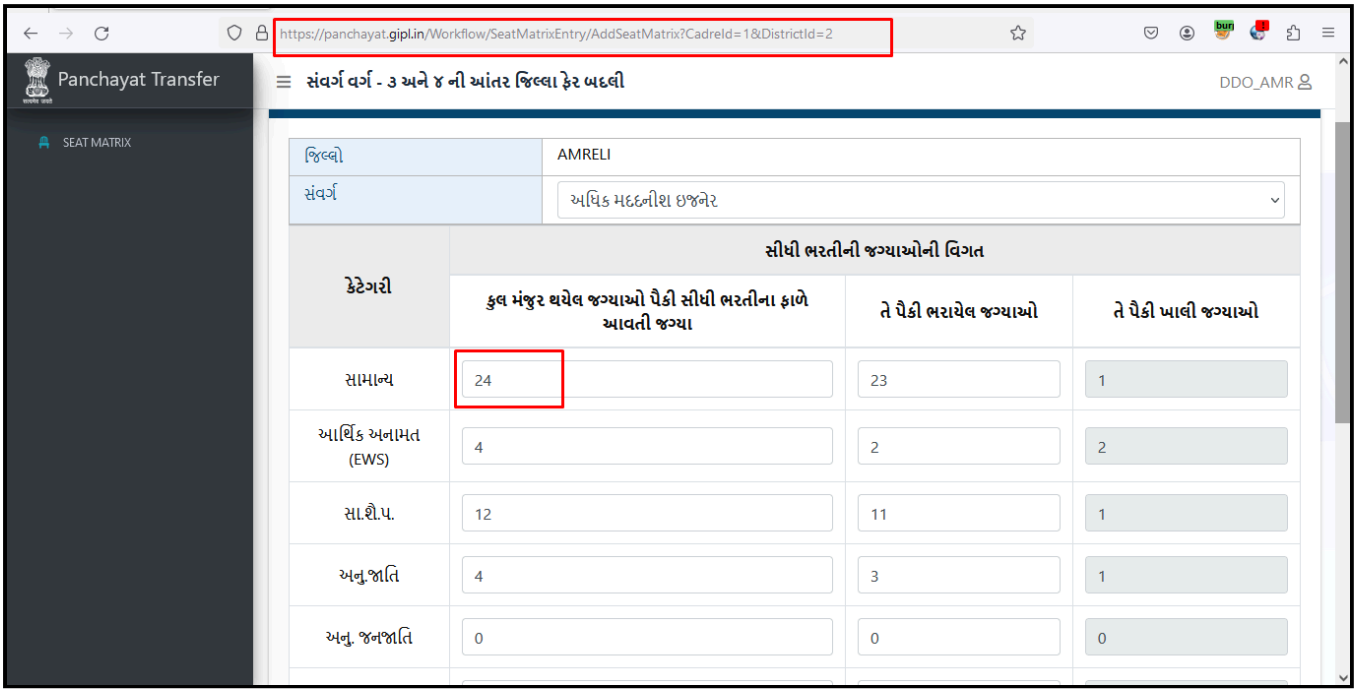


Figure 10 - Login again the account and you will see that the value has been change to 24.

Observation ID & Title	8.30 Application-Level Denial-of-Service (DoS)		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

An application-level Denial-of-Service (DoS) attack is a type of attack that targets a web application or other networked application, rather than the network itself. In this type of attack, the attacker sends a large number of requests to the application, overwhelming its resources and causing it to crash or become unresponsive.

Risk

The impact of an application-level DoS attack can be severe, as it can prevent users from accessing the application, potentially leading to lost revenue and damage to the organization's reputation. Additionally, depending on the nature of the application, an attack may be able to obtain sensitive information or perform unauthorized actions.

Recommendations

To mitigate the risk of an application-level DoS attack, it is recommended to:-

Implement appropriate security measures. This can include implementing rate limiting.

Use firewalls and other security measures to block or limit the number of requests that can be sent to the application.

Ensure that the application is properly designed and tested to handle a high volume of requests without crashing or becoming unresponsive.

Affected URL & Parameter

N/A

CVSS Vector

7.6 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.31 Server-Side Request Forgery (SSRF)		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

Server-Side Request Forgery (SSRF) is a type of attack that allows an attacker to send arbitrary requests from a vulnerable server to other internal resources. This can include accessing sensitive information, such as system files or passwords, or using the server to carry out other malicious actions.

Internal SSRF attacks are a specific type of SSRF attack that occur when the vulnerable server is able to access internal resources on the same network. This can include other servers, databases, or other sensitive assets.

Risk

The impact of an internal SSRF attack can be significant, as it allows the attacker to gain access to sensitive information and potentially compromise the security of the entire network. This can result in the disclosure of confidential data, or the manipulation of data in ways that can cause harm to the system or its users.

Recommendations

To mitigate the risk of an internal SSRF attack, it is highly recommended to:-

Use a whitelist of approved domains and protocols through which remote resources can be acquired by the web server.

User input should always be sanitised or validated.

One must verify that the server response received is as planned to avoid response data leakage to an attacker. The raw response body of the request sent by the server should not be delivered to the client under any circumstances.

If only HTTP or HTTPS are used by your application to make requests, allow only these URL schemas. If URL schemas like file:///, dict://, ftp:// and gopher:// are disabled, the attacker won't be able to use the web application to make dangerous requests using these URL schemas.

Services like Memcached, Redis, Elasticsearch, and MongoDB do not need authentication by default. Server Side Request Forgery vulnerabilities may be used by an attacker to access any of these services without any authentication. Therefore it is best to allow authentication wherever possible, even for services on the local network to ensure security for the web application.

Affected URL & Parameter

N/A

CVSS Vector

8.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.32 Second Factor Authentication (2FA) Bypass		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

During our testing we notice that the web application is vulnerable to Second Factor Authentication (2FA) Bypass. Second factor authentication, or 2FA, is a security measure that requires users to provide two different forms of authentication in order to access a system or service. A 2FA bypass vulnerability refers to a weakness in a system's 2FA implementation that allows an attacker to bypass the requirement for a second factor of authentication and gain unauthorized access to the system.

Risk

Second Factor Authentication (2FA) Bypass allows attackers to gain unauthorized access to a system or service without needing to provide the second factor of authentication. This can potentially allow attackers to steal sensitive information, such as passwords or financial data, or to perform other malicious actions on the system .

Recommendations

The Following are the recommendations that should be implemented.

Create complex passwords, as attackers can easily brute force simple passwords.

Avoid the use of short, numerical OTPs where possible, opting instead for a longer alphanumeric combination with upper and lower case characters.

Use biometric authentication as at least one factor of authentication – it's much harder to bypass a thumbprint than a 4-digit code.

Affected URL & Parameter

Throughout the Application

CVSS Vector

7.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.33 Hardcoded Password - Non-Privileged User		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

During our testing we notice that web application has Hardcoded Password of Non-privileged user. A hardcoded password vulnerability occurs when a password is stored in a system or application in an unencrypted or easily accessible format. This can happen if a developer hardcodes a password into the application's source code, or if a password is stored in a configuration file or other easily accessible location.

Risk

The impact of a hardcoded password vulnerability can allows an attacker to get unauthorized access to the system and get sensitive information about application or the user how's password is hardcoded in the application.

Recommendations

The Following are the recommendations that should be implemented.

Avoiding the use of hardcoded passwords in your applications and systems, and instead using secure, encrypted password storage mechanisms.

Implementing strong authentication and access control measures, such as two-factor authentication and role-based access control, to prevent unauthorized access to sensitive information and resources.

Restrict access to all files/systems that store credentials such as configuration files or databases.

Affected URL & Parameter

N/A

CVSS Vector

7.8 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.34 Token Leakage via Host Header Poisoning		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

Token leakage via host header poisoning is a type of security vulnerability that occurs when an attacker is able to gain access to sensitive information by manipulating the host header of an HTTP request. The host header is a part of the HTTP header that specifies the domain name of the server that the client is trying to access. By modifying the host header, an attacker can cause the client to send requests to a different server that is controlled by the attacker, allowing the attacker to access sensitive information, such as authentication tokens or other sensitive data.

Risk

Token Leakage via host header poisoning vulnerability can allow an attacker to take control of a particular individual's account or perform unauthorize action, access sensitive information of an individual by leveraging Password Reset Poisoning.

Recommendations

The Following are the recommendations that should be implemented.

Validate the headers that supplied into the requests.

Also use multi-factor authentication to prevent account hijacking , and one such method is SMS Authentication.

It is also important to check that you do not support additional headers that may be used to construct these attacks,

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.35 OAuth Misconfiguration - Account Takeover		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

OAuth is a widely-used standard for authorization, which allows users to grant third-party applications access to their data or resources without sharing their login credentials. OAuth misconfiguration can lead to account takeover attacks, where an attacker is able to gain unauthorized access to a user's account. This typically occurs when an OAuth implementation is not properly configured, allowing an attacker to obtain a user's access token and use it to access the user's account.

Risk

The impact of this vulnerability can be significant, as it can allow attackers to access sensitive information, perform actions on behalf of the user, and potentially compromise the security of the entire system.

Recommendations

To mitigate this vulnerability, it is important to properly configure your OAuth implementation and follow best practices for secure OAuth deployment. This may include implementing measures such as proper authentication and authorization protocols, regularly rotating access tokens, and enforcing secure password policies.

Affected URL & Parameter

N/A

CVSS Vector

8.6 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.36 Cryptographic Flaw		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

while conducting the testing for the web application, it come to our notice that the web application is vulnerable to Cryptographic Flaw vulnerability. A cryptographic flaw is a weakness in a cryptographic algorithm or system that can be exploited by attackers to compromise the security of the system. Cryptographic flaws can be caused by a variety of factors, such as design or implementation errors, or by the use of outdated or inadequate cryptographic techniques.

Risk

Cryptographic Flaw vulnerability can allow an attacker can get hold of a complete database having thousands of sensitive information, data theft, public listing, breaches, and many critical problems with business-related data

Recommendations

The Following are the recommendations that should be implemented.

Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management.

Ensure that cryptographic randomness is used where appropriate, and that it has not been seeded in a predictable way or with low entropy. Most modern APIs do not require the developer to seed the CSPRNG to get security.

Encrypt all data in transit with secure protocols such as TLS with forward secrecy (FS) ciphers, cipher prioritization by the server, and secure parameters.

Affected URL & Parameter

Throughout the Application

CVSS Vector

7.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.37 Authentication Bypass		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

Authentication bypass is a type of security vulnerability that allows an attacker to bypass the authentication process of a system, gaining unauthorized access to protected resources or functionality. This can be accomplished in a number of ways, such as by exploiting weaknesses in the authentication system, using default or easily guessable credentials, or manipulating the authentication process to gain access without providing valid credentials.

Risk

The impact of an authentication bypass vulnerability can be severe, as it allows attackers to gain unauthorized access to sensitive information or resources, potentially leading to data breaches, financial losses, and reputational damage.

Recommendations

To mitigate the risks associated with authentication bypass vulnerabilities, it is recommended to:

Implement strong authentication mechanisms and regularly update them to address known weaknesses. This can include using multi-factor authentication, requiring complex and unique passwords, and regularly monitoring and auditing authentication systems for signs of unauthorized access.

Implementing security measures such as firewalls and intrusion detection and prevention systems can help protect against authentication bypass attacks.

Affected URL & Parameter

N/A

CVSS Vector

8.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.38 Information Disclosure		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

while conducting the penetration testing, we found out that the application disclose sensitive information. It happens when a web site unintentionally leaks some sensitive information to its users. Website may leak all kind of information to potential attackers such as user's accounts credentials, technical details about the site, sensitive files, and many more.

Risk

If this vulnerability is found in an application, an attacker might be able to see sensitive information about the application such as components used in application, architecture of application, and many more. Based on these information attacker can easily attack the application.

Recommendations

The Following are the recommendations that should be implemented.

Make sure that your web server does not send out response headers or background information that reveal technical details about the backend technology type, version or setup.

Do not hardcode credentials, API keys, IP addresses, or any other sensitive information in the code.

Make sure that all exceptions are well handled when the web application fails and no technical information is reported in the errors.

Affected URL & Parameter

N/A

CVSS Vector

7.8 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.39 Unencrypted Communication		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

Upon testing the application , it came to our notice that the application is using using unencrypted Communication. The application allows users to connect to it over unencrypted connections. An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the application and obtain any information the user supplies. Furthermore, an attacker able to modify traffic could use the application as a platform for attacks against its users and third-party websites. Unencrypted connections have been exploited by ISPs and governments to track users, and to inject adverts and malicious JavaScript. Due to these concerns, web browser vendors are planning to visually flag unencrypted connections as hazardous.

Risk

Attackers can sniff the network traffic and accessing the information transferred through an unencrypted channel.

Recommendations

The Following are the recommendations that should be implemented.

Applications should use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server. The Strict-Transport-Security HTTP header should be used to ensure that clients refuse to access the server over an insecure connection.

Affected URL & Parameter

N/A

CVSS Vector

8.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.40 F5 Big IP Exploitation		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

F5 BIG-IP is a popular web application firewall and load balancer used by many organizations to improve the performance and security of their web applications. However, like any software, F5 BIG-IP is subject to exploitation by attackers who can exploit vulnerabilities in the software to gain unauthorized access to systems or steal sensitive information.

Risk

The impact of an F5 BIG-IP exploit can be significant for both the organization using the software and its users. For the organization, it can result in loss of user trust and damage to the organization's reputation. For the users, it can lead to financial loss or identity theft if their sensitive information is accessed and used by the attacker.

Recommendations

To mitigate the risk of F5 BIG-IP exploitation, it's important for organizations to regularly update their F5 BIG-IP software to the latest version, which includes patches for known vulnerabilities. Additionally, organizations can implement security measures such as encryption and authentication to protect sensitive information transmitted through the F5 BIG-IP system. Regular security assessments and penetration testing can also help to identify and address potential vulnerabilities in the system.

Affected URL & Parameter

Throughout the Application

CVSS Vector

7.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.41 Play Session Injection		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

During the testing of the web application, we found that the application was using Play framework, which was vulnerable to session injection attack. Session injection is a type of attack in which an attacker injects malicious data into a user's session in order to gain unauthorized access to a website or web application. In the case of the Play Framework, this attack can be performed by modifying the session identifier stored in a user's cookies, or by manipulating the data sent to and from the server in a user's session.

Risk

If an organization is successfully targeted by a session injection attack, the impact can be significant. The attacker may be able to gain unauthorized access to user accounts, steal sensitive information, or perform unauthorized actions on the user's behalf. This can result in loss of confidentiality, integrity, and availability of the organization's data, as well as damage to its reputation and financial losses.

Recommendations

The Following are the recommendations that should be implemented.

Use strong encryption for data transmission, such as Secure Sockets Layer (SSL) or Transport Layer Security (TLS), to protect against session injection attacks.

Implement robust authentication mechanisms, such as multi-factor authentication, to verify the identity of users and prevent attackers from gaining access to user accounts.

Regularly monitor systems for signs of malicious activity, such as unusual login attempts or unauthorized access to sensitive data.

Use the built-in security features of the Play Framework, such as support for secure cookies and CSRF protection, to help prevent session injection attacks.

Affected URL & Parameter

Throughout the Application

CVSS Vector

7.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.42 JS Prototype Pollution		
CVSS Risk Rating	High	Status	Not Applicable

Observation Details

During the testing phase, it came to our notice that the application is vulnerable to JS Prototype pollution. JS Prototype Pollution is a vulnerability that allows attackers to exploit java script runtimes. It allows attackers to overwrite the prototype of the base object and this prototype is passed to many objects that inherit the base object due to which the attacker has a control over it and can perform DOS or RCE.

Risk

If an application is vulnerable to JS Prototype Pollution, it allow attacker to inject malicious code in existing java script trigger java script exception causing denial of service, remote code execution, etc.

Recommendations

The Following are the recommendations that should be implemented.

Create object without java prototype.

Prevent any changes to prototype.

Affected URL & Parameter

Throughout the Application

CVSS Vector

8.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.43 Review Old, Backup and Unreferenced Files for Sensitive Information		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While most of the files within a web server are directly handled by the server itself, it isn't uncommon to find unreferenced or forgotten files that can be used to obtain important information about the infrastructure or the credentials.

Most common scenarios include the presence of renamed old versions of modified files, inclusion files that are loaded into the language of choice and can be downloaded as source, or even automatic or manual backups in form of compressed archives. Backup files can also be generated automatically by the underlying file system the application is hosted on, a feature usually referred to as "snapshots".

All these files may grant the tester access to inner workings, back doors, administrative interfaces, or even credentials to connect to the administrative interface or the database server.

Risk

- Unreferenced files may disclose sensitive information that can facilitate a focused attack against the application; for example include files containing database credentials, configuration files containing references to other hidden content, absolute file paths, etc.
- Unreferenced pages may contain powerful functionality that can be used to attack the application; for example an administration page that is not linked from published content but can be accessed by any user who knows where to find it.
- Old and backup files may contain vulnerabilities that have been fixed in more recent versions; for example viewdoc.old.jsp may contain a directory traversal vulnerability that has been fixed in viewdoc.jsp but can still be exploited by anyone who finds the old version.

Recommendations

To guarantee an effective protection strategy, testing should be compounded by a security policy which clearly forbids dangerous practices, such as:

- Editing files in-place on the web server or application server file systems. This is a particularly bad habit, since it is likely to generate backup or temporary files by the editors. It is amazing to see how often this is done, even in large organizations. If you absolutely need to edit files on a production system, do ensure that you don't leave behind anything which is not explicitly intended, and consider that you are doing it at your own risk.
- Carefully check any other activity performed on file systems exposed by the web server, such as spot administration activities. For example, if you occasionally need to take a snapshot of a couple of directories (which you should not do on a production system), you may be tempted to zip them first. Be careful not to leave behind those archive files.
- Appropriate configuration management policies should help prevent obsolete and un-referenced files.
- Applications should be designed not to create (or rely on) files stored under the web directory trees served by the web server. Data files, log files, configuration files, etc. should be stored in directories not accessible by the web server, to counter the possibility of information disclosure (not to mention data modification if web directory permissions allow writing).

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.44 Bypassing authentication schema		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Bypassing an authentication schema refers to the act of circumventing the controls put in place to verify a user's identity and grant them access to a system or application. This can be done in a variety of ways, including using stolen or default login credentials, exploiting vulnerabilities in the authentication process, or using social engineering techniques to trick a user into revealing their login information.

Risk

Bypassing an authentication schema can have serious consequences, as it allows attackers to gain unauthorized access to a system or application. This can allow them to steal sensitive information, perform unauthorized actions, or damage the system. It can also potentially lead to legal liabilities for the organization that operates the system or application.

Recommendations

To prevent bypassing of their authentication schema, it is advised to implement strong and secure authentication controls. This can include things like requiring complex and unique passwords, implementing two-factor authentication, and regularly monitoring and testing their authentication processes for vulnerabilities. By taking these steps, it can protect against unauthorized access and maintain the security of their systems and applications.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.45 Improper Cache Control		
CVSS Risk Rating	Medium	Status	Open

Observation Details

While testing web application, it came to our notice that, this application is vulnerable to Improper cache control. Cache-control is an HTTP header that dictates browser caching behavior. when someone visits a website, their browser will save certain resources, such as images and website data, in a store called the cache. When that user revisits the same website, cache-control sets the rules which determine whether that user will have those resources loaded from their local cache or not.

Risk

Using this vulnerabilities, an attacker can view sensitive information even if he is not logged into the account

Recommendations

The Following are the recommendations that should be implemented.

The web server should return the following HTTP headers in all responses containing sensitive content:

Cache-control: no-store
Pragma: no-cache

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.3 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

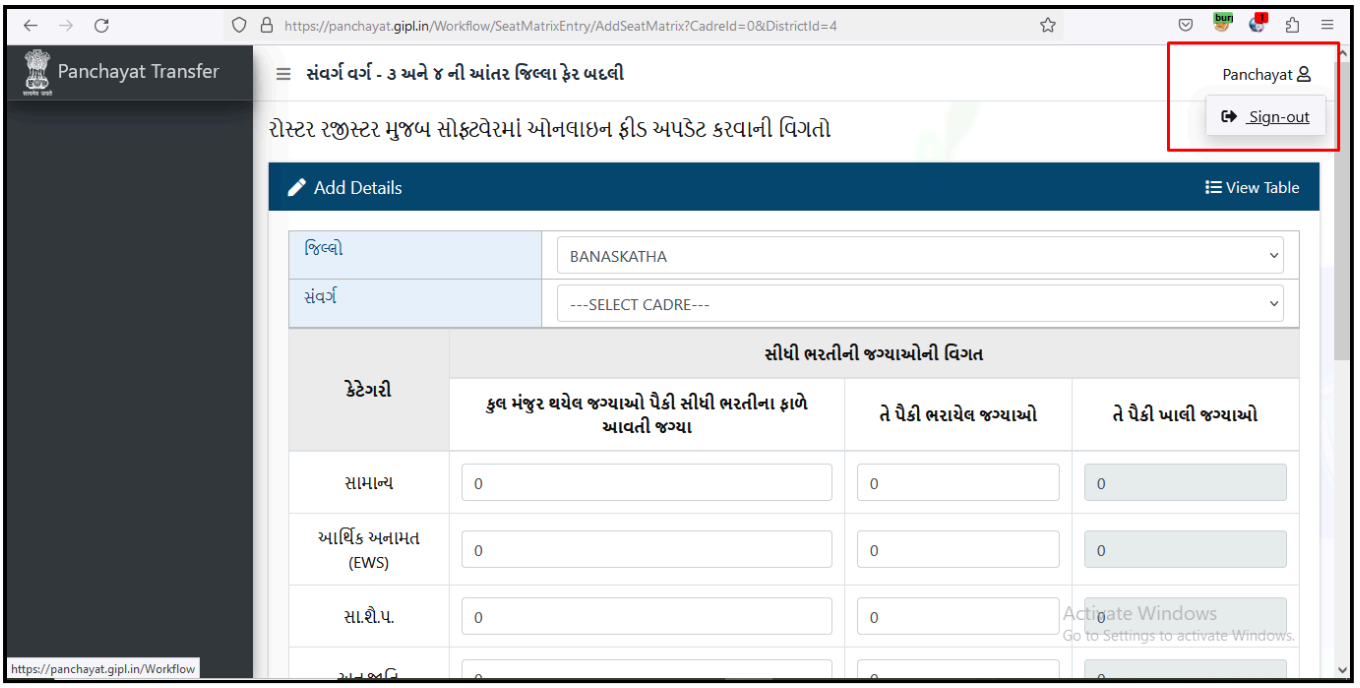


Figure 11 - Login in the account and access any page.

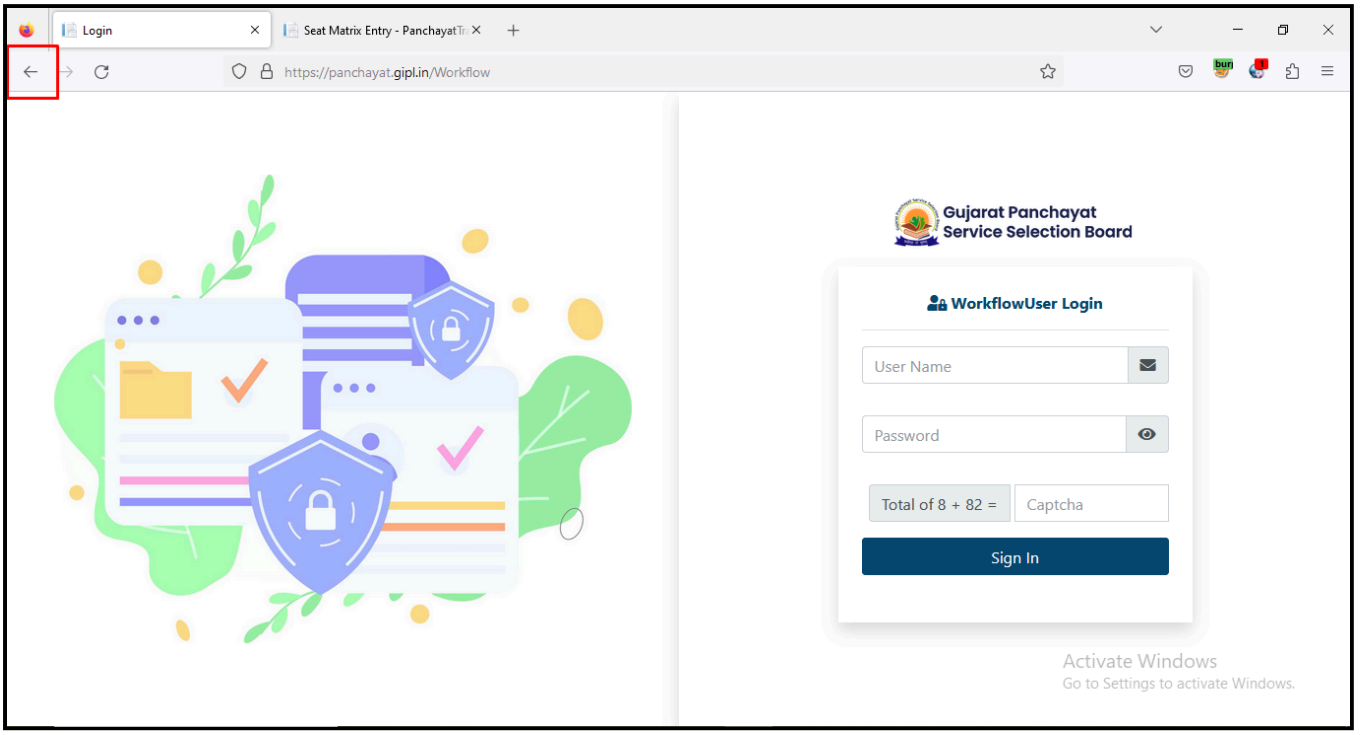


Figure 12 - Logout the account and click on back button.

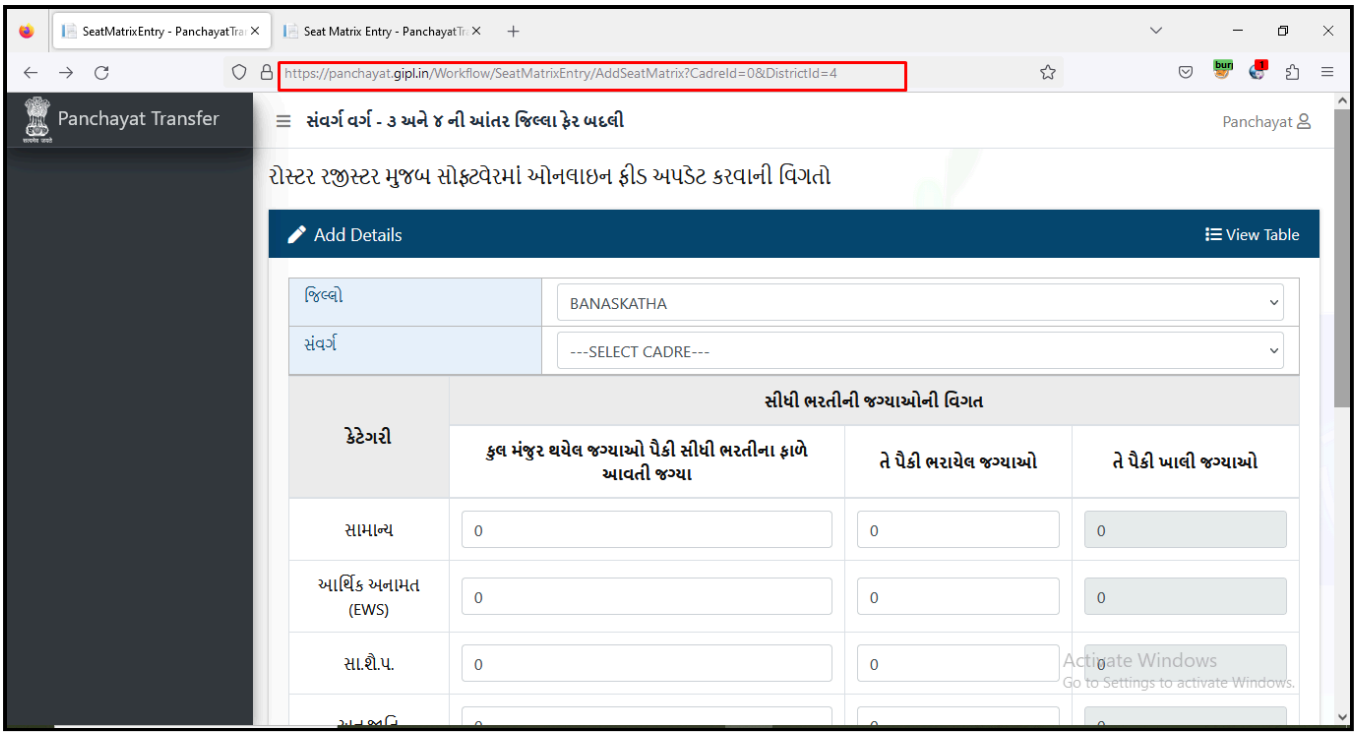


Figure 13 - As you can see that the application is returned to the previous page without any authentication.

Observation ID & Title	8.46 Weak password change or reset functionalities		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Having a weak password change or reset functionality in a web application can pose a potential security vulnerability. This is because it can make it easier for attackers to gain access to a user's account.

A weak password change or reset functionality might allow users to choose a new password that is easy to guess or that does not meet the minimum security requirements. For example, a password reset functionality might allow users to choose a new password that is only a few characters long or that does not contain any numbers or special characters. This can make it easier for attackers to guess or crack the user's password, potentially giving them access to the user's account.

Risk

An attacker can compromise the security of a user's account, get access to the user's account, access to sensitive information such as the user's personal details, financial information, etc. A weak password change or reset functionality can also leave the web application vulnerable to legal liability. If users' personal information is compromised due to a weak password change or reset functionality, the web application and its developers could be held legally responsible for any harm that results. This could result in costly legal fees and damages.

Recommendations

The Following are the recommendations that should be implemented.

It's important for web developers to ensure that their password change and reset functionality is secure and not vulnerable to attack. This can help protect their users' accounts and prevent attackers from gaining access to sensitive information.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.47 Directory traversal/file include		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While testing the web application, we found that directory traversal was possible in the given web application. Directory listing allows an attacker to access files outside the Webroot folder through an HTTP communication. An attacker can get access to sensitive information and can execute arbitrary codes on the server. Usually, every web server implements security in 2 levels:-

Access Control Lists: The access control list is used to give access, modify or execute arbitrary files on the server. The list consists of the users or group with their permission list.

Root directory restriction: Under this restriction, the users are not allowed to access any files outside the Webroot. The cause of directory traversal attack might be due to a flaw in the code.

Risk

If an attacker successfully exploits a server using a directory traversal attack, he will be able to access sensitive files outside the server. The attacker could access files like passwd to get the username and passwords all the users in the application. The attacker can also input malicious files into the server causing file inclusion attack and steal sensitive information from the server.

Recommendations

The Following are the recommendations that should be implemented.

Sanitize the user inputs from the application.

Set privileges to the API in such a way that it allows inclusion of files from one allowed directory only.

Implement blacklisting of all the special characters that are not used in the file names.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.48 Bypassing Session Management Schema		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Session management is a security mechanism used to track and manage user sessions in a web application. If an attacker is able to bypass the session management schema, they can potentially gain unauthorized access to a user's account or perform actions on behalf of the user.

Risk

The impact of this vulnerability can be significant, as it can allow an attacker to access sensitive information, perform actions on behalf of the user, and potentially compromise the security of the entire system.

Recommendations

To mitigate this vulnerability, it is important to properly implement and maintain secure session management in your web application. This may include using secure session identifiers, regularly rotating session tokens, and implementing proper authentication and authorization protocols.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.49 Session Fixation		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

After the conduction of testing of the web application, it came to our notice that the application is vulnerable to an exploit known as a "Session Fixation" attack. This enables an attacker to hijack and take control of a valid user session. Using this session, the attacker discovers many vulnerabilities. A hacker has the potential to hijack a legitimate user session on a server that has this vulnerability. The server does not create a new session ID when a user is authenticated for a session. Using an existing session ID is made feasible by this. The validated user session can be taken over by the attacker.

Risk

The attacker can hijack the user's validated session. This is possible because the attacker has the knowledge of the used session ID.

Recommendations

The Following are the recommendations that should be implemented.

Ignoring the session IDs provided by user's browser and generate a new session for every new successful authentication.

Ensure to not include the cookie value in the URL.

Updating the session software packages to the latest version.

Affected URL & Parameter

Throughout the Application

CVSS Vector

6.1 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.50 Cross Site Request Forgery		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Cross-Site Request Forgery (CSRF), commonly referred to as XSRF, is a type of attack that causes users who have access to a web application to carry out unauthorized operations on it. It will typically appear to be a regular connection to the web application. However, if the parameter is changed, the HTTP request will be sent to the web application in the attacker's planned manner. It was once used to send requests for state changes, including resetting the password or performing account transactions. The likelihood of the application being compromised increases as the privilege level does. If the victim is a normal user account, it may be used to change the password for the mail account or even to transfer money. If the Administrator account is compromised, the entire program will be at risk.

Risk

In this scenario, attackers might get full access to the application and they also might be able to perform account takeovers. This may lead to data loss and also result in loss of trust from the user.

Recommendations

It is highly recommended to:-

Use Anti-CSRF tokens.

Implement Same-site Cookies.

Affected URL & Parameter

Throughout the Application

CVSS Vector

6.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.51 HTTP Parameter pollution		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While conducting the penetration testing of the web application, it came to our notice that there was a HTTP Parameter Pollution vulnerability existing in the web application. HTTP Parameter Pollution (HPP) is a Web attack evasion technique that allows an attacker to craft a HTTP request in order to manipulate or retrieve hidden information. Some environments process such requests by concatenating the values taken from all instances of a parameter name within the request. This behavior is abused by the attacker in order to bypass pattern-based security mechanisms.

Risk

An attacker can use an HPP attack to perform many different unwanted actions. They can override the existing hardcoded HTTP parameters, modify the application behavior and access and exploit the user-uncontrollable variables. HPP attacks also enable people to bypass input validation checks and web application firewall (WAF) rules. It opens up routes to attacks, including cross-site scripting (XSS), structured query language (SQL) injection. HPP attacks can be performed by polluting HTTP GET/POST requests by injecting multiple parameters with the same name holding different values and kept apart by delimiters.

Recommendations

The Following are the recommendations that should be implemented.

Use a whitelist format for the application's inputs.

Use web application firewalls for utmost protection.

Encrypt the session cookies to prevent tampering.

If the cookie originated from the client-side, such as a referrer it should not be used to make any security decisions.

Avoid including parameters into the query string.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.6 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.52 ORM Injection		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

ORM injection is a type of vulnerability that occurs in web applications that use object-relational mapping (ORM) frameworks. ORM is a technique that allows developers to work with a database using objects, rather than writing raw SQL queries.

An ORM injection vulnerability occurs when an attacker is able to inject malicious code into an application's SQL queries through the use of an ORM framework. This can allow the attacker to gain unauthorized access to the database, modify or delete data, or perform other malicious actions.

Risk

If an attacker is able to successfully exploit an ORM injection vulnerability, they may be able to gain unauthorized access to the application's database and perform a variety of malicious actions.

Some potential impacts of an ORM injection attack include:

Unauthorized access to sensitive data: An attacker may be able to use an ORM injection vulnerability to gain access to sensitive information, such as financial records or personal information, that is stored in the application's database.

Modification or deletion of data: An attacker may be able to use an ORM injection vulnerability to modify or delete data in the application's database, which could cause serious damage to the integrity of the application.

Denial of service: An attacker may be able to use an ORM injection vulnerability to cause the application to crash or become unresponsive, resulting in a denial of service for legitimate users.

Reputation damage: If an attacker is able to successfully exploit an ORM injection vulnerability, it could damage the reputation of the organization that operates the application, potentially leading to loss of trust from customers and other stakeholders.

Recommendations

It is highly recommended that :-

Validate all user input: It is important to validate all user input to ensure that it is free from malicious code. This can help prevent attackers from injecting malicious code into your application's SQL queries.

Use parameterized queries: Parameterized queries are a type of SQL query that allows you to specify placeholders for values that are supplied at runtime. This can help prevent attackers from injecting malicious code into your application's SQL queries, as the placeholders will not be treated as part of the query.

Use an ORM framework that is designed to prevent SQL injection: Some ORM frameworks include built-in features that are designed to prevent SQL injection, such as automatic parameterization of queries. Using one of these frameworks can help reduce the risk of ORM injection vulnerabilities in your application.

Regularly review and test your application: Regularly reviewing and testing your application can help identify and prevent ORM injection vulnerabilities. This can include conducting regular code reviews, as well as performing penetration testing to identify potential vulnerabilities.

Affected URL & Parameter
Throughout the Application
CVSS Vector
6.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.53 XML Injection		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While testing web application, it came to our notice that the web application is vulnerable to XML Injection. XML injection vulnerabilities arise when user input is inserted into a server-side XML document or SOAP message in an unsafe way. It may be possible to use XML metacharacters to modify the structure of the resulting XML. Depending on the function in which the XML is used, it may be possible to interfere with the application's logic, to perform unauthorized actions or access sensitive data.

Risk

Due to this vulnerability attackers can interfere with the app's logic, gaining the access to the unauthorized parts and stealing the sensitive data

Recommendations

The Following are the recommendations that should be implemented.

Please sanitize user inputs to filter out unacceptable characters. You can do this by escaping or disallowing characters like ', ", <, >, /, etc.

Please Implement a content security policy (CSP).

Affected URL & Parameter

Throughout the Application

CVSS Vector

6.1 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.54 SSI Injection		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Upon testing the web application, we observed that the Server Side Includes Injection was possible. SSIs are directives present on the Web applications. These directories are used to feed an HTML page with dynamic page contents. The SSIs are used to execute some actions before a page is loaded and while the page is being visualised. For performing this action, the web server analyses the SSI before showing the page to the user. There are many web server that permits SSI execution without any proper validation. This vulnerability can lead to an attacker accessing and manipulating the file system of the server. The attacker can then process under the permission of the web server's administrator to completely exploit the system.

Risk

The attacker can perform actions such as reading, updating and deleting arbitrary data/tables from the database. The attacker could also execute commands on the underlying operating system if he's able to exploit this vulnerability.

Recommendations

It is highly recommended that:-

As part of patch management, implement version management for JavaScript libraries.

Remove libraries that are no longer in use to reduce your attack surface.

Frequently check for patches and upgrade JavaScript libraries to the latest version. Disable SSI execution on pages that do not require it.

Pages requiring SSI, only enable the SSI directives that are needed for this page and disable all others.

Encode user supplied data before passing it to a page with SSI execution permissions.

Use SUExec[5] to have the page execute as the owner of the file instead of the web server user.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.55 Buffer overflow		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

An application vulnerability known as a buffer overflow was found in the given web application. Buffer overflow or buffer overrun occurs when a program overruns the buffer's boundaries while sending data to it. The temporary storage space allotted to an application for data storage is called a buffer. The program overwrites the nearby memory addresses. The attacker has the ability to overwrite data on the stack, including the function's return pointer, and send data to the applications. A buffer overflow attack is used by the attacker to damage the victim server's execution stack.

Risk

Using this vulnerability, an attacker can crash the application by overloading the buffering, explicitly execute any malicious code outside the application’s scope or even
execute a denial of service attack.

Recommendations

It is recommended that:-

Try to keep up with the latest bug reports for the web application server products and other products in the Internet infrastructure.

Periodically scan the web application for buffer overflow vulnerability.

Apply the latest patches to all the products in the server.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.56 HTTP Splitting/Smuggling		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

During the penetration testing of the web application, we were able to conduct HTTP response splitting. This occurs when data enters a web application through an untrusted source, most frequently an HTTP request or the data is included in an HTTP response header sent to a web user without being validated for malicious characters. HTTP response splitting is a means to an end, not an end in itself.

Risk

The vulnerability allows the attacker to set arbitrary headers, take control of the body, or break the response into two or more separate responses. Impacts depend on the technological stack, with outcomes including Cross-Site Scripting, Cookie Injection, CORS Headers Injection, CSP (Content Security Policy) Bypass, Cache Poisoning attacks, and many others.

Recommendations

The Following are the recommendations that should be implemented.

Using URL-encoded strings before inclusion into HTTP headers. This include Location or Set-Cookie.

Carefully validate and sanitize any user-provided content that might be used to compose response headers.

Encode dangerous characters such as \r and \n.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.57 Improper Error handling		
CVSS Risk Rating	Medium	Status	Open

Observation Details

Improper error handling refers to the failure to properly manage and handle errors that occur during the execution of a program. This can include not properly logging or reporting errors, not providing adequate feedback to the user, or not properly securing error messages.

Risk

If errors are not properly handled, they can potentially be exploited by attackers to gain unauthorized access to sensitive information or to perform other malicious actions.

Some potential impacts of improper error handling include:

Unauthorized access to sensitive information: If an attacker is able to view or exploit error messages, they may be able to gain access to sensitive information, such as passwords or financial data, that is stored in the program's database.

Reputation damage: If an attacker is able to successfully exploit improper error handling, it could damage the reputation of the organization that operates the program, potentially leading to loss of trust from customers and other stakeholders.

Legal and regulatory consequences: Depending on the nature and severity of the attack, the organization may be subject to penalties or fines from regulatory authorities for failing to properly handle errors in their program.

Recommendations

It is recommended that it is important to properly handle errors that occur during its execution. Some recommended steps for implementing effective error handling include:

Properly log and report errors: It is important to log and report errors in a way that allows developers to quickly identify and fix the problem. This can include providing detailed information about the error, such as the location and cause of the error, as well as any relevant context.

Provide adequate feedback to the user: When an error occurs, it is important to provide clear and concise feedback to the user about what has happened and what steps they can take to resolve the issue. This can help prevent confusion and frustration on the part of the user.

Properly secure error messages: It is important to properly secure error messages to prevent attackers from being able to view or exploit them. This can include not displaying sensitive information in error messages, and properly configuring error handling to prevent stack traces from being visible to unauthorized users.

Regularly review and test your program: It is important to regularly review and test your program to identify and prevent errors, as well as vulnerabilities that may be related to error handling. This can include conducting regular code reviews and performing penetration testing to identify potential weaknesses.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

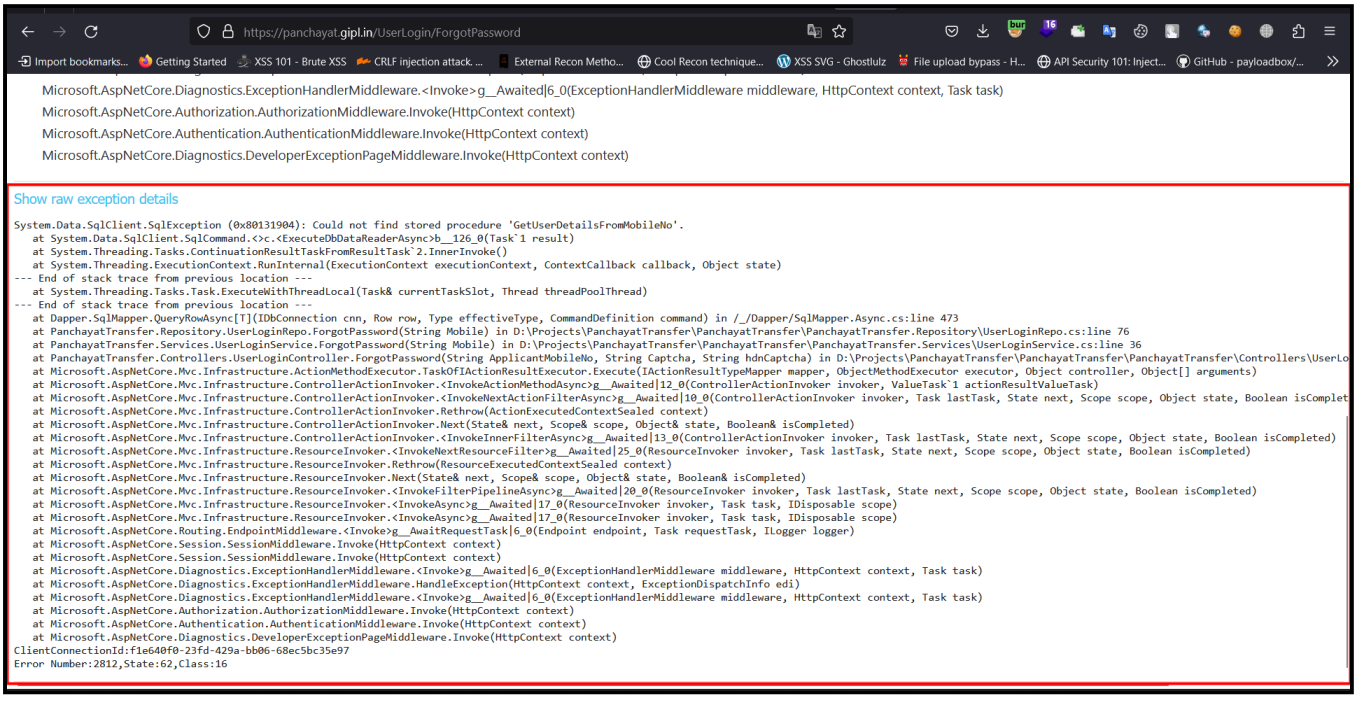


Figure 14 - As you can see that after submitting the mobile number in forgot password, application is giving code errors.

Observation ID & Title	8.58 Padding Oracle		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

A padding oracle attack is a type of cryptographic attack that exploits vulnerabilities in the way that some web applications implement encryption. In a padding oracle attack, an attacker is able to gain access to sensitive information by sending carefully crafted requests to the web application and observing the responses.

Risk

If an attacker is able to successfully exploit a padding oracle vulnerability, they may be able to gain access to sensitive information, such as passwords or financial data, that is stored in the application's database.

In addition to exposing sensitive information, a padding oracle attack can also damage the reputation of the organization that operates the web application. If an attacker is able to successfully exploit a padding oracle vulnerability, it could lead to loss of trust from customers and other stakeholders, which could have significant financial implications.

Furthermore, a padding oracle attack could also lead to legal and regulatory consequences for the organization. Depending on the nature and severity of the attack, the organization may be subject to penalties or fines from regulatory authorities.

Overall, padding oracle attacks can have serious consequences for the security and integrity of a web application, as well as for the reputation and financial health of the organization that operates the application. It is important to take steps to protect against these attacks and prevent them from happening.

Recommendations

To protect against padding oracle attacks, it is important to follow best practices when implementing encryption in your web application. Some recommended steps for preventing padding oracle attacks include:

Use secure encryption algorithms: When encrypting data in your web application, it is important to use secure algorithms that have been proven to be resistant to attack. This includes algorithms such as AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman).

Properly implement padding schemes: When using padding schemes, it is important to implement them properly in order to prevent attackers from being able to exploit vulnerabilities. This includes using secure padding schemes such as PKCS#7 padding and avoiding schemes that are known to be vulnerable to attack.

Regularly review and test your application: It is important to regularly review and test your web application to identify and prevent vulnerabilities, including padding oracle vulnerabilities. This can include conducting regular code reviews and performing penetration testing to identify potential weaknesses.

Use secure communication protocols: When transmitting encrypted data between the client and server, it is important to use secure communication protocols such as HTTPS (Hypertext Transfer Protocol Secure). This can help prevent attackers from being able to intercept and view encrypted data in transit.

Affected URL & Parameter
Throughout the Application
CVSS Vector
5.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.59 Business Logic Data Validation		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Business logic data validation is the process of ensuring that the data that is input into a web application adheres to the rules and constraints defined by the business logic of the application. This is an important aspect of web application security, as it helps prevent attackers from injecting malicious data into the application.

Risk

Failing to properly validate data according to the business logic of a web application can have serious consequences for the security and integrity of the application. Some potential impacts of inadequate business logic data validation include:

Unauthorized access to sensitive information: If an attacker is able to inject malicious data into an application, they may be able to gain access to sensitive information, such as financial records or personal information, that is stored in the application's database.

Modification or deletion of data: An attacker may be able to use malicious data to modify or delete data in the application's database, which could cause serious damage to the integrity of the application.

Denial of service: If an attacker is able to inject malicious data into an application, it could cause the application to crash or become unresponsive, resulting in a denial of service for legitimate users.

Reputation damage: If an attacker is able to successfully exploit a lack of business logic data validation, it could damage the reputation of the organization that operates the application, potentially leading to loss of trust from customers and other stakeholders.

Recommendations

It is recommended that To prevent attacks and protect the security and integrity of a web application, it is important to properly validate data according to the business logic of the application. Some recommended steps for implementing effective business logic data validation include:

Define the rules and constraints of the business logic: Before implementing business logic data validation, it is important to clearly define the rules and constraints that data must adhere to in order to be considered valid. This should be done in collaboration with the business stakeholders of the application, and should take into account the specific needs and requirements of the application.

Implement server-side validation: Business logic data validation should be implemented on the server-side of the application, as this is the most secure place to perform this type of validation. This can be done using a combination of programming languages and frameworks, such as Java and Spring MVC.

Use parameterized queries: When working with databases, it is important to use parameterized queries to prevent SQL injection attacks. This can help ensure that any data that is input into the application is treated as a value, rather than part of the query itself.

Regularly review and test the validation process: It is important to regularly review and test the business logic data validation process to ensure that it is working properly and is able to effectively prevent malicious data from being input into the application. This can include conducting regular code reviews and performing penetration testing to identify potential vulnerabilities.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.60 Number of Times a Function Can be Used Limits		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Many of the problems that applications are solving require limits to the number of times a function can be used or action can be executed. Applications must be "smart enough" to not allow the user to exceed their limit on the use of these functions since in many cases each time the function is used the user may gain some type of benefit that must be accounted for to properly compensate the owner.

Risk

It allows an attackers to circumvent the business logic and execute a function more times than “allowable” exploiting the application for personal gain.

Recommendations

The Following are the recommendations that should be implemented.

Please ensure that the business logic is being followed and that if a function/action can only be executed a certain number of times, when the limit is reached the user can no longer execute the function.

To prevent users from using a function over the appropriate number of times the application may use mechanisms such as cookies to keep count or through sessions not allowing users to access to execute the function additional times

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.61 Upload of Malicious Files		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

The upload of malicious files, also known as file upload vulnerabilities, is a type of security weakness that allows attackers to upload malicious files onto a system or website. This can be accomplished by exploiting vulnerabilities in the file upload process, or by tricking users into uploading malicious files themselves.

Risk

Once a malicious file has been uploaded, it can be used to gain unauthorized access to sensitive information, disrupt the operation of the system or website, or spread malware to other users. This can have significant negative impacts, such as data breaches, financial losses, and reputational damage.

Recommendations

To mitigate the risks associated with file upload vulnerabilities, it is recommended to:-

Implement strict controls and validation checks on the types of files that can be uploaded to a system or website. This can include limiting the file types and sizes that are allowed, as well as scanning uploaded files for known malware signatures.

Implement security measures such as firewalls, intrusion detection and prevention systems, and antivirus software can help protect against the upload of malicious files.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.62 Vulnerable JavaScript Library		
CVSS Risk Rating	Medium	Status	Open

Observation Details

jQuery, Angular, Vue, and React are a few well-known Javascript libraries. An unpatched JavaScript library can leave a website significantly open to numerous types of attacks. A number of DOM-based vulnerabilities, such as DOM-XSS, can be drawn by third-party JavaScript modules and used to steal user accounts. The given web application was found to be using a vulnerable JavaScript library as well, which immediately should be remediated.

Risk

JavaScript library's security vulnerabilities can be exploited to perform cross-site scripting, cross-site request forgery, and buffer overflow attack.

Recommendations

It is highly recommended that:-

As part of patch management, implement version management for JavaScript libraries.

Remove libraries that are no longer in use to reduce your attack surface.

Frequently check for patches and upgrade JavaScript libraries to the latest version.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

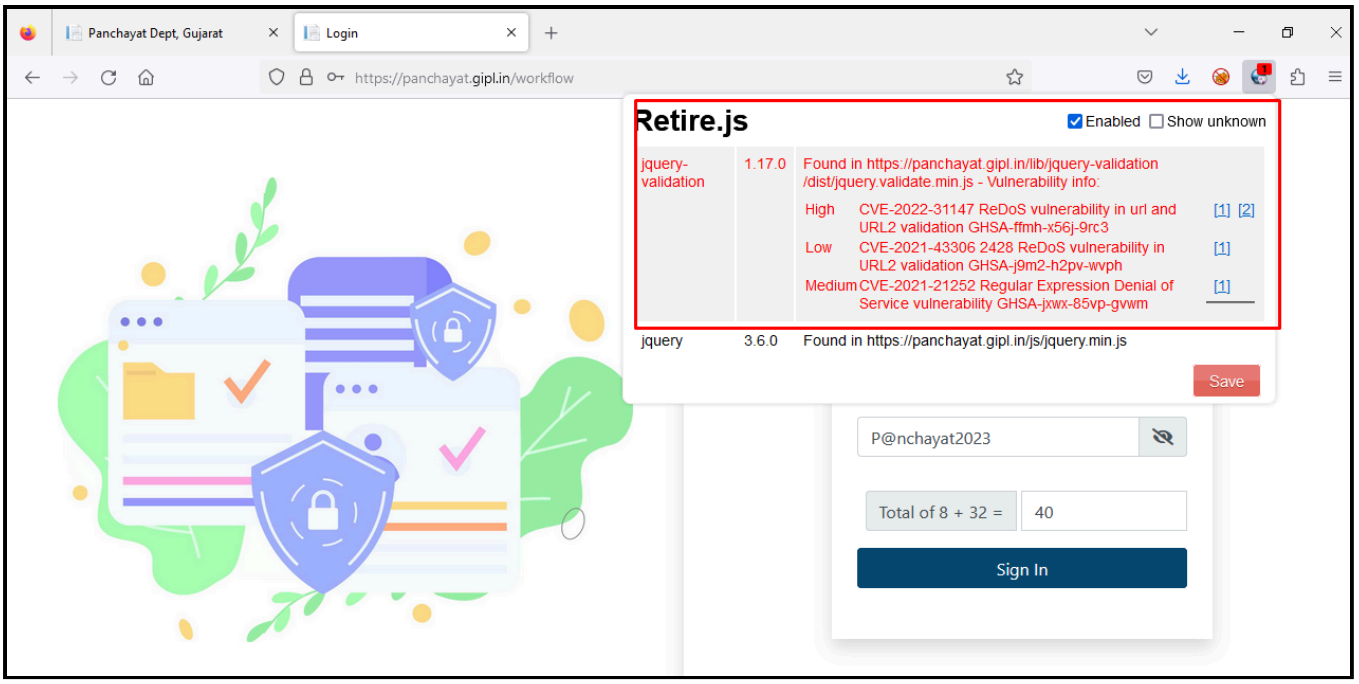


Figure 15 - As you can see that the application is vulnerable to old Jquery version.

Observation ID & Title	8.63 WebSockets hijacking		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

A WebSocket is a bidirectional, full-duplex protocol that is used in a client-server communication. It is a stateful protocol, which means the connection between client and server will keep alive until it is terminated by either party (client or server). WebSocket hijacking is mainly done through Cross Site Request Forgery. Cross-site WebSocket hijacking (also known as cross-origin WebSocket hijacking) involves a cross-site request forgery (CSRF) vulnerability on a WebSocket handshake. It arises when the WebSocket handshake request relies solely on HTTP cookies for session handling and does not contain any CSRF tokens or other unpredictable values. During the testing phase of the web application, this vulnerability was found to be in the web application communication protocol.

Risk

An attacker can perform unauthorized actions masquerading as the victim user. As with regular CSRF, the attacker can send arbitrary messages to the server-side application. If the application uses client-generated WebSocket messages to perform any sensitive actions, then the attacker can generate suitable messages cross-domain and trigger those actions.
The attacker would also be able to retrieve sensitive data that the user can access. Unlike with regular CSRF, cross-site WebSocket hijacking gives the attacker two-way interaction with the vulnerable application over the hijacked WebSocket. If the application uses server-generated WebSocket messages to return any sensitive data to the user, then the attacker can intercept those messages and capture the victim user's data.

Recommendations

It is recommended to:-

Use individual random tokens (e.g. CSRF tokens) in the handshake request and validate them against the server.

Check the Origin header of the WebSocket handshake request on the server.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.64 Local Storage/Session Storage		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While testing the application we found that the application is vulnerable to Local Storage/Session Storage vulnerability. A Local Storage/Session Storage vulnerability occurs when a website stores sensitive data, such as user credentials or other sensitive information, in Local Storage or Session Storage. This data is stored locally on the user's computer and is accessible to any website that the user visits. If an attacker is able to access this data, they could potentially use it to gain unauthorized access to the user's accounts or perform other malicious actions.

Risk

Local Storage/Session Storage vulnerability allow an attacker to gain unauthorized access to user's account and his sensitive information and potentially cause damage or loss.

Recommendations

The Following are the recommendations that should be implemented.

Applications should be storing sensitive data on the server side, and not on the client-side, in a secured manner following best practices.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.65 Race Condition		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

During our testing we found out that web application is vulnerable to Race Condition vulnerability. A race condition is a type of security vulnerability that occurs when two or more processes or threads try to access a shared resource at the same time. This can lead to unpredictable and potentially harmful behavior, such as data corruption or incorrect results.

Risk

The impact of an OAuth misconfiguration can be significant, depending on the nature of the misconfiguration and the data or resources that are accessed by the attacker. In the worst case, it could allow an attacker to gain access to sensitive information, such as user passwords or financial data, or to execute arbitrary code on the server.

Recommendations

The Following are the recommendations that should be implemented.

Using proper synchronization techniques, such as locks or semaphores, to ensure that only one process or thread can access a shared resource at a time.

Regularly monitoring and testing your system to detect and address potential race condition vulnerabilities.

Implementing security measures, such as access control lists or authentication protocols, to prevent unauthorized access to shared resources

Affected URL & Parameter

Throughout the Application

CVSS Vector

6.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.66 OAuth Misconfiguration		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

OAuth is a widely-used standard for authorization, which allows users to grant third-party applications access to their data or resources without sharing their login credentials. A misconfiguration in an OAuth implementation can potentially allow an attacker to gain unauthorized access to a user's data or resources.

Risk

The impact of an OAuth misconfiguration can be significant, depending on the nature of the misconfiguration and the data or resources that are accessed by the attacker. In the worst case, it could allow an attacker to gain access to sensitive information, such as user passwords or financial data, or to execute arbitrary code on the server.

Recommendations

To mitigate the risks associated with OAuth misconfigurations, it is recommended to:-

Properly configure and secure the OAuth implementation. This can include implementing appropriate authentication and authorization controls, using secure cryptographic algorithms and keys, and regularly updating and rotating the keys.

Regular security assessments and testing can also help identify and address potential vulnerabilities in a timely manner.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.67 Path Traversal		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While testing the web application we found out that this application is vulnerable to Path Traversal vulnerability. Path Traversal vulnerability allows attackers to access files and directories that are stored outside the web root folder. Path traversal vulnerabilities happen when a malicious user can include an arbitrary file path in user input and use special characters to access files from a different directory on the server. Special characters used for this are dot-dot-slash combinations: ../ for Linux/UNIX or ..\ for Windows. These combinations allow access to parent directories from a relative path.

Risk

Path traversal vulnerability lets an attacker to access and view files located in the web server file system but outside of the web application's document root folder as well which can cause sensitive information disclosure. Attackers can then use the information from these files to find other appsec attack methods to increase attack vector.

Recommendations

You can mitigate Path traversal attacks by running your web application in a limited environment.

Make sure the application always validates used-supplied input before processing it.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.68 Weak 2FA Implementation		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While conducting the testing of the web application, we found that the application has weak 2FA implementation. Two-factor authentication (2FA) reinforces access security by demanding two methods to verify a user's identity. It does not replace passwords but simply adds a second layer of security in addition to passwords. There are many methods of two factor authentication available, all of which have a common goal: make it harder for a cybercriminal to get access to an account. The main methods through which two factor authentication can be enabled are SMS, email, and authenticator apps.

Risk

If Two Factor Authentication is not implemented properly, there might be a chance that the attacker would be able to bypass this authentication, and can gain access to the victim account or access to the unauthorized resources or critical data.

Recommendations

The Following are the recommendations that should be implemented.

Server should always issue randomized tokens

Server should have validation check whether the user has provided OTP value in the fields

Server should not reveal any kind of OTP codes in the response

Affected URL & Parameter

Throughout the Application

CVSS Vector

6.4 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.69 Token is Not Invalidated After New Token is Requested		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Upon testing the web application, it was found that the token is not being invalidated after new token is being requested. Token invalidation is the process of making a previously issued token no longer valid. This is typically done to prevent an attacker from using an old or expired token to gain unauthorized access to a web application or system. Invalidating a token can help prevent potential security vulnerabilities and ensure that only valid and current tokens are used for authentication and access.

Token invalidation is often performed when a user logs out of the application, when a new token is requested, or when a token is expired or otherwise no longer valid. This helps to prevent attackers from using old or expired tokens to gain unauthorized access to the application.

Risk

If a token is not invalidated after a new token is requested, it can potentially allow an attacker to gain unauthorized access to a user's account or perform actions on behalf of the user. This is because the old token remains valid, and an attacker can potentially use it to gain access to the application.

The impact of this vulnerability can be significant, as it can allow an attacker to access sensitive information, perform actions on behalf of the user, and potentially compromise the security of the entire system.

Recommendations

It is highly recommended to:-

Implement and maintain secure token management in your web application.

Invalidate old tokens when a new token is requested.

Properly encrypt and sign tokens.

Regularly rotate and invalidate tokens.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.70 Failure to Invalidate Session on Logout		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While testing web application it came to our notice that the web application is vulnerable to Content Spoofing vulnerability. Content spoofing occurs when an attacker alters the content of a website or application in a way that is intended to deceive or trick the user. This can be done by modifying the HTML or other markup language used to display the content, or by injecting malicious code into the page. It can be used to impersonate legitimate websites or applications, to steal sensitive information, or to spread malware

Risk

Due to Content Spoofing vulnerability attacker can spoofed website or application to trick users, which lead to loss of sensitive information such as password or financial information.

Recommendations

The Following are the recommendations that should be implemented.

Using proper synchronization techniques, such as locks or semaphores, to ensure that only one process or thread can access a shared resource at a time.

Regularly monitoring and testing your system to detect and address potential race condition vulnerabilities.

Implementing security measures, such as access control lists or authentication protocols, to prevent unauthorized access to shared resources

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.71 Missing Security Header		
CVSS Risk Rating	Medium	Status	Open

Observation Details

Upon testing the web application, we found that there were important security headers missing from the web application. Whenever a browser requests a page from any web server, the server responds with the content along with HTTP response headers. These HTTP security headers tell the browser how to behave while handling the website content.

Risk

If security headers are missing in a web application or have not been implemented properly, attackers can conduct various attacks such as cross site scripting, Spectre, data injection, HTTP redirection, downgrade attacks, etc.

Recommendations

The Following are the recommendations that should be implemented.

Apply Webserver Configuration (Apache, Nginx, and HSTS).

Apply X-Frame-Options.

Apply X-XSS-Protection.

Apply X-Content-Type-Options

Apply HTTP Strict-Transport-Security

Apply Same-Site Cookie

Apply Content-Security-Policy

Apply Referrer-Policy

Apply Cache-Control

Apply Access-Control-Allow-Origin

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

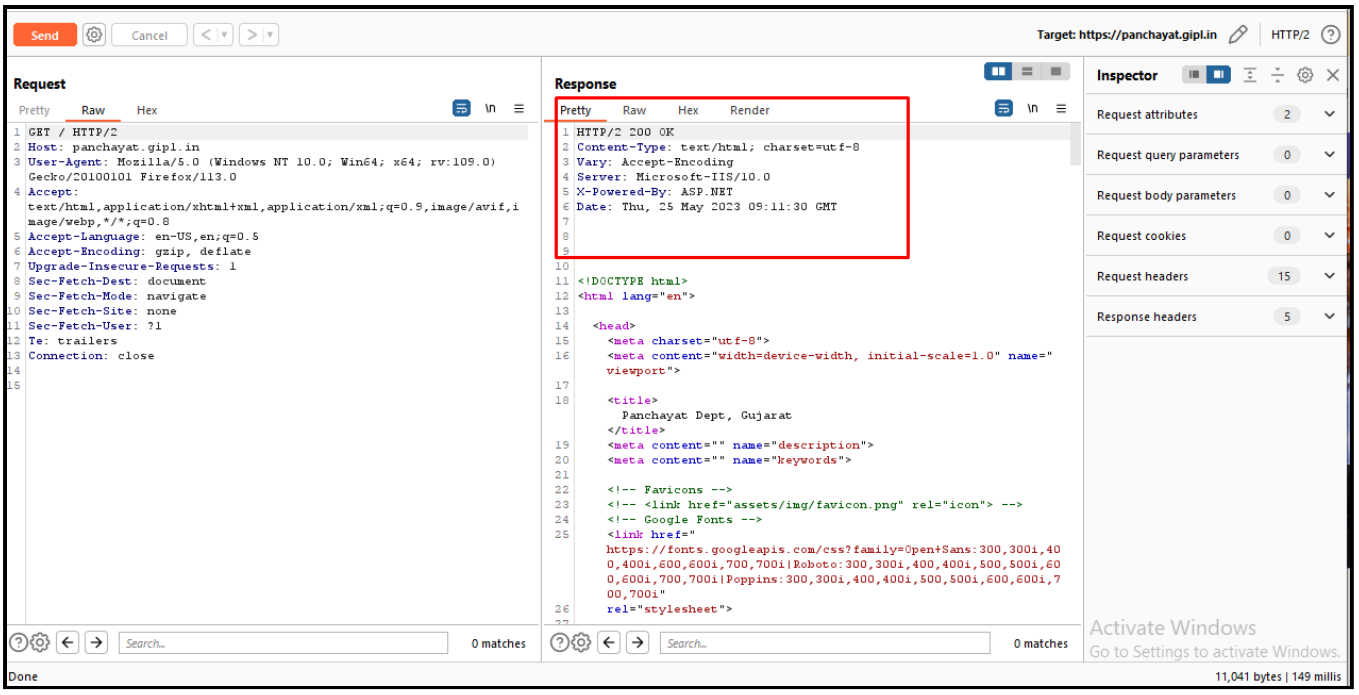


Figure 16 - As you can see that the headers are missing in the application.

Observation ID & Title	8.72 Mail Server Misconfiguration		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

As we tested the application, we noticed that the application is vulnerable to Mail Server Misconfiguration, and it lacks the DMARC record and its policy which can lead to spamming and social engineering any user with the company mail. When someone sends an email using a falsified sender address, this is known as email spoofing. As mail lacks built-in authentication, spammers, phishers, and attackers utilize spoofing to exploit the confidence that the faked domain holds and trick users into disclosing critical information. This occurs due to Mail Server Misconfiguration.

Risk

An attacker can send emails to their clients and customers on the behalf of the company and trick users or the client to click on fake links or fake web address that pretends to be of the company URL. Anyone can distribute fake email content/files using a company's email. As a result, the company's image might be jeopardized.

Recommendations

It is recommended that: -

Publish the DMARC policy.

Fixing vulnerability requires you to add SPF details on your domain as a TXT record. Your hosting or email solution provider will share the SPF details. If you don't have one, you may want to check with your hosting provider.

You must authenticate the DMARC records by putting some of the protocols to prevent the email spoofing.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.73 Server-Side Credentials Storage - Plaintext		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

During the testing of the web application, it was noticed that the server was storing credentials in plaintext. Storing credentials in plaintext is not considered a good practice. Data in both the states, in rest and in transit, should be encrypted. This will prevent the attackers from being able to access any information in plaintext.

Risk

If credentials are stored in plaintext on the server side, an attacker might be able to get hold of these unencrypted credentials and he would be able to gain access to a number of user accounts. The scenario could be worse if the attacker is able to get hold of admin credentials. In that case, he would be able to gain access to critical information.

Recommendations

It is highly recommended to store credentials in encrypted format. Strong encryption algorithms such as AES-256, Triple DES, Blowfish, etc. should be used for encrypting credentials.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.74 SQL Possible		
CVSS Risk Rating	Medium	Status	Open
Observation Details			
As per observation, while resetting the password, application is giving the sql error which is further possible to extract the data if the attacker wants.			
Risk			
Attacker can extract the data of the server files with the help of sql error.			
Recommendations			
The Following are the recommendations that should be implemented. Application should not give such SQL error which can result to the high impact to the organization.			
Affected URL & Parameter			
Throughout the Application			
CVSS Vector			
4.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L			

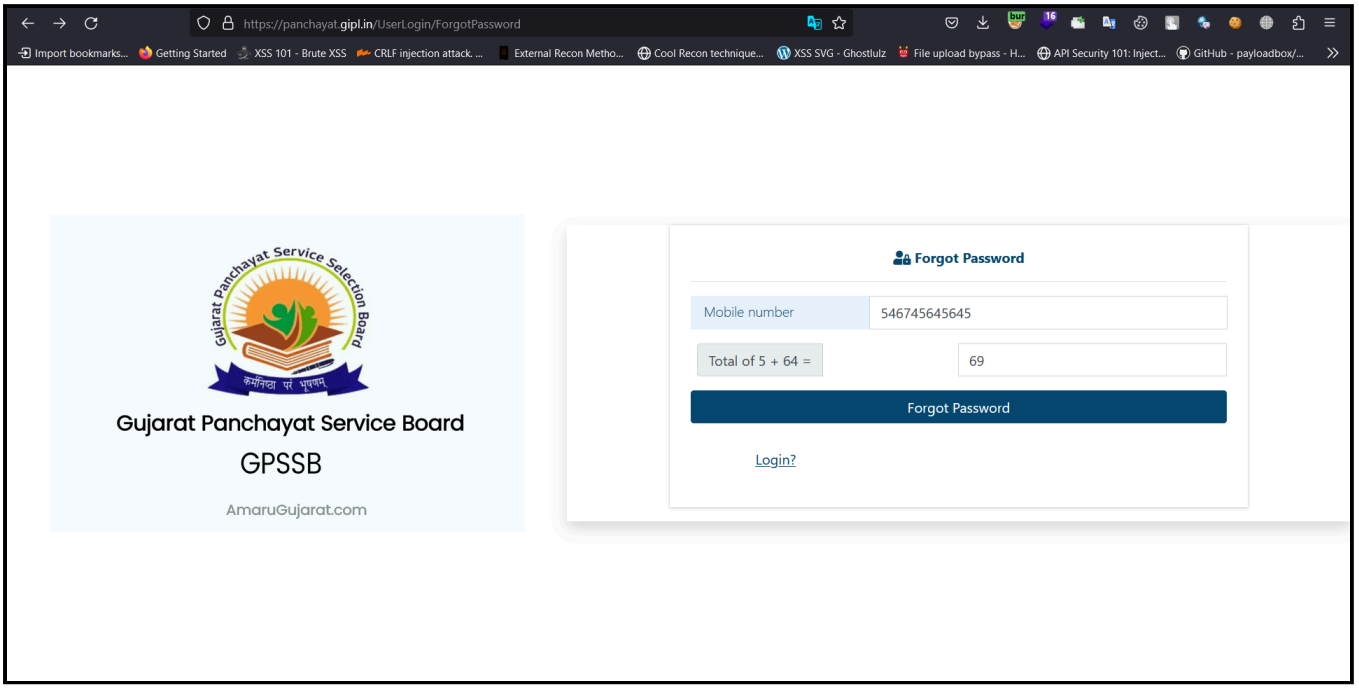


Figure 17 - Fill up the details and click on submit.

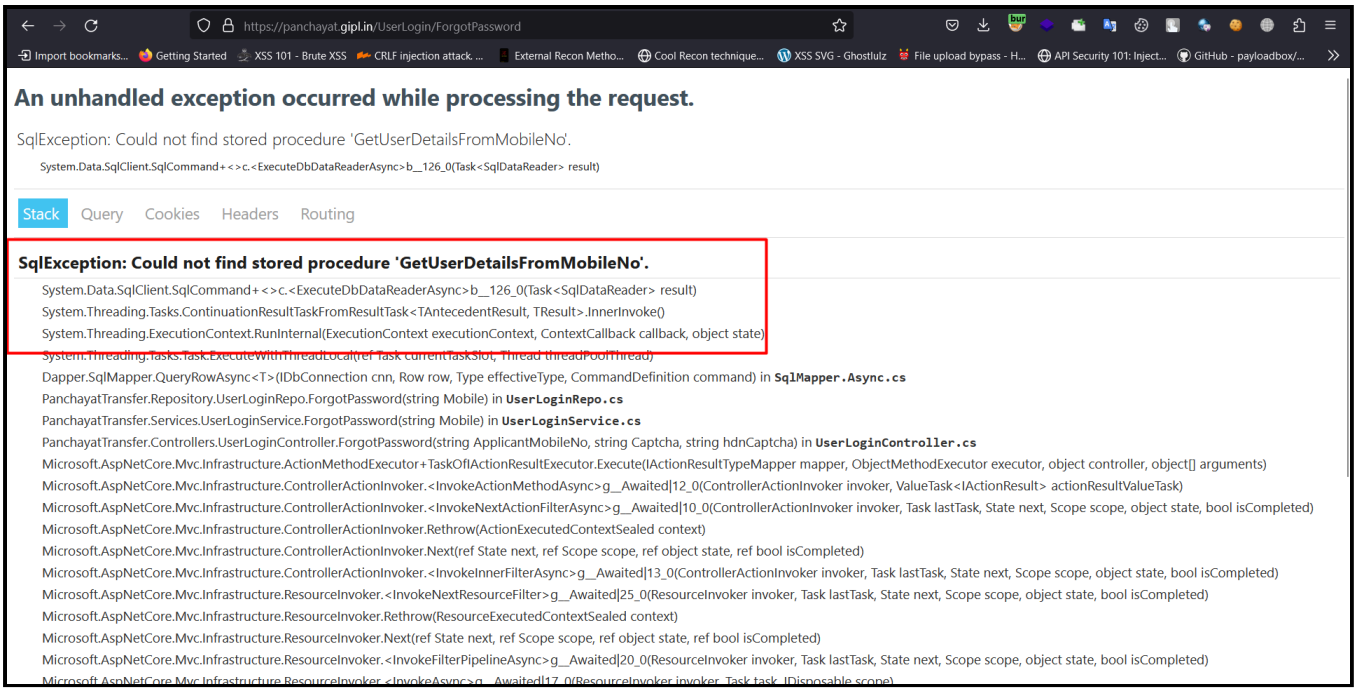


Figure 18 - As you can observe that the application is giving the SQL errors.

Observation ID & Title	8.75 EXIF Geolocation Data Not Stripped From Uploaded Images		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

EXIF geolocation data is metadata embedded in digital images that contains information about the location where the photo was taken. This data is often stored in the form of GPS coordinates, which can be used to pinpoint the exact location of the photo on a map. A vulnerability exists when an image uploading system does not strip this geolocation data from the images that are uploaded to it. This can potentially allow attackers to use the geolocation data to identify the location where the photo was taken, and potentially use that information for malicious purposes.

Risk

Failure to invalidate session is a security vulnerability that allow attackers to continue using the old password to access the user's account, even after the password has been changed which can lead to sensitive information disclosure, unauthorized access to the user's account.

Recommendations

The Following are the recommendations that should be implemented.

It is important to ensure that the image uploading system is properly configured to strip geolocation data from images before they are uploaded.

Implementing security protocols that automatically remove geolocation data from images as they are being uploaded.

Keep systems and software up to date with the latest security patches and updates

Affected URL & Parameter

N/A

CVSS Vector

5.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.76 Failure to Invalidate Session - Password Change		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

A failure to invalidate session after a password change is a security vulnerability that occurs when a user changes their password, but the system does not properly log them out of all active sessions. This can potentially allow attackers to continue using the old password to access the user's account, even after the password has been changed.

Risk

If a session is not properly invalidated after a user changes their password, it can potentially allow an attacker to gain unauthorized access to the user's account. This is because the old password may still be valid, and an attacker can potentially use it to gain access to the application.

The impact of this vulnerability can be significant, as it can allow an attacker to access sensitive information, perform actions on behalf of the user, and potentially compromise the security of the entire system.

Recommendations

It is highly recommended to:

Invalidate the user's session after a password change.

Properly encrypt and sign session tokens.

Regularly rotate and invalidate sessions.

Implement additional security controls.

Affected URL & Parameter

N/A

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.77 Web Application Firewall (WAF) Bypass		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

During the testing of web application we notice that the web application is vulnerable to Web Application Firewall bypass vulnerability. It is a type of security vulnerability that occurs when an attacker is able to bypass the protections of a WAF and gain access to the underlying application. This can be done using a variety of methods, such as exploiting vulnerabilities in the application or using evasion techniques to avoid detection by the WAF.

Risk

Web application Firewall Bypass vulnerability can allow an attacker to gain unauthorized access to an application, potentially leading to security breaches or other forms of damage. In other cases, a WAF bypass can cause an application to crash or become unstable, leading to downtime and lost productivity.

Recommendations

The Following are the recommendations that should be implemented.

Regularly testing and monitoring your WAF to ensure that it is functioning properly and providing adequate protection.

Keeping your WAF and your underlying application up to date with the latest security patches and updates.

Using a WAF that is specifically designed to protect against the types of attacks that are most commonly used to bypass WAFs.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.78 HTTP Response Manipulation		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While conducting the testing for the web application, it came to our notice that the HTTP response could be manipulated. HTTP Response manipulation is a type of attack in which an attacker modifies the data sent in an HTTP response in order to gain unauthorized access to a website or web application. This can be done by altering the data sent in the response, or by injecting malicious code into the response.

Risk

If an organization is successfully targeted by an HTTP response manipulation attack, the impact can be significant. The attacker may be able to gain unauthorized access to user accounts, steal sensitive information, or perform unauthorized actions on the user's behalf. This can result in loss of confidentiality, integrity, and availability of the organization's data, as well as damage to its reputation and financial losses.

Recommendations

The Following are the recommendations that should be implemented.

Use encryption for data transmission, such as Secure Sockets Layer (SSL) or Transport Layer Security (TLS), to protect against response manipulation attacks.

Implement robust authentication mechanisms, such as multi-factor authentication.

Regularly monitor systems for signs of malicious activity, such as unusual login attempts or unauthorized access to sensitive data, and respond quickly to any security incidents.

Use web application firewalls (WAFs) to detect and block response manipulation attacks.

Educate employees about the risks associated with response manipulation attacks and the importance of following best practices for online security.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.3 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.79 TLS Fallback SCSV Support		
CVSS Risk Rating	Medium	Status	Open

Observation Details

The TLS Signalling Cipher Suite Value (SCSV) protects against TLS/SSL downgrade attack such as PODDLE. If enables the server, ensure that the strongest protocol that both client and server understand is used. It has been observed that the web application is not using or not enabled the TLS_FALLBACK_SCSV Support.

Risk

Attackers with the network level access can attempt downgrade attacks; where the attacker forces a client to negotiate to a weaker or known vulnerable encryption scheme. This encrypted session can then be later broken or decrypted by the malicious user.

Recommendations

The Following are the recommendations that should be implemented.

Ensure that all web servers and client devices support the TLS_FALLBACK_SCSV cipher suite. This will prevent attackers from being able to perform a downgrade attack and intercept sensitive information.

Regularly update the web servers and client devices to the latest versions of the TLS protocol. This will ensure that they are able to support the latest security features and protections against attacks.

OpenSSL 1.0.1 users should upgrade to 1.0.1j
OpenSSL 1.0.0 users should upgrade to 1.0.0o
OpenSSL 0.9.8 users should upgrade to 0.9.8zc

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

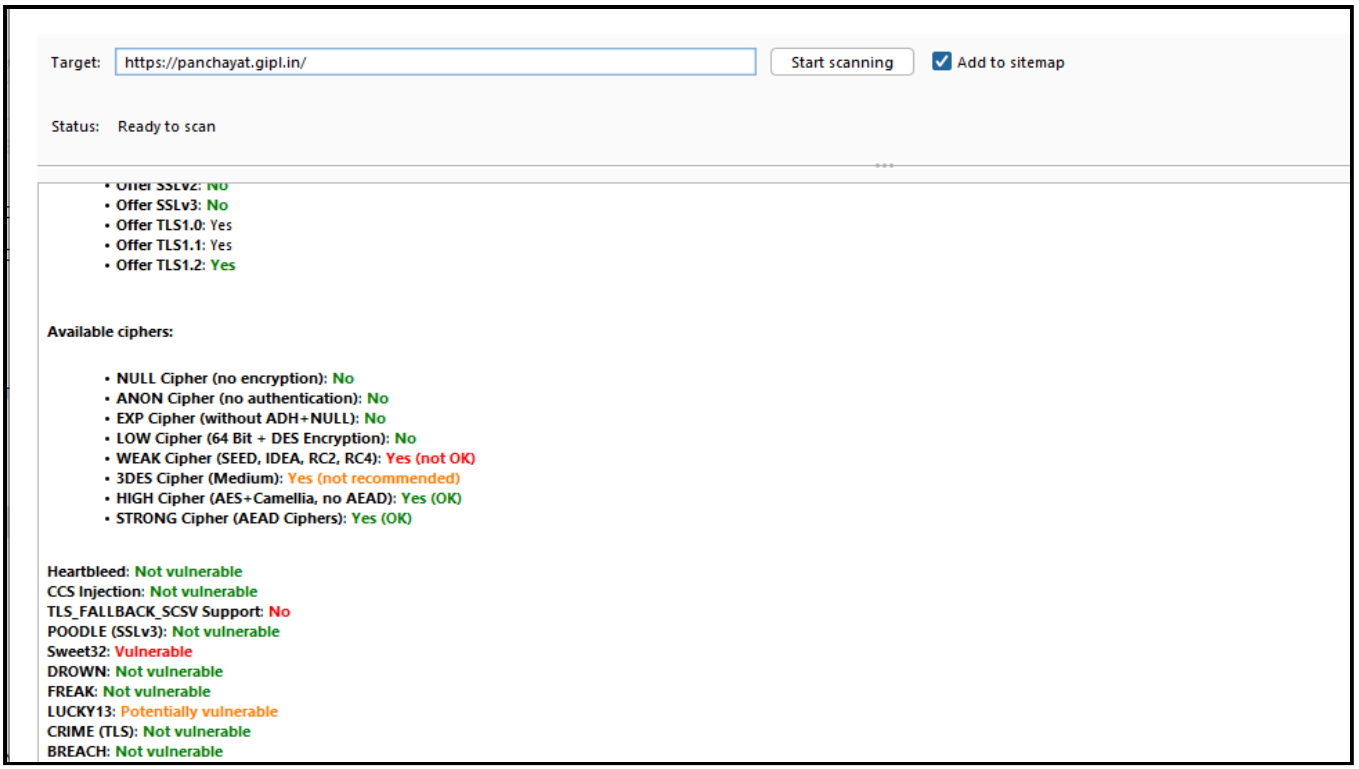


Figure 19 - As you can see that the TLS fallback is not supported.

Observation ID & Title	8.80 Broken Access Control		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While conducting the penetration testing of the web application, we found out that the application has broken access control. Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

Risk

This vulnerability might allow an attacker to change the primary key to another user's record and permitting viewing or editing someone else's account. It also might let the attacker to perform privilege escalation.
The attacker might also be able to bypass access control checks by modifying the URL, HTML page or using a custom API attack tool.

Recommendations

The Following are the recommendations that should be implemented.

Use a central application component to verify access control.

Drive all the access control decisions from a lower privileged user's session.

Implement least privilege policy throughout the application.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.8 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.81 Information Disclosure through XMLRPC		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While conducting the penetration testing of the web application, it was found that the application was disclosing information through XML-RPC. XML-RPC (eXtensible Markup Language - Remote Procedure Call) is a remote procedure call (RPC) protocol that uses XML to encode its calls and HTTP as a transport mechanism. XML-RPC is used to call methods on remote systems over the internet, and it is often used in web services and other types of applications that require remote communication.

Risk

This vulnerability can have serious consequences for both the system hosting the XML-RPC server and its users. If sensitive information is disclosed, it can lead to loss of user trust and damage to the system's reputation. Additionally, it can expose users to financial loss or identity theft if their sensitive information is accessed and used by unauthorized parties.

Furthermore, information disclosure can also make the system hosting the XML-RPC server more vulnerable to other security threats. If an attacker is able to access login credentials through information disclosure, they could potentially gain unauthorized access to the system and its data. This could result in further security breaches, data loss, or other damage to the system.

Recommendations

To mitigate this type of vulnerability, it's important to properly configure XML-RPC servers to avoid disclosing sensitive information. This may involve implementing encryption and other security measures to protect the information being transmitted over the network. Additionally, it's important to regularly review and update the security measures in place to ensure that they are effective in preventing information disclosure.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.82 Nginx Rate Filtering Shaping Overflow		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Nginx is a popular open-source web server and reverse proxy. A rate filtering and shaping overflow vulnerability in Nginx could allow an attacker to send a large number of requests to the server, potentially overwhelming it and causing it to crash or become unresponsive. This type of attack is known as a Denial of Service (DoS) attack.

Risk

The impact of a DoS attack against Nginx could be significant, depending on the nature and scale of the attack. In the worst case, it could cause the server to become unavailable, disrupting service for legitimate users and potentially leading to lost revenue or other negative consequences.

Recommendations

To mitigate the risks associated with this type of vulnerability, it is recommended to:-

Keep Nginx up to date with the latest security patches.

Implement appropriate rate limiting and filtering controls can help prevent attackers from overwhelming the server with a large number of requests.

Regular security assessments and monitoring can also help identify and address potential vulnerabilities in a timely manner.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.83 JWT Token Exploit Possible		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

JSON Web Tokens (JWTs) are a commonly used method for authenticating users and transmitting information securely over the internet. A JWT token exploit involves an attacker using a JWT token to gain unauthorized access to a system or to perform unauthorized actions.

Risk

The impact of a JWT token exploit can be significant, depending on the nature of the exploit and the data that is accessed or modified. In the worst case, it could allow an attacker to gain access to sensitive information, such as user passwords or financial data, or to execute arbitrary code on the server.

Recommendations

To mitigate the risks associated with JWT token exploits, it is recommended to:-

Implement appropriate authentication and authorization controls.

Properly validate and verify JWT tokens before allowing access to protected resources.

Use secure cryptographic algorithms and keys, and properly encrypting and signing JWT tokens, can help prevent attackers from being able to manipulate or forge JWT tokens.

Affected URL & Parameter

N/A

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.84 GraphQL Injection		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

GraphQL is a query language for APIs that is used to fetch data from servers. A GraphQL injection vulnerability occurs when an attacker is able to inject malicious queries into a GraphQL endpoint, potentially allowing them to access sensitive data or execute arbitrary code on the server.

Risk

The impact of a GraphQL injection vulnerability can be significant, depending on the nature of the injected query and the data that is exposed. In the worst case, it could allow an attacker to gain access to sensitive information, such as user passwords or financial data, or to execute arbitrary code on the server.

Recommendations

To mitigate the risks associated with GraphQL injection, it is important to properly validate and sanitize any user-provided input that is used in GraphQL queries.

Implement appropriate authentication and authorization controls can help prevent unauthorized access to sensitive data.

Affected URL & Parameter

N/A

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.85 PHP Object Injection		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

PHP object injection is a type of vulnerability that occurs when an attacker is able to inject a malicious object into a PHP script. This can be used to modify the behavior of the script, potentially allowing the attacker to execute arbitrary code on the underlying system.

Risk

The impact of this vulnerability can be significant, as it allows an attacker to gain complete control over the vulnerable system. This can be used to steal sensitive information, perform actions on behalf of the user, and potentially compromise the security of the entire system.

Recommendations

It is highly recommended to:-

Avoid using exec(), shell_exec(), system() or passthru()

Avoid using strip_tags() for sanitisation.

Code serialization.

Use a PHP security linter.

Utilize a SAST tool to identify code injection issues.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.4 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.86 PHP Include And Post Exploitation		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

As we conducted the testing of the web application, it came to our notice that the PHP include was exploitable. PHP include is a way for one PHP file to include code from another file. This can be useful for modularizing and organizing your code, but it can also be a security vulnerability if not used carefully.

Risk

In terms of exploitation, an attacker could potentially include malicious code in a file that is intended to be included by another file. For example, if an attacker can control the name of the file that is being included, they could potentially include their own code instead of the intended file. This could allow the attacker to execute arbitrary code on the server, potentially allowing them to gain access to sensitive information or compromise the server.

Recommendations

To mitigate this type of attack, it is important to properly validate and sanitize any user-provided input that is used to specify the file to be included. This can help prevent attackers from being able to manipulate the include statement to include their own code. Additionally, it is a good idea to use full, absolute paths for included files, rather than relative paths, to ensure that the correct file is always included.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.6 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.87 JSON Web Encryption		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

JSON Web Tokens (JWT) are a popular way to securely transmit information between parties. They are typically used to authenticate users and authorize access to certain resources. However, like any other security measure, JWTs are subject to various types of attacks.

One type of attack that can be used against JWTs is known as a "JSON token attack." This is a technique where an attacker obtains a JWT and modifies its contents in order to gain unauthorized access to protected resources. The attacker can do this by altering the data contained in the JWT, such as the user's identity or the permissions granted to them.

Risk

The impact of a JSON token attack can be significant, as it allows the attacker to gain access to resources that they would not normally be able to access. This can result in the unauthorized disclosure of sensitive information, or the manipulation of data in ways that can cause harm to the system or its users.

Recommendations

It is highly recommended to:-

Implement and maintain secure token and cookie validation in your web application.

Use secure and unique tokens, properly encrypting and signing cookies, and regularly rotating and invalidating tokens and cookies.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.88 Length Extension Attack		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

A length extension attack is a type of attack against cryptographic hash functions. It exploits the fact that many hash functions process data in blocks, with a fixed block size. This means that if an attacker knows the length of the input to a hash function, they can create a new input that has the same length and produces the same hash as the original input.

Risk

The impact of a length extension attack is that it allows an attacker to forge messages without knowing the original message or its hash. This can be used to impersonate someone else, or to create messages that are not authentic.

Recommendations

To mitigate length extension attacks, it is important to use hash functions that are not vulnerable to this type of attack such as SHA-2 and SHA-3 families of hash functions.

Use a "salt" value when generating hashes, which is a random value that is added to the input to make it more difficult for an attacker to forge a message.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.89 OAuth2: State Fixation		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Upon testing the web application, we came across a state fixation vulnerability in the OAuth framework. OAuth is a commonly used authorization framework that enables websites and web applications to request limited access to a user's account on another application. Crucially, OAuth allows the user to grant this access without exposing their login credentials to the requesting application. This means users can fine-tune which data they want to share rather than having to hand over full control of their account to a third party.

The basic OAuth process is widely used to integrate third-party functionality that requires access to certain data from a user's account. The same mechanism is also used to provide third-party authentication services, allowing users to log in with an account that they have with a different website.

Risk

The attacker can use the state parameter to encode an application state that will put the user where they were before the authentication process started. For example, if a user intends to access a protected page in your application, and that action triggers the request to authenticate, the attacker can store that URL to redirect the user back to their intended page after the authentication finishes. This might help the attacker conduct Cross Site Request Forgery (CSRF) or account takeover as the state parameter in the OAuth prevents attacks such as CSRF and if the state itself gets fixed by the attacker, the application would be vulnerable CSRF.

Recommendations

The Following are the recommendations that should be implemented.

Make sure that adding a new social connection requires a valid csrf_token, so it is not possible to trigger process with CSRF. Ideally, use POST instead of GET.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.90 SAML: Comment Injection		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Upon testing the web application, it came to our notice that the web application is vulnerable to comment injection. Comments injected into an application through input can be used to compromise a system. As data is parsed, an injected/malformed comment may cause the process to take unexpected actions that result in an attack.

Risk

The Following are the recommendations that should be implemented.

If an application is vulnerable to comment injection, it could have an effect on data integrity and confidentiality. Additionally, it may have an impact on the application's authentication and authorization features. Depending on the backend database configuration, access privileges and the web server, an intruder might read, edit, or even delete the information being stored in the database.

Recommendations

It is highly recommended to:-

Use prepared statements (with parameterized queries).

Use stored procedures.

Apply input validation and sanitization.

Apply user-supplied input escaping mechanism.

Ensure that all SAML providers/consumers do proper input validation.

Affected URL & Parameter

Throughout the Application

CVSS Vector

6.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.91 Development Configuration File Leakage		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

A development configuration file contains sensitive information, such as database passwords and API keys, that is used during the development process of a website or application. If these files are not properly secured, they can be accidentally leaked, which can have serious consequences. The development configuration file in this web application was found to be not properly secured.

Risk

If a development configuration file is accidentally leaked, it can have serious consequences. An attacker who gains access to this information could use it to compromise the security of your website or application, or steal sensitive data. They could also use the information to impersonate your organization and carry out other malicious activities.

Recommendations

It is highly recommended to:-

Use version control systems: Use a version control system (such as Git) to manage your configuration files. This will help to ensure that only authorized users have access to these files, and that any changes to the files are tracked and can be easily rolled back if necessary.

Do not store sensitive information in development configuration files: Avoid storing sensitive information, such as passwords and API keys, in development configuration files. Instead, use environment variables or encrypted secrets management systems to store this information.

Use separate development and production environments: Use separate development and production environments for your website or application. This will help to ensure that development configuration files are not accidentally deployed to production, where they could be accessed by unauthorized users.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.92 Null Byte Injection		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

While testing the web application, we found out that the web application is vulnerable to Null Byte Injection vulnerability. Null byte injection is an active exploitation technique that adds URL-encoded null byte characters (%00, or 0x00 in hex) to the user-supplied data in order to get over sanity checking filters in online infrastructure. The application's intended logic can be changed by the injection process, which also gives an attacker access to the system files without authorization.

Risk

An injected NULL character or null byte during data parsing could lead to the software believing the input was terminated sooner than it actually was, or it could cause the input to be misread in various ways. This also might be used to evade validation checks and other security measures or insert potentially harmful input that comes after the null byte.

Recommendations

The Following are the recommendations that should be implemented.

Apply input sanitization.
Use an appropriate combination of blacklists and whitelists to ensure only valid, expected and appropriate input is processed by the system.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.93 XQUERY Injection		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

It was observed during the testing phase of the web application that the application was vulnerable to XQuery Injection. XQuery Injection uses improperly validated data that is passed to XQuery commands. This in turn will execute commands on behalf of the attacker that the XQuery routines have access to. XQuery injection can be used to enumerate elements on the victim's environment, inject commands to the local host, or execute queries to remote files and data sources. Like SQL injection attacks, the attacker tunnels through the application entry point to target the resource access layer. XQuery Injection is a variant of the classic SQL injection attack against the XML XQuery Language.

Risk

XQuery injection attacks are dangerous as they allow attackers to bypass login and account security. It also allows it to be done in an automated fashion using a standard language that does not vary by application. Attackers can automatically scan websites and applications for this vulnerability and act as soon as it's discovered. Beyond compromising account security, XQuery attacks can also be used for data exfiltration. For example, an attacker could transfer all records out of the XML database.

Recommendations

To mitigate this vulnerability from getting exploitedIt is recommended that
Apply input validation and sanitization.

Implement whitelisting inputs.

Use a parameterized XPath interface, similar to prepared statements for SQL queries.

Be sure to apply least privilege to all applications. That might mean creating a user with read-only privileges to perform all application queries.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.94 Origin Bypass		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Upon testing the web application, it came to our notice that we were able to conduct a origin bypass attack. Origin bypass is a type of security vulnerability that occurs when an attacker is able to gain access to a web application or other system by bypassing the same-origin policy. The same-origin policy is a security measure that restricts access to web applications or systems based on their origin, which is typically determined by the domain, protocol, and port of the application or system.

Risk

The impact of origin bypass can be significant for both the organization hosting the web application or system and its users. For the organization, it can result in loss of user trust and damage to the organization's reputation. For the users, it can lead to financial loss or identity theft if their sensitive information is accessed and used by the attacker.

Recommendations

To mitigate the risk of origin bypass, it's important for organizations to implement security measures to enforce the same-origin policy and prevent unauthorized access to their web applications and systems. This may involve implementing authentication and access controls, as well as regularly reviewing and updating the security measures in place to ensure that they are effective. Additionally, regular security assessments and penetration testing can help to identify and address potential vulnerabilities in the system.

Affected URL & Parameter

Throughout the Application

CVSS Vector

4.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.95 Sensitive Path Disclosure		
CVSS Risk Rating	Medium	Status	Not Applicable

Observation Details

Upon testing the web application, it came to our notice that the web application had a sensitive path disclosure vulnerability, which is a type of security vulnerability that occurs when an application reveals the physical location of sensitive files or directories on the server. This can happen if the application is configured to return error messages that include the path of the sensitive files or directories, or if the application is misconfigured and allows direct access to these files or directories.

Risk

The impact of sensitive path disclosure can be significant for both the organization hosting the application and its users. For the organization, it can result in loss of user trust and damage to the organization's reputation. For the users, it can lead to financial loss or identity theft if their sensitive information is accessed and used by the attacker.

Recommendations

To mitigate the risk of sensitive path disclosure, it's important to:-

Properly configure their applications to avoid revealing the physical location of sensitive files or directories. This may involve implementing security measures such as authentication and access controls to prevent unauthorized access to these files or directories.

Regular security assessments and penetration testing can help to identify and address potential vulnerabilities in the application.

Affected URL & Parameter

Throughout the Application

CVSS Vector

5.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.96 Banner Grabbing		
CVSS Risk Rating	Low	Status	Open

Observation Details

Web server fingerprinting is a critical task for the penetration tester. Knowing the version and type of a running web server allows testers to determine known vulnerabilities and the appropriate exploits to use during testing.
There are several different vendors and versions of web servers on the market today. Knowing the type of web server that is being tested significantly helps in the testing process and can also change the course of the test. This information can be derived by sending the web server specific commands and analysing the output, as each version of web server software may respond differently to these commands. By knowing how each type of web server responds to specific commands and keeping this information in a web server fingerprint database, a penetration tester can send these commands to the web server, analyse the response, and compare it to the database of known signatures.

Risk

Banner grabbing does not involve the leakage of critical data but rather information that may aid the attacker during the exploitation phase of the attack. For example, if the target leaks the version of PHP running on the server and that version happens to be vulnerable to remote command/code execution (RCE) because it wasn't updated, attackers may exploit the known vulnerability and take full control of the web application.

Recommendations

The Following are the recommendations that should be implemented.
While exposed server information is not necessarily in itself a vulnerability, it is information that can assist attackers in exploiting other vulnerabilities that may exist. Exposed server information can also lead attackers to find version-specific server vulnerabilities that can be used to exploit unpatched servers. For this reason, it is recommended that some precautions be taken. These actions include:
• Obscuring web server information in headers, such as with Apache's mod headers module.
• Using a hardened reverse proxy server to create an additional layer of security between the web server and the Internet.
• Ensuring that web servers are kept up to date with the latest software and security patches.

Affected URL & Parameter

Throughout the Application

CVSS Vector

2.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

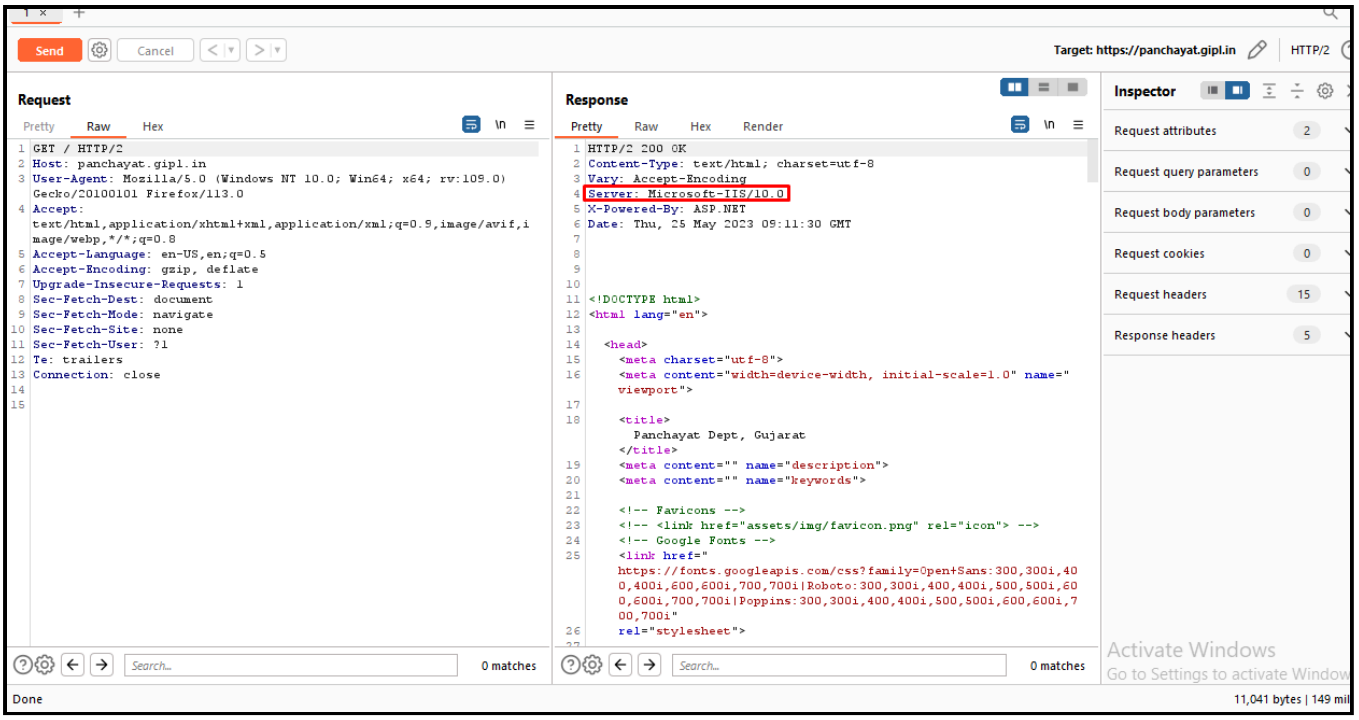


Figure 20 - As you can see that the application version is clearly visible.

Observation ID & Title	8.97 Lucky13		
CVSS Risk Rating	Low	Status	Open

Observation Details

The SSL LUCKY13 is a cryptographic timing attack that can be used against implementations of the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols using the Cipher Block Chaining (CBC) mode of operation. This can also be considered a type of man-in-the-middle attack. The TLS protocol, the successor of the Secure Sockets Layer (SSL) protocol, provides privacy, data integrity, and secure traffic between communicating networks or applications.
The vulnerability that makes the SSL LUCKY 13 possible affects the TLS 1.1 and 1.2 and DTLS 1.0 or 1.2 implementations. It also affects previous versions such as SSL 3.0 and TLS 1.0.

Risk

As a result of a successful attack, an attacker exploiting this vulnerability is able to read the plaintext of a TLS encrypted session. This can result in the loss of sensitive information.

Recommendations

The Following are the recommendations that should be implemented:
• Write the code in such a way that success and error cases consume same time so that an attacker cannot measure the time difference.
• Use AEAD Ciphers like AES-GCM, these ciphers calculate MAC while encrypting the message itself. Use Encrypt-then-MAC.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Figure 21 - As you can see that application is vulnerable to Lucky 13.

Observation ID & Title	8.98 No Session Timeout		
CVSS Risk Rating	Low	Status	Open

Observation Details

A session timeout vulnerability is a security flaw that arises when a web application fails to properly manage the session expiration of a user's authenticated session. This can result in a user's session remaining active even after they have left the site, potentially allowing an attacker to gain unauthorized access to sensitive information.

Risk

Unauthorized Access: If an attacker is able to hijack an active session, they can gain unauthorized access to sensitive information such as user credentials, personal data, and financial information.
Data Theft: A session timeout vulnerability can allow an attacker to steal data from a user's session, potentially leading to identity theft, financial fraud, and other forms of cybercrime.
Data Manipulation: An attacker may be able to manipulate data within a user's active session, potentially leading to the insertion or alteration of sensitive information.

Recommendations

To avoid session timeout vulnerabilities, web applications should implement appropriate session management
techniques such as:
• Setting a reasonable session timeout duration, based on the sensitivity of the data being accessed and the likelihood of the user remaining idle for an extended period of time.
• Implementing an automatic logout feature that terminates a user's session after a set period of inactivity.
• Ensuring that session IDs are securely generated and transmitted using secure channels.
• Using secure session storage mechanisms, such as HTTP-only cookies or server-side storage.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

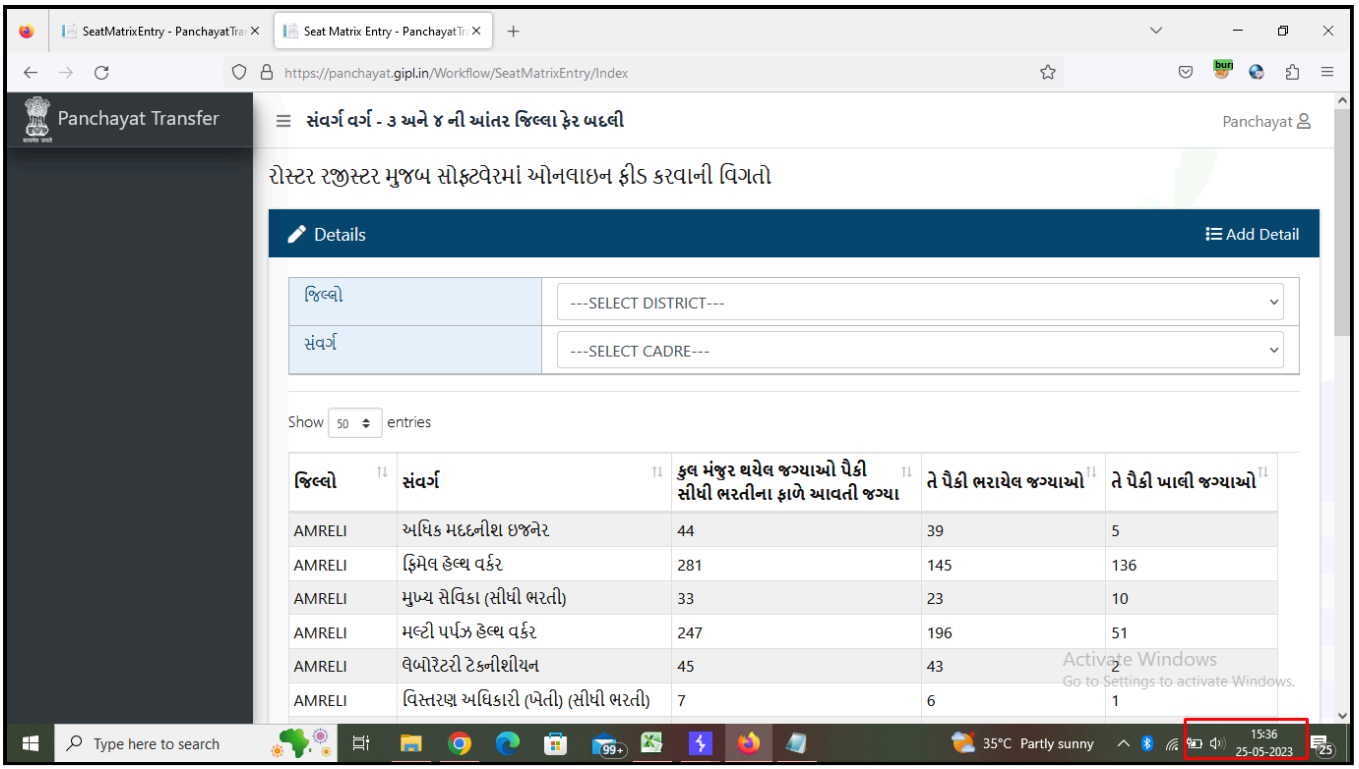


Figure 22 - As you can see that the application is not terminating the session even after 30 min of idle time.

Observation ID & Title	8.99 Fingerprint Web Application Framework		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

While conducting the testing of the application, the team came across an informational vulnerability. We were able to gather information about the web application frameworks being used. Fingerprinting web application frameworks is one the foremost tests that is conducted during the information gathering phase. By knowing the version and type of a running web server helps in determining known vulnerabilities. Mostly, web applications have known HTML headers, cookies, and directory structures that can be enumerated to identify the application. Most of the web frameworks have several markers in those locations which help an attacker to recognize them.

Risk

An attacker might be able to gain information about any outdated content management systems such as Wordpress, which the attacker might exploit in order to further gain access to sensitive information.

Recommendations

The Following are the recommendations that should be implemented.

Correct the information leakage from headers.

Disable all HTTP-headers that disclose information of the technologies used.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.100 Fingerprint Web Application		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Web server fingerprinting is one of the critical task for a penetration tester. By knowing the version and type of a running web server allows testers to determine known vulnerabilities and the appropriate exploits to use during testing.

Risk

An attacker can know the server or application services version and can use that version to exploit the whole web servers for the malacious use.

Recommendations

The Following are the recommendations that should be implemented.

While efforts can be made to use different cookie names (through changing configs), hiding or changing file/directory paths (through rewriting or source code changes), removing known headers, etc. such efforts boil down to “security through obscurity”. System owners/admins should recognize that those efforts only slow down the most basic of adversaries. The time/effort may be better used on stakeholder awareness and solution maintenance activities.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.101 File Extensions Handling for Sensitive Information		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

The way that a web application handles file extensions for sensitive information can pose a potential security vulnerability. This is because some file extensions are associated with specific types of files, and an attacker could potentially exploit this association to trick a user into revealing sensitive information.

Risk

If a web application allows users to upload files with the ".pdf" file extension, an attacker could potentially create a malicious file with the same file extension and trick a user into downloading and opening it. As the file has the ".pdf" file extension, the user might assume that it is a safe and legitimate PDF document, and not realize that it is actually a malicious file.

Recommendations

The Following are the recommendations that should be implemented.

To prevent this type of attack, it's recommended to carefully handle file extensions for sensitive information. This can include things like checking the content of the file to ensure that it matches the expected file type, and providing users with clear warnings when they download or open files with potentially dangerous file extensions.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.102 Dangerous Method Enabled		
CVSS Risk Rating	Low	Status	Open

Observation Details

While testing the application it came to our notice that the application is vulnerable is allowing dangerous methods like PUT, HEAD, DELETE these method allow an attacker to modify the files stored on the web server and, in some scenarios, steal the credentials of legitimate users

Risk

Web server accepting these methods may allow an attacker to gain full control over the application and its environment. The same methods can be also be used to cause Denial of Service (DoS) by destroying the application structure . An unauthenticated attacker can gain access to complete information held in databases that is not meant to be accessed by them. The attacker could also gain control over the entire database server which can leads to confidentially, integrity and availability.

Recommendations

The Following are the recommendations that should be implemented.

1) Always enable deny all option.
2) Configure your web and application server to disallow HEAD requests entirely

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.3 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

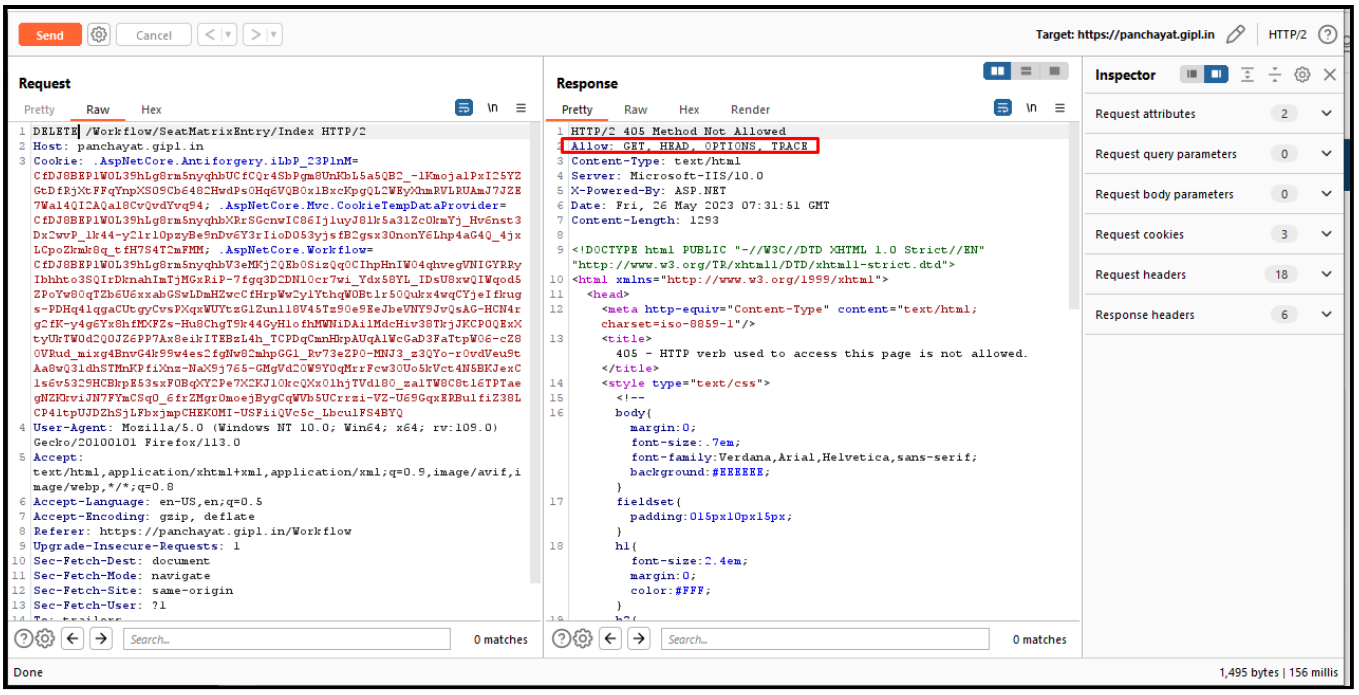


Figure 23 - As you can see that HEAD, TRACE method is allowed.

Observation ID & Title	8.103 HTTP Strict Transport Security		
CVSS Risk Rating	Low	Status	Open

Observation Details

During the testing phase, it came to our notice that the web application does not have HTTP Strict Transport Security mechanism implemented. The HTTP Strict Transport Security (HSTS) is a policy mechanism that is implemented through the use of a special response header. HSTS basically prevents any communications from being sent over HTTP and will only allow communications over HTTPS (encrypted channel). This mechanism will automatically redirect HTTP requests over to HTTPS for the target domain. It also prevents HTTPS click through prompts on browsers.

Risk

The attacker can conduct various attacks if the HSTS policy mechanism is not implemented. These attacks include:

Protocol Downgrade attacks: The Protocol downgrade attack is an attack through which, an attacker can downgrade the used protocol in the web application to communicate using any vulnerable protocol. Using the vulnerable protocol, the attacker can steal data sent between the server and the client.

Man-In-The-Middle attacks: The man-in-the-middle (MITM) attack is an attack through which an attacker can sniff data passed via the communication channel. The attacker can access sensitive information from login credentials to application's cookies.

Cookie-hijacking: The cookie hijacking attack is an attack through which, an attacker can hijack information present in the user's cookie from the communication channel. The cookie details include information about the user's session. The attacker can use this information to steal user's sessions.

Recommendations

The Following are the recommendations that should be implemented.

Configure the remote web server to communicate using HSTS.

If there is any preload directive in the application, it is recommended to switch back to HTTP. An attacker can send a preload directive from the application. These preload directives might have serious issues on the server. The preload directive can be used to prevent the users from accessing the web application along with any of its subdomains.

The web application must instruct the user's web browser to only access the application using HTTPS. To do this, the application must enable HTTP Strict Transport Security (HSTS). The HSTS can be enabled by adding the response header 'Strict-Transport-Security'. Set the value 'max-age=expireTime'. It is also recommended to add the 'includeSubDomains' flag.

Affected URL & Parameter

Throughout the Application
CVSS Vector
3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

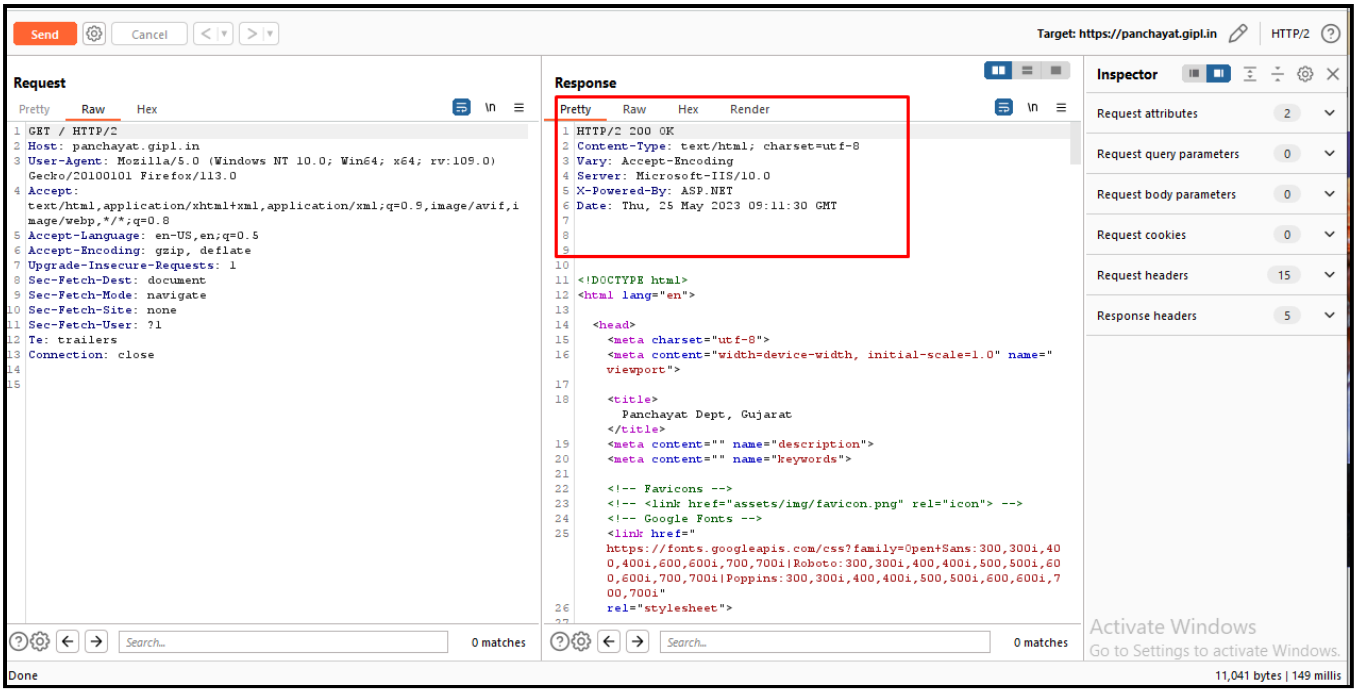


Figure 24 - As you can see that HTTP Strict Transport Policy is not enforced.

Observation ID & Title	8.104 RIA cross domain policy		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

RIA cross domain policy vulnerabilities refer to security issues that arise when a web application built using a rich internet application (RIA) framework, such as Adobe Flash or Silverlight, fails to properly implement cross-domain policies. These policies are designed to prevent one domain from being able to access the data of another domain, and are an important security measure to protect sensitive information from being accessed by unauthorized parties.

Risk

Using this vulnerability, an attacker can perform Cross-Site Request Forgery (CSRF) attacks or even read files that are protected using the cross-domain policy.

Recommendations

It is highly recommended to properly implement the cross-domain-policy.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.105 Account Enumeration and Guessable User Account		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

During our testing, i came to our notice that this application is vulnerable to Account Enumeration and Guessable user account vulnerability. It happen when web applications reveal when a username exists on system, either as a consequence of mis-configuration or as a design decision

Risk

Due to this vulnerability, it is possible for attackers to collect a set of valid usernames by interacting with the authentication mechanism of the application, or by using brute-force.

Recommendations

The Following are the recommendations that should be implemented.

Ensure the application returns consistent generic error messages in response to invalid account name, password or other user credentials entered during the log in process.

Ensure default system accounts and test accounts are deleted prior to releasing the system into production (or exposing it to an untrusted network).

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.106 Weak or unenforced username policy		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

During our testing, It came to our notice that this application is vulnerable to weak or unenforced username policy. A weak username policy vulnerability is a security vulnerability that occurs when an application or website allows users to choose weak or easily guessable usernames. This can make it easier for attackers to gain access to user accounts, either by guessing the username or by using a brute-force attack to try a large number of possible username combinations.

Risk

Due to this vulnerability, it make it easier for attackers to gain access to user accounts, either by guessing the username or by using a brute-force attack to try a large number of possible username combinations.

Recommendations

The Following are the recommendations that should be implemented.

Implement strong username policies that require users to choose unique and secure usernames that are not easily guessable.

Ensure the application returns consistent generic error messages in response to invalid account name, password or other user credentials entered during the log in process.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.107 Credentials Transported over an Encrypted Channel		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Transporting credentials over an encrypted channel can still be vulnerable to attack in certain situations. For example, if an attacker is able to compromise the encryption keys or the encryption algorithms being used, they could potentially intercept and read the login information as it is being transmitted. A lot of times, attackers can read intercept and credentials being sent over an encrypted channel. This means that the secure channel transmission has not been implemented properly.

Risk

If the user's device or the server is not properly authenticated/configured, an attacker could potentially impersonate the user or the server and trick the user into sending their login information to the attacker instead.

Recommendations

It is highly recommended to use:-

Use strong and up-to-date encryption algorithms and keys to encrypt the login information. This makes it much more difficult for attackers to intercept and read the login information.

Implement strong authentication measures for both the user's device and the server receiving the login information. This can include things like two-factor authentication, where the user is required to provide an additional form of authentication (such as a one-time code sent to their phone) in order to access their account.

Regularly monitor and test their systems for vulnerabilities.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.108 Weak lock out mechanism		
CVSS Risk Rating	Low	Status	Open

Observation Details

A lock-out mechanism is a security feature that is designed to prevent brute-force attacks by temporarily locking an account after a certain number of failed login attempts. However, if the lock-out mechanism is weak, it may not be effective at preventing brute-force attacks, allowing an attacker to continue trying to guess a user's password until they are successful.

Risk

The impact of this vulnerability can be significant, as it can allow an attacker to gain unauthorized access to a user's account. This can be used to steal sensitive information, perform actions on behalf of the user, and potentially compromise the security of the entire system.

Recommendations

It is highly recommended to:-

Implement a strong lock-out mechanism that is effective at preventing brute-force attacks.

Use longer lock-out periods, requiring additional authentication factors.

Implement CAPTCHA challenges to verify that the user is human.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

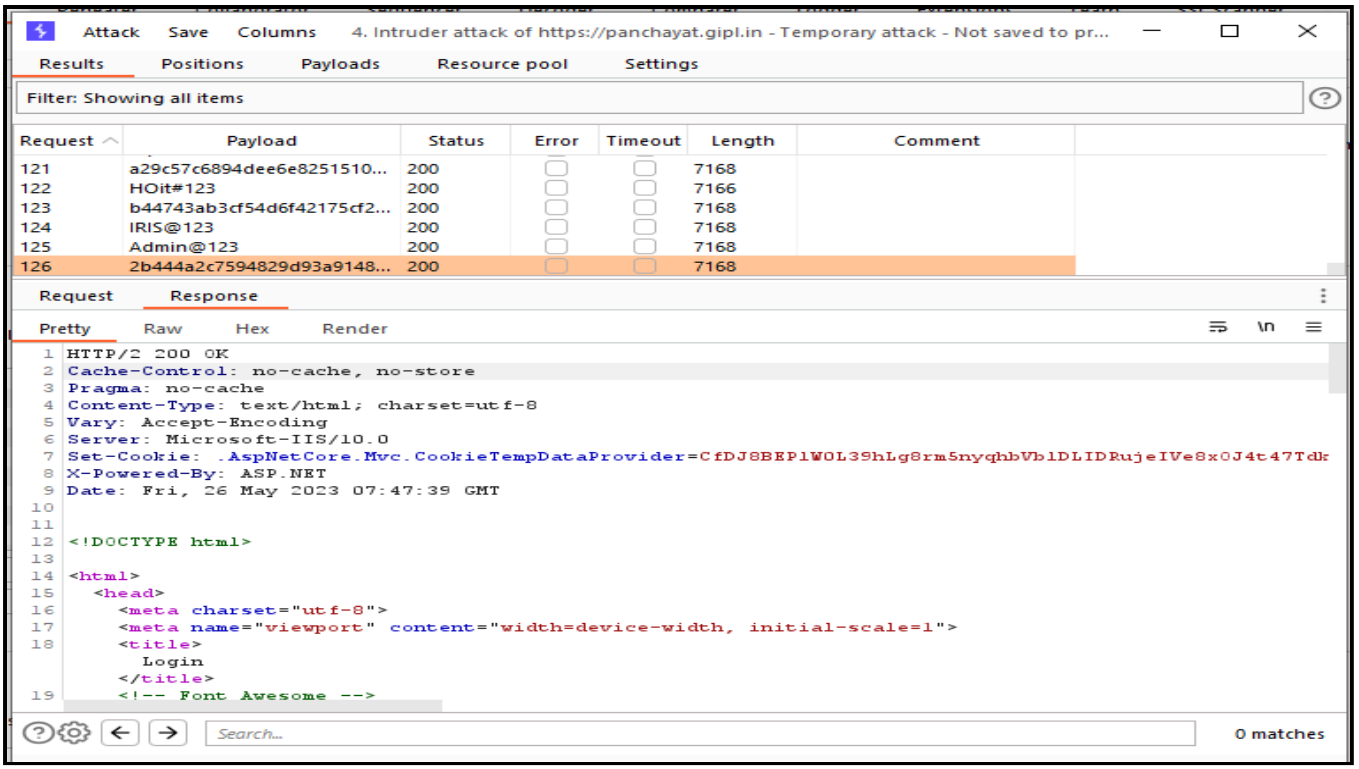


Figure 25 - As you can see that after attempting so many passwords, account is not getting lockout.

Observation ID & Title	8.109 AutoComplete Enabled		
CVSS Risk Rating	Low	Status	Open

Observation Details

AutoComplete is basically a feature which is provided by many web browsers / web applications. This eases the tiring process of typing the credentials again and again. Although seen as a feature, having AutoComplete enabled in a web browser can pose a potential security vulnerability. This is because AutoComplete automatically fills in login information (such as a username and password) on websites that the user has previously visited. This can make it easier for attackers to gain access to a user's account if they have physical access to the user's device.

Risk

If an attacker is able to gain access to the user's device, they could potentially open the web browser and navigate to a website where the user has previously logged in. If AutoComplete is enabled, the login information will be automatically filled in, and the attacker could potentially use this information to gain access to the user's account.

Recommendations

To mitigate the potential security vulnerability, it is advised to:-

Disable AutoComplete in the web browser.

Use a password manager.

Be careful about where you leave your device.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

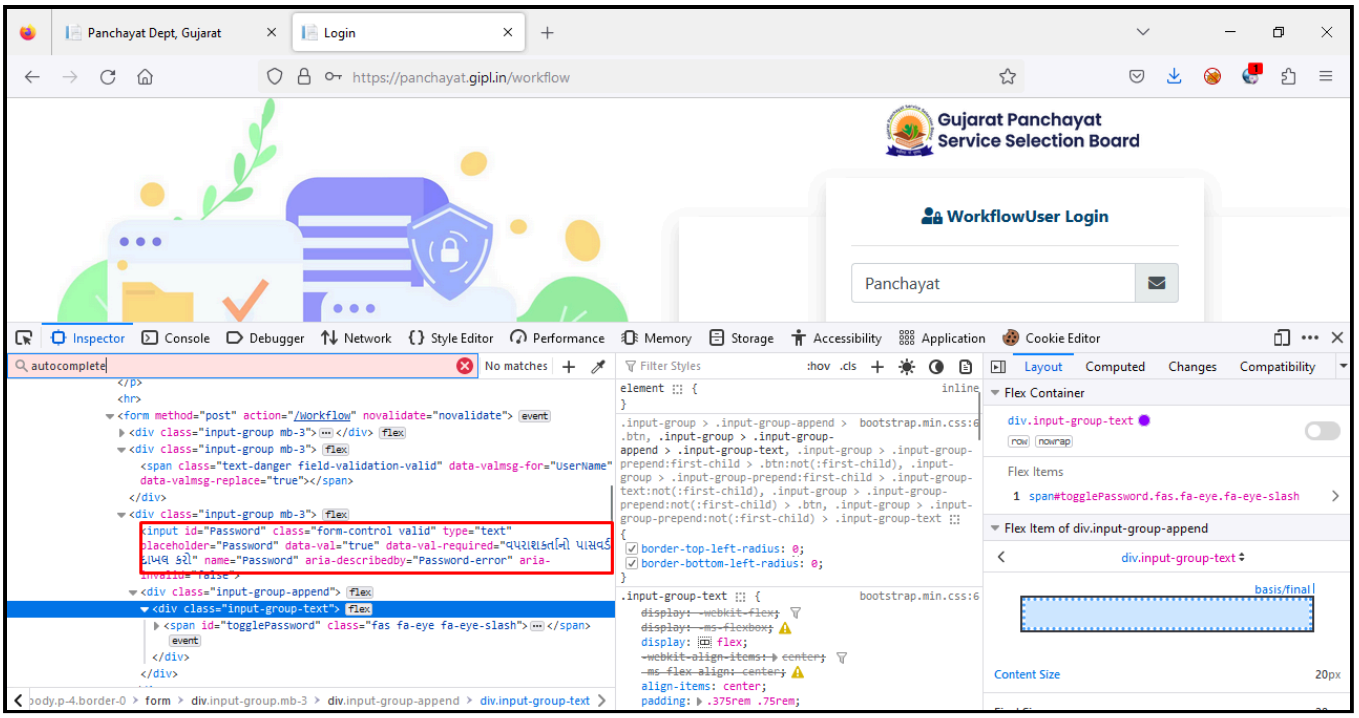


Figure 26 - As you can see that the autocomplete is not set to OFF.

Observation ID & Title	8.110 Weak password policy		
CVSS Risk Rating	Low	Status	Open

Observation Details

While testing the web application, it came to our notice that the application has weak password policies. Weak password policies vulnerability comes under the category of Broken authentication, and this happens due to insufficient mention of password hardening rules in the policy. Password hardening rules such as keeping the minimum length of password of 8 characters, using alphanumeric characters along with special characters etc. This ensures that passwords cannot be easily guessed or brute forced.

Risk

Attackers have to gain access to only a few accounts or just one admin account to compromise the whole system. Depending on the domain of the application this may allow social security fraud, or identity theft and disclose legally protected highly sensitive information.

Recommendations

The Following are the recommendations that should be implemented.

Tight password policy, not allowing weak or well-known passwords and not the usage of default admin credentials.

Ensure passwords have a minimum length of 8, consist alphanumeric, special characters and both lowercase and uppercase characters.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.6 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

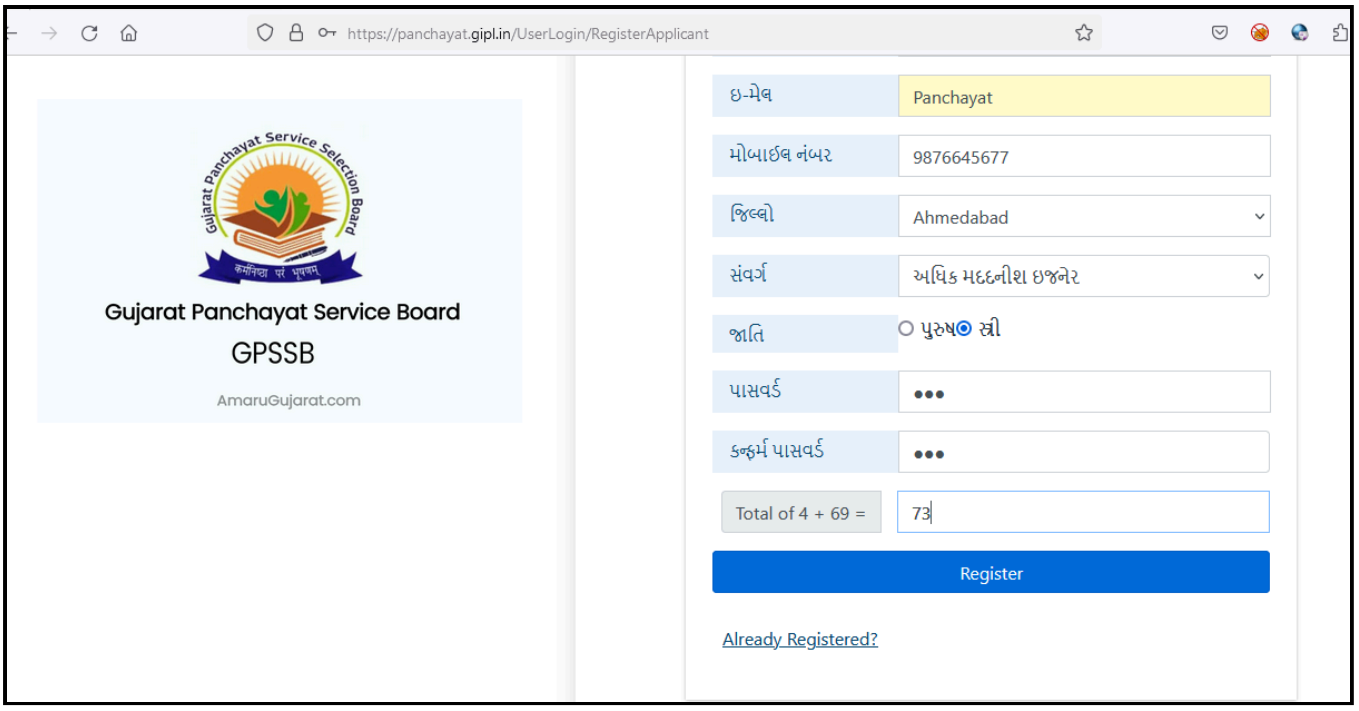


Figure 27 - As you can see that the application is accepting only 3 char of password.

Observation ID & Title	8.111 Weak security question/answer		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

A weak security question/answer is a question and answer pair that is easily guessed or obtained by someone trying to gain unauthorized access to an account or system. For example, a common security question is "What is your mother's maiden name?" An easy-to-guess answer for this question might be the name of the person's hometown or the name of a well-known person.

Risk

A weak security question/answer can have serious consequences, such as unauthorized access to an account system, identity theft or fraud, such as stealing your personal information, financial information, or using the account to send spam or malicious messages.

Recommendations

To make your security questions and answers more secure, it's important to choose questions and answers that are unique and not easily guessable by others. Additionally, it's a good idea to avoid using answers that can be found on social media or other public information sources.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.112 Weaker authentication in alternative channel		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

While testing the application we found that the application si vulnerable to Weaker authentication in alternative channel vulnerability. Weaker authentication in alternative channel vulnerability occurs when an attacker is able to bypass the normal authentication process by using an alternative channel or method that has weaker security measures in place.

Risk

Weaker authentication in alternative channel vulnerability allows attackers to gain unauthorized access to sensitive information or systems. This can lead to data breaches, financial loss, and reputational damage for the affected organization

Recommendations

The Following are the recommendations that should be implemented.

Please ensure that all authentication methods, including alternative channels, use strong and secure authentication criteria.

Ensure a consistent authentication policy is applied across all channels so that they are equally secure.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.113 Missing secure Flag Not Set		
CVSS Risk Rating	Low	Status	Open

Observation Details

During the of the web application, it was observed that there was no secure flag set in the web application. When an HTTP protocol is used for communication between client and server, the data traffic is sent in plaintext. An HHTP allows the attacker to see/modify the traffic using a Man-In-The-Middle attack (MITM). HTTPS is a secure version of HTTP. This protocol uses SSL/TLS to protect the data in the application layer. HTTPS is used for better authentication and data integrity. A secure flag is set by the application server while sending a new cookie to the user using an HTTP Response. The secure flag is used to prevent cookies from being observed and manipulated by an unauthorized party or parties. This is because the cookie is sent as a normal text. A browser will not send a cookie with the secure flag that is sent over an unencrypted HTTP request. That is, by setting the secure flag the browser will prevent/stop the transmission of a cookie over an unencrypted channel.

Risk

Using this vulnerability, an attacker can redirect the user to a malicious site to steal information/data or even show user false data which will, in turn, affect the credibility of the website.

Recommendations

It is highly recommended to set the HTTP Secure Flag to prevent cookies from being manipulated.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

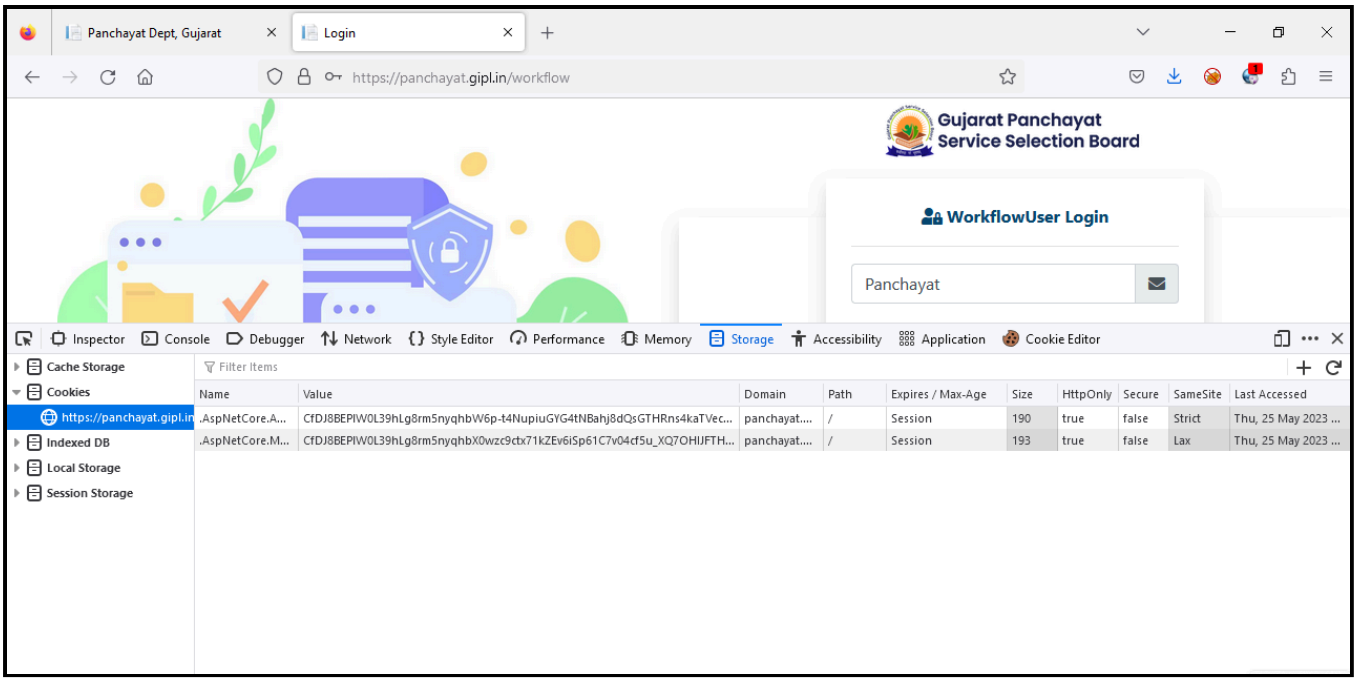


Figure 28 - As you can see that the secure flag is set to false.

Observation ID & Title	8.114 Missing HTTP only Flag Set		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

An HTTP cookie is a small piece of information that a server sends to the user's web browser. The Cookie header stores the HTTP cookies previously sent by the web server with the Set-Cookie header. The session cookies are deleted when the browser shuts down and if the cookies are permanent, they will expire at the time defined by Expires or Max-Age. The risk of client-side scripts accessing the protected cookie can be mitigated by including an additional "HttpOnly" flag in the Set-Cookie HTTP response header. As a result, the browser will not reveal the cookie to a third party even if a cross-site scripting (XSS) flaw exists in the web application. The given web application was found to be not having HTTP Only Flag set. The missing flag should immediately be set.

Risk

During a cross-site scripting attack, an attacker might easily access cookies and using these he may hijack the victim's session. An attacker can grab the sensitive information contained in the cookie.

Recommendations

It is highly recommended that:-

Set HTTPOnly on the cookie. This helps mitigate a large part of XSS attacks attempting to capture the cookies and possibly leaking sensitive information or allowing the attacker to impersonate the user.

The HTTP TRACE method combined with XSS can read the authentication cookie, even if the HttpOnly flag is used. So make sure that the HTTP TRACE method is disabled.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.115 Simultaneous Login Possible		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

While testing the application it came to our notice that the application is allowing login from two different browser or devices at the same time, and both the browser have the same account logged in. Enabled simultaneous login is basically which allows users to log in at the same time using the same user/password. Essentially, this is how many concurrent user sessions the server will allow for the web application when using the same user account. Although this is considered as a security vulnerability.

Risk

Due to this vulnerability an attacker can hijack one of the valid session and perform malicious activity.

Recommendations

It is recommended to make sure that the application does not allow the same user ID to be logged in from two different locations at the same time. The application should warn the user that another valid session with the same user ID is being used. This can be achieved by storing some identifying information like IP address or a unique GUID in every user's session state.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.116 Session puzzling		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Session Puzzles are vulnerabilities at the application level that can be exploited by overriding session attributes, also referred to as 'Session Variable Overloading.' We observe this vulnerability occurs when the same session variable is used for more than one purpose, which makes it possible to grant access to pages in an order that is unanticipated by developers, whereby a session variable set in one context may be used in another. In this case, the given web application was also vulnerable to this attack, and we were able to successfully bypass authentication.

Risk

Session Puzzling is an application level vulnerability which can enable an attacker to perform a variety of malicious actions such as:-

Bypass efficient authentication enforcement mechanisms, and impersonate legitimate users.

Elevate the privileges of a malicious user account, in an environment that would otherwise be considered foolproof.

Skip over qualifying phases in multi-phase processes, even if the process includes all the commonly recommended code level restrictions.

Manipulate server-side values in indirect methods that cannot be predicted or detected.

Execute traditional attacks in locations that were previously unreachable, or even considered secure.

Recommendations

In order to mitigate this vulnerability, it highly recommended to:-

Avoid storing unnecessary values in the session.

Avoid using the same session variable with identical names in different modules and entry points, mainly public and private entry points.

Avoid storing objects in the session instead of variables.

Avoid using a session as a temporary container of values.

Perform validations on the session originating values before using them in the application code.

Affected URL & Parameter

Throughout the Application

CVSS Vector
3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.117 Improper Session Expiry		
CVSS Risk Rating	Low	Status	Open

Observation Details

Improper session expiry refers to the failure of a website or web application to properly manage the expiration of user sessions. This can occur if the website or application does not properly set the expiration time for a user's session, or if it fails to invalidate the user's session when it expires. The given web application was found to have improper session expiry.

Risk

If an organization is successfully targeted by an attack that exploits improper session expiry, the impact can be significant. The attacker may be able to gain unauthorized access to user accounts, steal sensitive information, or perform unauthorized actions on the user's behalf. This can result in loss of confidentiality, integrity, and availability of the organization's data, as well as damage to its reputation and financial losses.

Recommendations

To mitigate the risks associated with improper session expiry, appropriate security controls should be implemented, such as using strong encryption for data transmission and authentication mechanisms for verifying the identity of users. They should also ensure that their website or web application properly manages the expiration of user sessions, including setting appropriate expiration times and invalidating sessions when they expire.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

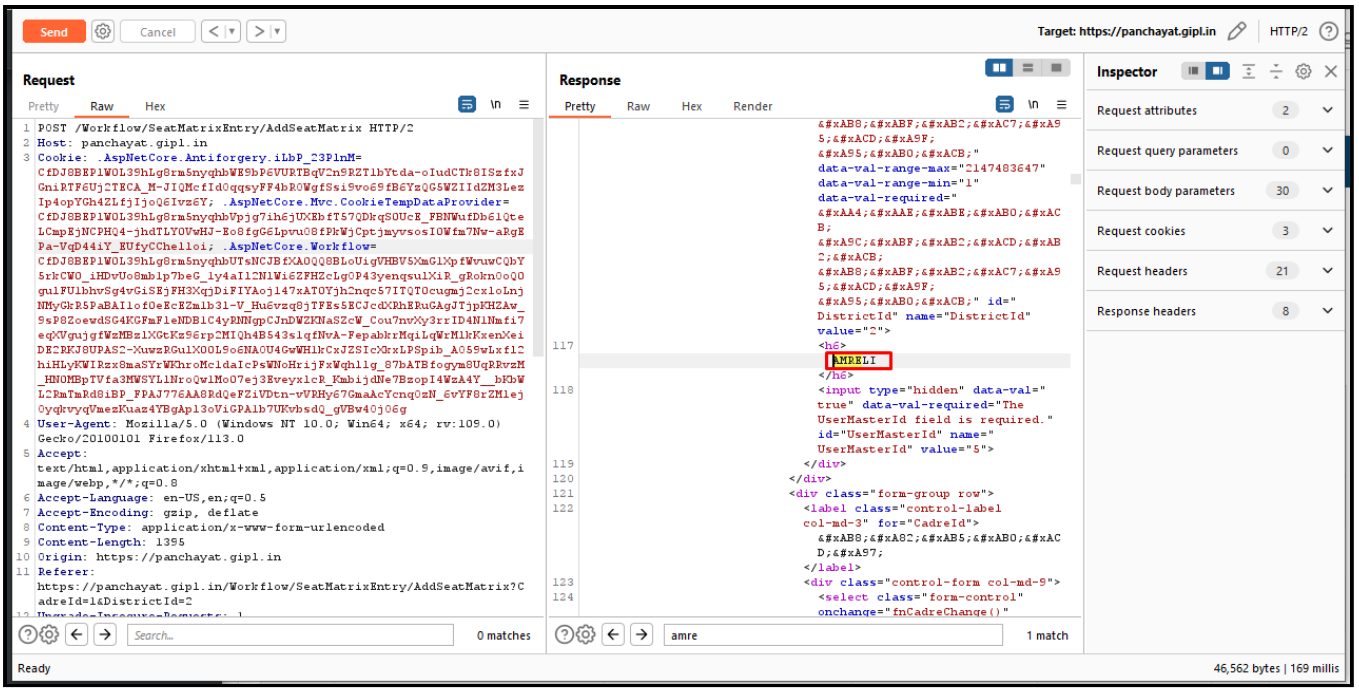


Figure 29 - Intercept the request of the page containing the information > Send the request to repeater

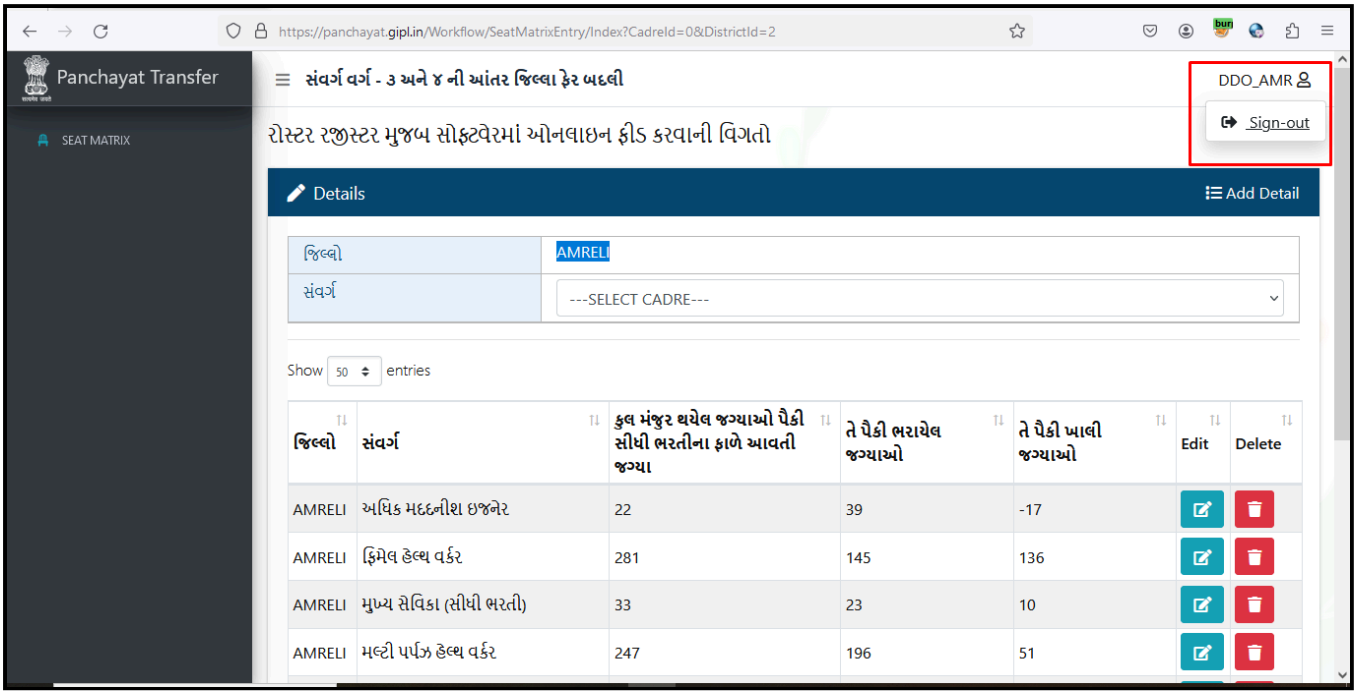


Figure 30 - Signout the account.

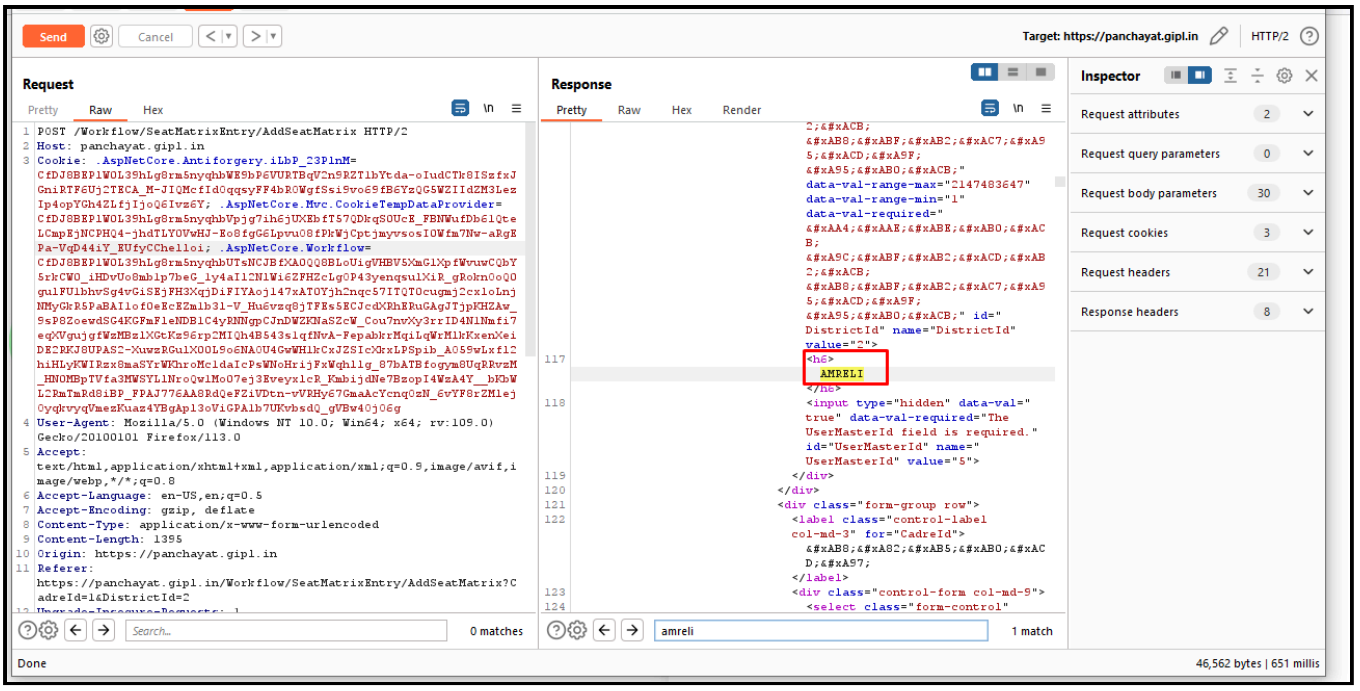


Figure 31 - Now again go to Repeater and send the request > As you can see that the application is showing the data.

Observation ID & Title	8.118 Cross-Site Scripting		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Cross-Site Scripting (XSS) is a type of security vulnerability that allows an attacker to inject malicious code into a website. An IE-only XSS attack is a specific type of XSS attack that only affects users who are using the Internet Explorer web browser.

In an IE-only XSS attack, the attacker creates a malicious website or web page that is designed to exploit a weakness in the Internet Explorer web browser. When a user visits the malicious website or web page using Internet Explorer, the malicious code is executed in the user's browser, allowing the attacker to steal sensitive information, such as login credentials or other sensitive data.

Risk

This type of attack can be particularly dangerous because it only affects users who are using the Internet Explorer web browser, which is commonly used by many people. These are specifically designed to exploit vulnerabilities in the Internet Explorer web browser.

Recommendations

To protect against IE-only XSS attacks, it is recommended to sanitize and validate all data that is sent to the website, including data from users who are using the Internet Explorer web browser. This can help prevent attackers from injecting malicious code into the website. Encrypt and secure cookies to protect against XSS attacks and other types of security vulnerabilities. Users can also protect themselves against IE-only XSS attacks by using a different web browser or by keeping their web browser up-to-date with the latest security patches.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.119 IMAP/SMTP Injection		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

IMAP and SMTP are protocols used for accessing and managing email messages on a server. An injection vulnerability in an IMAP or SMTP server could potentially allow an attacker to inject malicious commands into the server, potentially allowing them to access or modify email messages or gain access to sensitive information.

Risk

The impact of an IMAP/SMTP injection vulnerability can be significant, depending on the nature of the injected commands and the data that is accessed or modified. In the worst case, it could allow an attacker to gain access to sensitive information, such as user passwords or financial data, or to execute arbitrary code on the server.

Recommendations

To mitigate the risks associated with IMAP/SMTP injection vulnerabilities, it is highly recommended to:-

Properly validate and sanitize any user-provided input that is used in IMAP or SMTP commands. This can help prevent attackers from being able to inject malicious commands into the server.

Implement appropriate authentication and authorization controls.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.120 Insufficient Transport Layer Protection		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Insufficient transport layer protection is a type of vulnerability that occurs when a web application does not properly secure the data that is transmitted between the client and the server. This can allow attackers to intercept and view sensitive information, such as passwords or financial data, that is transmitted over the network.

Transport layer protection typically involves using secure communication

Risk

If a web application does not properly implement transport layer protection, it may be vulnerable to attacks that can expose sensitive information and compromise the security of the application.

Some potential impacts of inadequate transport layer protection include:

Unauthorized access to sensitive information: If an attacker is able to intercept and view the data that is transmitted between the client and the server, they may be able to gain access to sensitive information, such as passwords or financial data.

Reputation damage: If an attacker is able to successfully exploit a lack of transport layer protection, it could damage the reputation of the organization that operates the web application, potentially leading to loss of trust from customers and other stakeholders.

Legal and regulatory consequences: Depending on the nature and severity of the attack, the organization may be subject to penalties or fines from regulatory authorities for failing to properly implement transport layer protection.

Recommendations

The Following are the recommendations that should be implemented.

To protect against attacks and ensure the security and integrity of a web application, it is important to properly implement transport layer protection. Some recommended steps for implementing effective transport layer protection include:

Use secure communication protocols: When transmitting data between the client and the server, it is important to use secure communication protocols such as HTTPS (Hypertext Transfer Protocol Secure) to encrypt the data. This can help prevent attackers from being able to view the data as it is transmitted over the network.

Use strong encryption algorithms: When encrypting data, it is important to use strong algorithms that are resistant to attack. This includes algorithms such as AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman).

Properly validate certificates: When using secure communication protocols, it is important to properly validate the certificates that are used to establish secure connections. This can help prevent attackers from being able to intercept and view encrypted data.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.121 Ability to Forge Requests		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

The ability to forge requests refers to the ability of an attacker to create and send malicious requests to a web application. This can include forging the source of the request, the contents of the request, or the intended destination of the request.

Risk

Forged requests can be used to perform a variety of malicious actions, such as:

Gaining unauthorized access to sensitive information: An attacker may be able to use forged requests to gain access to sensitive information, such as financial records or personal information, that is stored in the application's database.

Modifying or deleting data: An attacker may be able to use forged requests to modify or delete data in the application's database, which could cause serious damage to the integrity of the application.

Denial of service: An attacker may be able to use forged requests to cause the application to crash or become unresponsive, resulting in a denial of service for legitimate users.

Reputation damage: If an attacker is able to successfully forge requests, it could damage the reputation of the organization that operates the application, potentially leading to loss of trust from customers and other stakeholders.

Recommendations

it is recommended that it is important to implement proper security measures, such as implementing authentication and authorization controls and regularly reviewing and testing the application for vulnerabilities.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.122 HTML Injection		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

During the testing of web application, it came to notice that this application is vulnerable to HTML Injection. Html injection allows attacker to inject certain HTML tags. When an application does not properly handle user supplied data, an attacker can supply valid HTML code, typically via a parameter value, and inject their own content into the page.

Risk

This vulnerability allow the attacker to manipulate a trustful but vulnerable website against HTML Injection. They can modify or create a fake webpage by using stored HTML Injection or they achieve XSS. After achieving XSS, they can steal cookies, hijack accounts, steal credentials and other sensitive information.

Recommendations

The Following are the recommendations that should be implemented.

Please use a proper input validation mechanism, check user inputs. Filter special tags.

Please encode user input and then process it for further action.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.123 Client Side URL Redirect		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

A client-side URL redirect is a technique in which a web page is automatically redirected to a different URL by the client (typically the user's web browser) rather than by the server. This can be useful in some situations, but it can also be a security vulnerability if not implemented properly.

Risk

The impact of a client-side URL redirect vulnerability can be significant, depending on how it is exploited. For example, an attacker could potentially redirect a user to a malicious website that is designed to steal sensitive information, such as user passwords or financial data. In the worst case, this could lead to data breaches, loss of sensitive information, or other security incidents.

Recommendations

To mitigate the risks associated with client-side URL redirects, it is recommended to:-

Properly validate and sanitize any user-provided input that is used to specify the redirect URL. This can help prevent attackers from being able to manipulate the redirect to redirect users to a malicious website.

Implement appropriate authentication and authorization controls, and regularly scanning and testing the website for vulnerabilities, can help identify and address potential vulnerabilities in a timely manner.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.6 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.124 CSS Injection		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

A CSS Injection vulnerability involves the ability to inject arbitrary CSS code in the context of a trusted web site which is rendered inside a victim's browser. The impact of this type of vulnerability varies based on the supplied CSS payload. It may lead to cross site scripting or data exfiltration.

This vulnerability occurs when the application allows user-supplied CSS to interfere with the application's legitimate stylesheets. Injecting code in the CSS context may provide an attacker with the ability to execute JavaScript in certain conditions, or to extract sensitive values using CSS selectors and functions able to generate HTTP requests. Generally, allowing users the ability to customize pages by supplying custom CSS files is a considerable risk.

Risk

The impact of this type of vulnerability varies based on the supplied CSS payload. The attacker might be able to conduct attacks such as cross site scripting or even be able to exfiltrate critical data.

Recommendations

It is highly recommended to:-

Always filter user input from malicious injections.

Use CSP Header (Content-Security-Policy) which allows you to prevent browsers from executing malicious code on your website.

Ensure that user input is adequately escaped before embedding it in CSS blocks, and consider using a whitelist to prevent loading of arbitrary style sheets.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.125 IFRAME Injection		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

During our testing, we notice that the web application is vulnerable to IFRAME Injection. An IFrame injection is a common attack that combines malicious JavaScript with an iframe that loads a legitimate page in an effort to steal data from an unsuspecting user.

Risk

Due to CORS misconfiguration the application is 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 the website's authenticated users such as retrieving user setting information or saved payment card data and many more.

Recommendations

The Following are the recommendations that should be implemented.

If a web resource contains sensitive information, the allowed origin(s) should be specified in full in the Access-Control-Allow-Origin header.

Please avoid using wildcards in internal networks.

Origins specified in the Access-Control-Allow-Origin header should only be sites that are trusted.

Avoid using wildcards in internal networks.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.7 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.126 Cross Origin Resource Sharing		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

While testing the web application, it came to our notice that this web application is vulnerable to Cross Origin Resource Sharing vulnerability. Cross-origin resource sharing (CORS) is a browser mechanism which enables controlled access to resources located outside of a given domain. It extends and adds flexibility to the same-origin policy (SOP). However, it also provides potential for cross-domain attacks, if a website's CORS policy is poorly configured and implemented.

Risk

Due to CORS misconfiguration the application is 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 the website's authenticated users such as retrieving user setting information or saved payment card data and many more.

Recommendations

The Following are the recommendations that should be implemented.

If a web resource contains sensitive information, the allowed origin(s) should be specified in full in the Access-Control-Allow-Origin header.

Please avoid using wildcards in internal networks.

Origins specified in the Access-Control-Allow-Origin header should only be sites that are trusted.

Avoid using wildcards in internal networks.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.127 Cross Site Flashing		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

The Cross Site Flashing is similar to a Reflected Cross-site Scripting (XSS) vulnerability. An attacker could execute arbitrary Flash code within the WordPress security sandbox. This means that an attacker can craft code to send a request to any URL within the domain WordPress is hosted on, using the victim's cookies. Cross Site Flashing vulnerability was found in the given web application.

Risk

An attacker is able to trick the victim into executing a Flash document that passes commands or calls to a Flash player browser plugin, allowing the attacker to exploit native Flash functionality in the client browser. An attacker can cause arbitrary content to be referenced and possibly executed by the targeted Flash application.

While the attacker may not manipulate the DOM, if the victim has Adobe Flash enabled, the attacker can also send requests and read responses to obtain WordPress CSRF tokens and perform actions on behalf of the user. If that user happens to be an administrator, this includes the possibility of installing malicious plugins which the attacker may then escalate to a much more grievous attack.

Recommendations

It is recommended to:-

Use individual random tokens (e.g. CSRF tokens) in the handshake request and validate them against the server.

Check the Origin header of the WebSocket handshake request on the server.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.128 Clickjacking		
CVSS Risk Rating	Low	Status	Open

Observation Details

While testing the application it came to our notice that the application is vulnerable to clickjacking that is UI redress attack. In this attack click is hijacked that means any one can put multiple transparent or opaque layers to trick user into clicking on a button or link of another page when they were intending to click on the top level of page.

Risk

By exploiting this vulnerability, an attacker can trick the user to click on their desired page most likely owned by another application. Thus an attacker can steal sensitive information such as user credentials, credit card details etc.

Recommendations

The Following are the recommendations that should be implemented.

Implementing security policy such as Content Security Policy (CSP), or implementing the X-Frame option headers that will instruct the browser to not allow framing from other domains is recommended.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

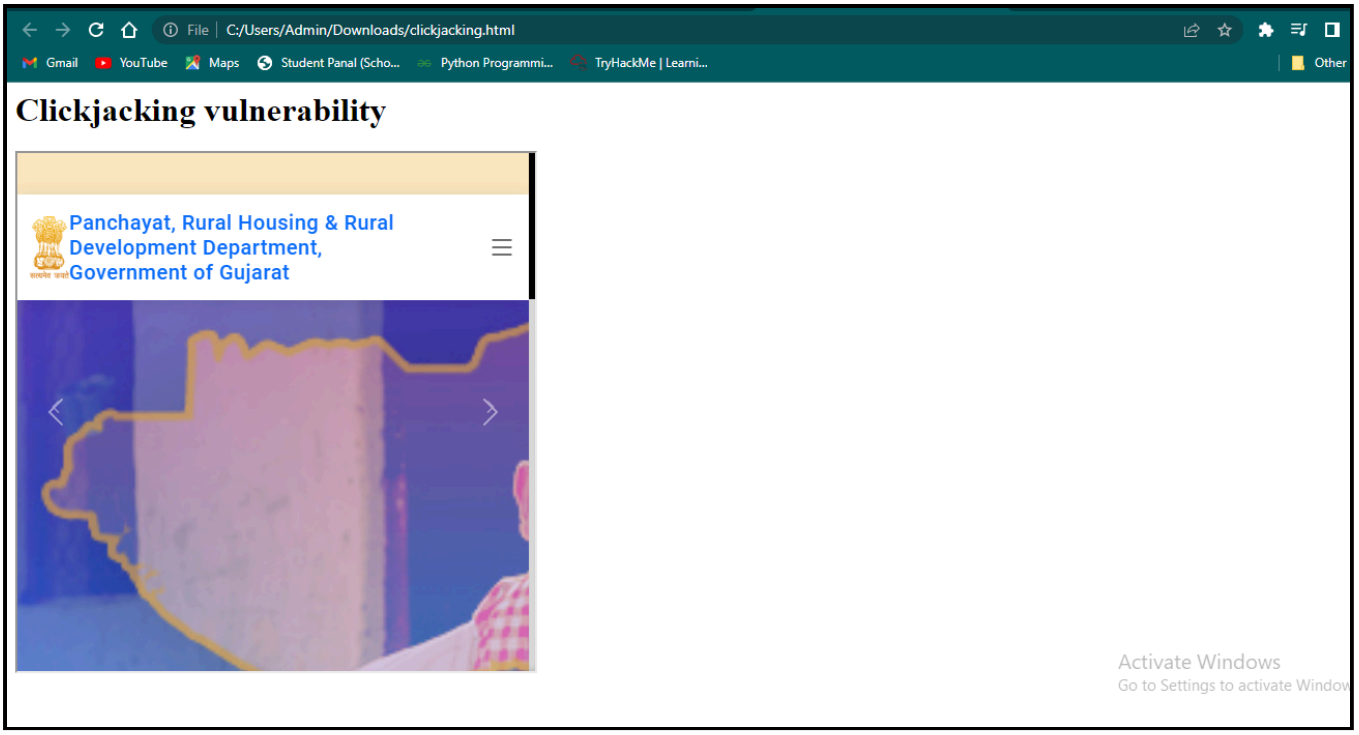


Figure 32 - As you can see that the application is vulnerable to clickjacking attack.

Observation ID & Title	8.129 Cache Poisoning		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Web cache poisoning involves sending a specially crafted request that generates a malicious response that is saved in the web cache and sent to other legitimate users. If a response is cached in a shared web cache, such as in proxy servers, then all users who share that cache will receive the malicious response on each request afterward until the cache entry is removed, otherwise, only the user of the local browser instance will get affected.

Risk

The impact of the manipulated response could be different according to the usage of the web cache for the request; whether it is cached either by a web-cache used by multiple users or a single user. The poisoned response will only be served to users who visit the affected page while the cache is poisoned. As a result, the impact can range from non-existent to massive depending on whether the page is popular or not. If an attacker managed to poison a cached response on the home page of a major website, the attack could affect thousands of users without any subsequent interaction from the attacker.

Recommendations

The Following are the recommendations that should be implemented.

Disabling caching entirely would not be a workable solution. However, it would be better to opt for certain preventive measures, such as:

Ensure only cache static resources, such as *.js, *.css, *.png files that are always identical.

Make sure you are not vulnerable to XSS as it won't affect the client browser even if the vulnerability exists.

Understand and restrict the caching. Handle caching at a singular point (Cloudflare) framework that implements their own caching.

Avoid using user inputs as the cache key.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.130 TLS 1.0 And TLS 1.1 Enabled		
CVSS Risk Rating	Low	Status	Open

Observation Details

Transport Layer Security (TLS) are cryptographic protocols that provide communications security over a computer network. Several versions of the protocols find widespread use in applications such as web browsing, email, instant messaging, and voice over IP (VoIP). Websites can use TLS to secure all communications between their servers and web browsers. Since applications can communicate either with or without TLS (or SSL), it is necessary for the client to indicate to the server the setup of a TLS connection. During the testing of the web application, it was found that TLS version 1.0 / 1.1 was being used for communication in the web application. These versions, as discussed above are outdated.

Risk

TLS 1.0 version is vulnerable to many implementations and it fails to shield against attacks such as BEAST and POODLE. This version of TLS can be easily breached by the attackers.

TLS 1.1 is an outdated version. The pseudo random function in TLS is based on a combination on a MD5 and SHA-1.The attacker can easily break these function and in return can cause severe damage to the server. TLS 1.1 was defined in RFC 4346 in April 2006 It is an update from TLS version 1.0. Using even any one of the above mentioned versions of TLS, the attacker could access sensitive information, with attacks such MiTM, etc.

Recommendations

The Following are the recommendations that should be implemented.

As the web application is using an outdated version of TLS, it is highly recommended to update TLS to the latest version, TLS 1.3.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

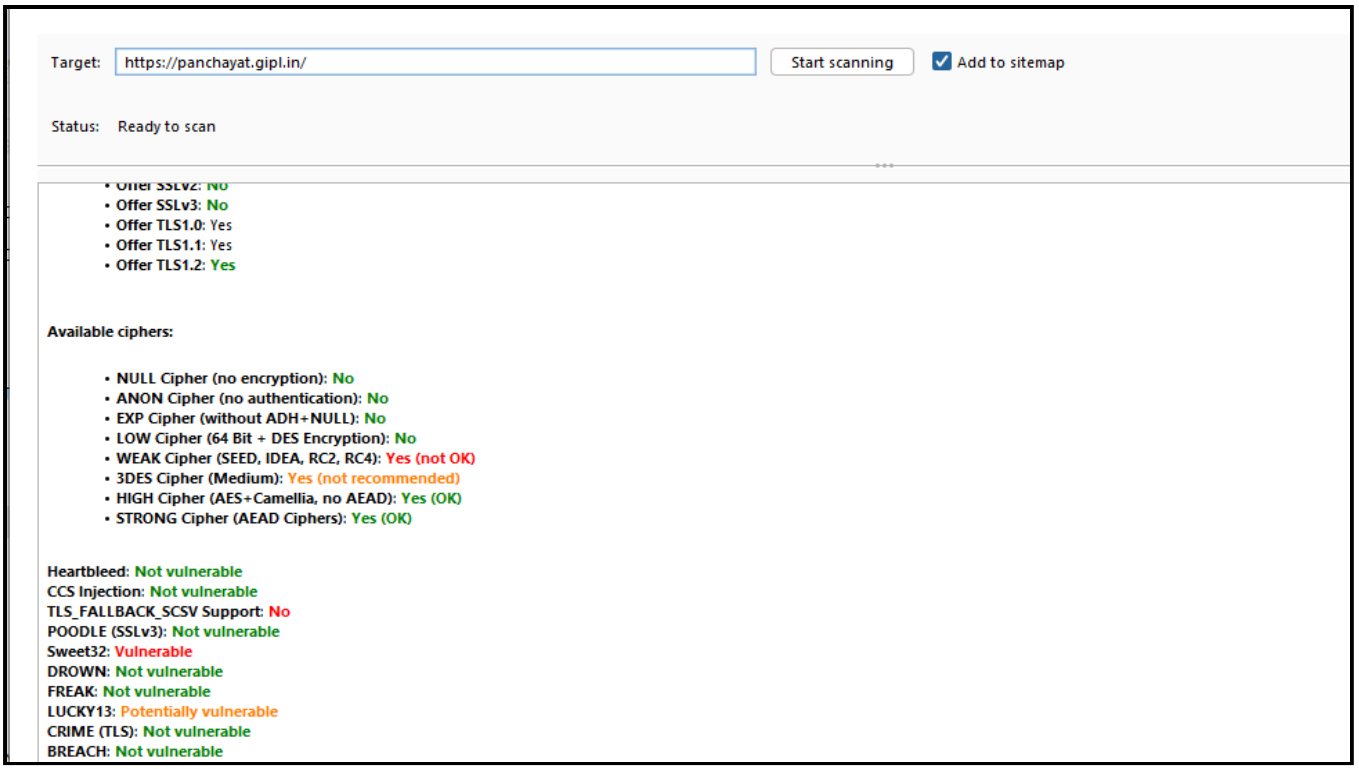


Figure 33 - As you can see TLS 1.0 & 1.1 is enabled.

Observation ID & Title	8.131 Sweet32 Attack		
CVSS Risk Rating	Low	Status	Open

Observation Details

The Sweet32 is an attack first found by researchers at the French National Research Institute for Computer Science (INRIA). The attack targets the design flaws in some ciphers that are used in TLS, SSH, IPsec, and OpenVPN. The Sweet32 attack allows an attacker to recover small portions of plaintext. It is encrypted with 64-bit block ciphers (such as Triple-DES and Blowfish), under certain circumstances. The SWEET32 attack can be used to exploit the communication that uses a DES/3DES based cipher suite. The SWEET32 attack affects the commonly used algorithm like AES (Advanced Encryption Standard), Triple-DES (Data Encryption Standard) and Blowfish for encrypting communication for TLS, SSH, IPsec and OpenVPN protocol. The given web application was found to be having this vulnerability.

Risk

An attacker can exploit this vulnerability to conduct many attacks. A man-in-the-middle attacker could use this flaw to recover some plaintext data. The attacker can steal large amounts of encrypted traffic between TLS/SSL server and client.

Recommendations

It is highly recommended to :-

Use OpenSSL security update RHSA-2016:1940.

Try to avoid the usage of legacy 64-bit block ciphers.

Use 128-bit ciphers for encryption in servers and VPNs.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

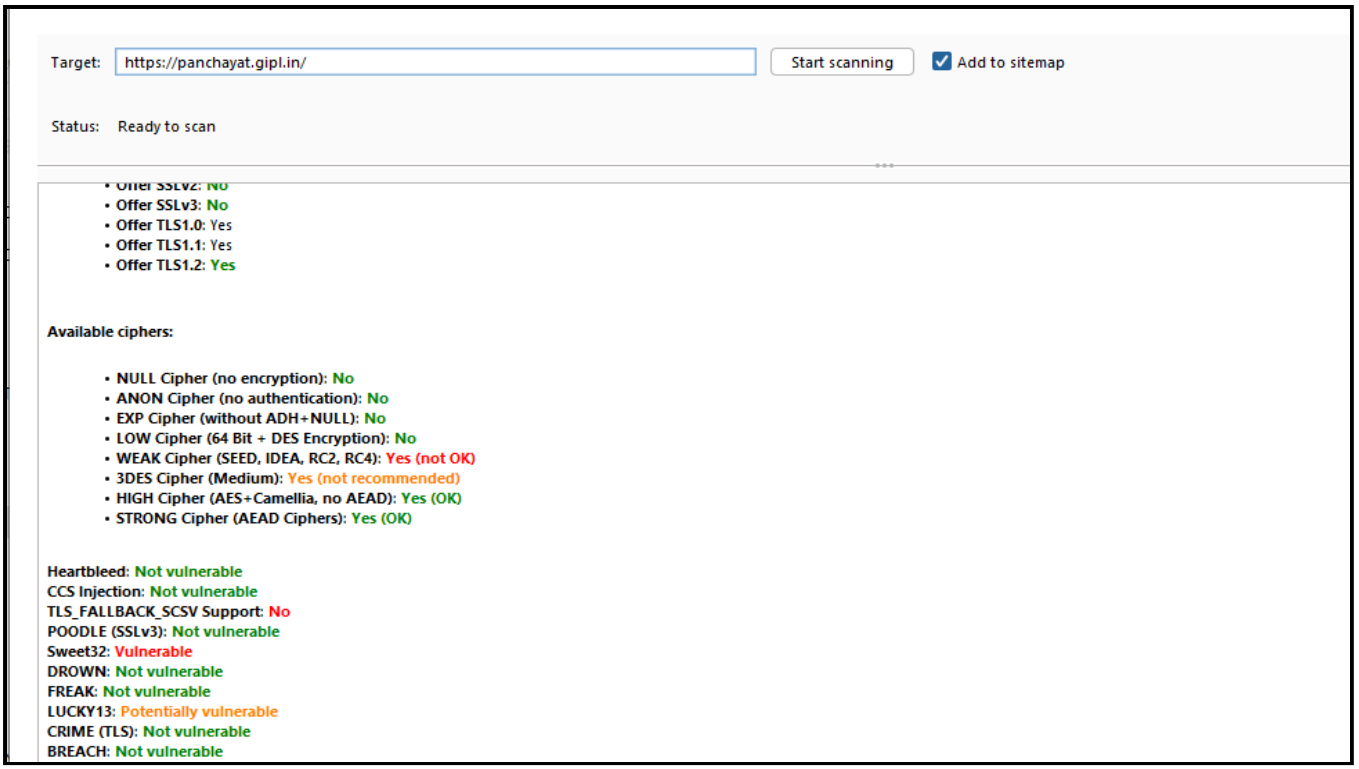


Figure 34 - As you can see that the application is vulnerable to TLS Sweet 32.

Observation ID & Title	8.132 BREACH Attack		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

The BREACH attack is a security vulnerability that affects web applications that use the HTTP compression algorithm. This attack allows an attacker to obtain sensitive information, such as security tokens and user passwords, by exploiting the way the HTTP compression algorithm works.

Risk

The BREACH attack steals information about how data is encrypted from HTTPS-enabled Web applications by essentially combining two existing types of attacks: using cross-site request forgery (CSRF) to change data in transport, and injecting data into the HTTPS headers using a man-in-the-middle attack.

Recommendations

It is highly recommended to:-

Disable HTTP compression.

Separate secrets from user input.

Randomize secrets per request.

Mask secrets (effectively randomizing by XORing with a random secret per request).

Protect vulnerable pages with CSRF.

Hide length by adding random number of bytes to the responses).

Rate-limit the requests.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.133 Old Cipher Supported		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Old ciphers are encryption algorithms that are no longer considered secure and should not be used for protecting sensitive information. If an application or system still supports old ciphers, it is vulnerable to attacks that can break the encryption and allow attackers to access the protected information.

Risk

The impact of this vulnerability can be significant, as it can expose sensitive information such as user passwords, financial data, and other confidential information to unauthorized access.

Recommendations

To mitigate this vulnerability, it is important to regularly review and update the supported ciphers in your application or system to ensure that only secure and up-to-date algorithms are used for encryption. This may involve disabling support for old ciphers, and enabling support for newer, more secure ciphers.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.134 Improper Input Validation		
CVSS Risk Rating	Low	Status	Open

Observation Details

While testing the web application we found out that this application is vulnerable to Improper Input Validation vulnerability. Improper Input Validation can allow an attacker to supply malicious user input that is then executed by the vulnerable web application. Improper input validation can be used to bypass security mechanisms, such as authentication and authorization controls. It can also be used to inject malicious code into the web application, which can be executed by the server or client.

Risk

Improper Input Validation allows an attackers to craft the input in a form that is not expected by the rest of the application. This will lead to parts of the system receiving unintended input, which may result in altered control flow, arbitrary control of a resource, or arbitrary code execution, leakage of sensitive information, injections attacks that split database.

Recommendations

The Following are the recommendations that should be implemented.

Sanitize data enter by user before processing.

They can be removed by constrain input, reject known bad input, validate data for type, length and range

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.6 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

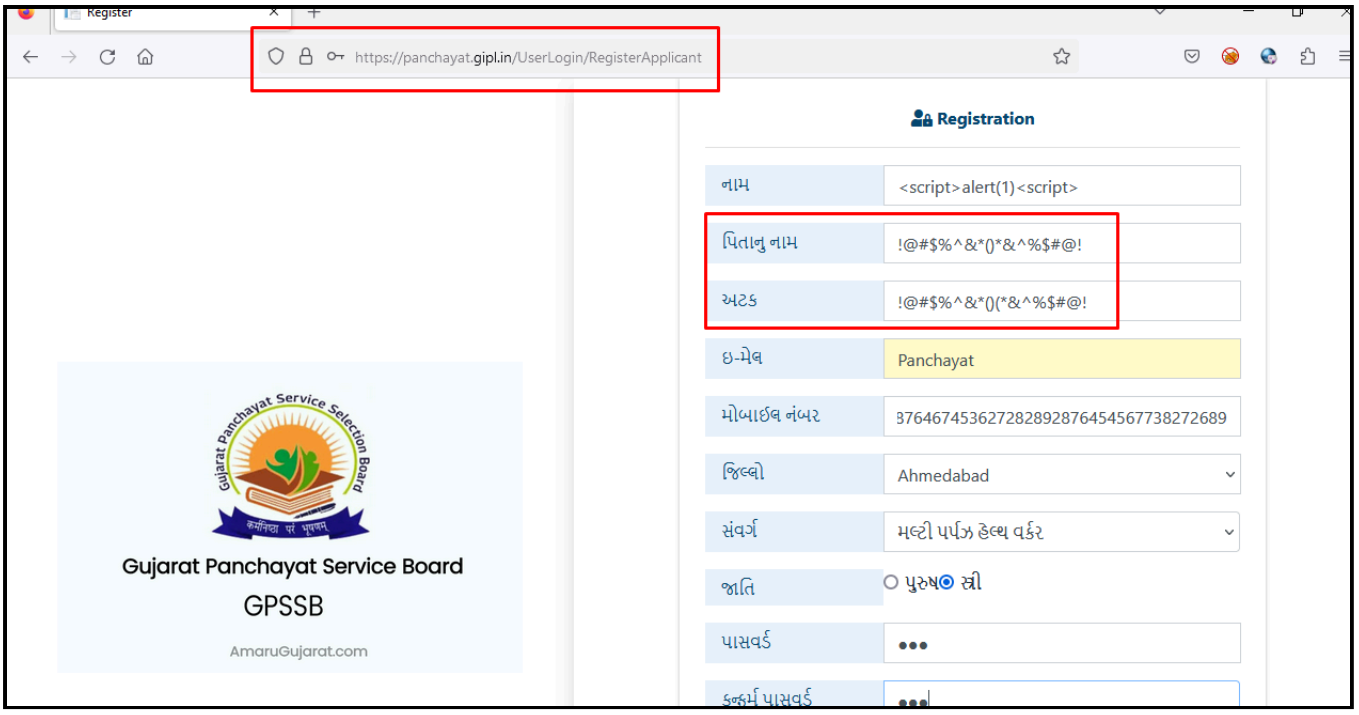


Figure 35 - As you can see that the application is accepting the special characters and those which has to be blacklisted.

Observation ID & Title	8.135 Directory Listing Enabled		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

While testing the web application we found that this web application is vulnerable to Directory Listing vulnerability. Directory listing is a web server function that displays the directory contents when there is no index file in a specific website directory. It is dangerous to leave this function turned on for the web server because it leads to information disclosure, expose sensitive file in the web browser.

Risk

Directory Listing vulnerability allows an attacker can see the files located in the directory and could potentially access files which disclose sensitive information, internal directory structure of a web application .

Recommendations

The Following are the recommendations that should be implemented.

Configure your web server to prevent directory listings for all paths beneath the web root

Place into each directory a default file (such as index.htm) that the web server will display instead of returning a directory listing.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.136 Captcha Bypass		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

As we were conducting the testing of the web application, we were able to bypass the CAPTCHA that was prompted upon the website. Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) is a type of challenge-response test that is used to distinguish humans from automated programs i.e., bots. Many websites use CAPTCHAs on their login pages to protect against automated attacks such as user identifier enumeration, brute-force password guessing, credential stuffing and password spraying attacks. Typically, CAPTCHAs are image based that require users to recognize a distorted sequence of alpha-numeric letters or solve a simple mathematical puzzle.

Risk

If this vulnerability is found in an application, an attacker can create a bot to bypass the captcha and automate the tasks to sent unlimited requests to a multiple URLs or lists with random/fake users, emails, IP address etc. for spamming or evil purposes (collect data, analyze traffic behaviors, etc.)

Recommendations

The Following are the recommendations that should be implemented.

IP Blacklisting.

Response Time Monitoring.

Switching between CAPTCHAs.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.137 No Token/Cookie Validation		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Failing to validate tokens or cookies in a web application can allow an attacker to gain unauthorized access to a user's account or perform actions on behalf of the user. This is because tokens and cookies are often used to identify and authenticate users, and if they are not properly validated, an attacker can potentially forge or manipulate them to gain access to the application.

Risk

The impact of this vulnerability can be significant, as it can allow an attacker to access sensitive information, perform actions on behalf of the user, and potentially compromise the security of the entire system.

Recommendations

It is highly recommended to :-

Use secure and unique tokens.

Properly encrypt and sign cookies.

Regularly rotate and invalidate tokens and cookies.

Implement additional security controls.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.8 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.138 Sensitive Token in URL		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

During our testing, we have noticed that the application reveal sensitive token in URL. In this a kind of vulnerability sensitive information is embedded in various locations in URL which can be seen in the user's browser, the web servers, and the end points.

Risk

If a sensitive token is placed in the url, it increases the chances of getting exploited by attackers as it can be easily accessible by him. Attackers can use this token to personate as a victim and perform an action, Loss of Credentials may take place.

Recommendations

The Following are the recommendations that should be implemented.

Sensitive information should be transmitted in different way such as HTTP cookies.

information should be transmitted by post method in from of hidden filed

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.8 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.139 Token Leakage via Referer		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Token leakage via the Referer header is a type of vulnerability that occurs when an authentication token is included in the Referer header of an HTTP request. This can potentially allow an attacker to gain unauthorized access to a user's account or perform actions on behalf of the user.

Risk

The impact of this vulnerability can be significant, as it can allow an attacker to access sensitive information, perform actions on behalf of the user, and potentially compromise the security of the entire system.

Recommendations

It is highly recommended to:-

Remove the token from the Referer header.

Properly encrypt and sign tokens.

Regularly rotate and invalidate tokens.

Implement additional security controls.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.6 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.140 Content Spoofing		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

While testing web application it came to our notice that the web application is vulnerable to Content Spoofing vulnerability. Content spoofing occurs when an attacker alters the content of a website or application in a way that is intended to deceive or trick the user. This can be done by modifying the HTML or other markup language used to display the content, or by injecting malicious code into the page. It can be used to impersonate legitimate websites or applications, to steal sensitive information, or to spread malware

Risk

Due to Content Spoofing vulnerability attacker can spoofed website or application to trick users, which lead to loss of sensitive information such as password or financial information.

Recommendations

The Following are the recommendations that should be implemented.

Using proper synchronization techniques, such as locks or semaphores, to ensure that only one process or thread can access a shared resource at a time.

Regularly monitoring and testing your system to detect and address potential race condition vulnerabilities.

Implementing security measures, such as access control lists or authentication protocols, to prevent unauthorized access to shared resources

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.141 Exposed Admin Portal		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

CAPTCHA is a type of challenge-response test used by many web applications to ensure that the response is not generated by a computer. It is observed that that the captcha is not implemented on login page.

Risk

Using this vulnerability, an attacker can gain complete access to the application, perform manipulation of data, leak sensitive information, read, update and delete sensitive data/tables from the database etc.

Recommendations

The Following are the recommendations that should be implemented.

Install the updated patches.

Use a web application firewall.

Try to password protect admin directory.

Use strong passwords.

Try to implement 2-step verification.

Limit access to few IP addresses.

Disable login hints.

Create a custom login and registration page.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.142 No Captcha Implementation		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

CAPTCHA is a type of challenge-response test used by many web applications to ensure that the response is not generated by a computer. It is observed that that the captcha is not implemented on login page.

Risk

It seems to be a medium impact, but consider a situation when user forget to logout from his account or someone get access to his phone and delete the account. This situation is more severe than account takeover as there is no way to get account again. All the save information and data including previous record, card information etc will be deleted.

Recommendations

It is highly recommended to implement the security while performing suck task such as deleting account, Manage 2FA and etc. Proper authentication is requested for the user while doing such task.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.143 Lack of Password Confirmation		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

During testing the website, it comes in notice that the application is not asking for any kind of authentication while performing any sensitive task like deleting the account. This vulnerability is due to the fact that if you want to delete the account in the system, the system must ask you to re-confirm the password, otherwise, the account will not be deleted. Regarding the following operations, the system must receive a password confirmation from the user:

1. Change Email Address
2. Change Password
3. Delete Account
4. Manage 2FA

Risk

It seems to be a medium impact, but consider a situation when user forget to logout from his account or someone get access to his phone and delete the account.
This situation is more severe than account takeover as there is no way to get account again. All the save information and data including previous record, card information etc will be deleted.

Recommendations

It is highly recommended to implement the security while performing suck task such as deleting account, Manage 2FA and etc. Proper authentication is requested for the user while doing such task.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.144 Same-Site Scripting		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Cross-site scripting (XSS) is a type of security vulnerability that occurs when an attacker injects malicious code into a web application. This code is executed when a user visits the web application, allowing the attacker to perform various actions, such as stealing sensitive information or taking control of the user's account. Same-site scripting is a variant of XSS that occurs when the attacker injects code into a web application that they control.

Risk

The impact of a same-site scripting attack can vary depending on the nature of the code that is injected, but it can potentially allow an attacker to steal sensitive information, such as user passwords or financial data. It can also allow an attacker to perform actions on behalf of the user, such as making purchases or transferring funds.

Recommendations

To mitigate the risk of a same-site scripting attack, it is recommended to:-

Implement appropriate security measures. This can include implementing input validation, which ensures that all user-supplied data is properly sanitized before it is processed by the web application.

Use web application firewalls and other security measures to block or detect attempts to inject malicious code into the application.

Regularly update and patch the web application to fix any known vulnerabilities that could be exploited by an attacker.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.145 No Secure Integrity Check - Executable Download		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

An attacker can execute malicious code by compromising the host server, performing DNS spoofing, or modifying the code in transit.

Risk

An attacker can execute malicious code by compromising the host server, performing DNS spoofing, or modifying the code in transit.

Recommendations

The Following are the recommendations that should be implemented.

Encrypt the code with a reliable encryption scheme before transmitting.

This will only be a partial solution, since it will not detect DNS spoofing and it will not prevent your code from being modified on the hosting site.

If you are providing the code that is to be downloaded, such as for automatic updates of your software, then use cryptographic signatures for your code and modify your download clients to verify the signatures.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.146 Server-Side Request Forgery (SSRF)		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Server-Side Request Forgery (SSRF) is a type of security vulnerability that allows an attacker to send malicious requests to a server from a vulnerable web application. In an external SSRF attack, the attacker targets a server that is external to the vulnerable web application, such as a server on a different network or domain.

In an external SSRF attack, the attacker exploits a vulnerability in the web application to send a malicious request to an external server. This can be done through a variety of methods, such as manipulating the parameters of a URL or crafting a maliciously formatted request. When the external server receives the request, it processes the request as if it were legitimate, allowing the attacker to steal sensitive information or perform other malicious actions.

Risk

External SSRF attacks can be particularly dangerous because they can be used to target servers on different networks or domains, allowing the attacker to gain access to sensitive information that may not be accessible from the vulnerable web application. They often involve manipulating the parameters of a request in a way that is not easily detectable.

Recommendations

It is advised to:-

Use a whitelist of approved domains and protocols through which remote resources can be acquired by the web server.

User input should always be sanitised or validated.

One must verify that the server response received is as planned to avoid response data leakage to an attacker. The raw response body of the request sent by the server should not be delivered to the client under any circumstances.

If only HTTP or HTTPS are used by your application to make requests, allow only these URL schemas. If URL schemas like file:///, dict://, ftp:// and gopher:// are disabled, the attacker won't be able to use the web application to make dangerous requests using these URL schemas.

Services like Memcached, Redis, Elasticsearch, and MongoDB do not need authentication by default. Server Side Request Forgery vulnerabilities may be used by an attacker to access any of these services without any authentication. Therefore it is best to allow authentication wherever possible, even for services on the local network to ensure security for the web application.

Affected URL & Parameter

Throughout the Application

CVSS Vector
3.6 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.147 Weak Password Reset Implementation		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

A weak password reset implementation is a security vulnerability that occurs when a web application does not properly validate the user's identity before resetting their password. This can allow an attacker to reset the password of a user's account, potentially allowing them to gain access to the account and perform unauthorized actions.

Risk

The impact of a weak password reset implementation can be severe, as it can allow an attacker to gain access to a user's account and potentially steal sensitive information or perform unauthorized actions. This can lead to lost revenue and damage to the organization's reputation.

Recommendations

To mitigate the risk of a weak password reset implementation, it is recommended to:-

Implement appropriate security measures. This can include implementing strong password policies, which require users to choose complex and unique passwords for their accounts.

Properly validate the user's identity before resetting their password, using methods such as two-factor authentication or security questions.

Implement password expiration policies, which require users to change their password on a regular basis.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.148 Weak Registration Implementation		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

A failure to invalidate session after a password change is a security vulnerability that occurs when a user changes their password, but the system does not properly log them out of all active sessions. This can potentially allow attackers to continue using the old password to access the user's account, even after the password has been changed.

Risk

If a session is not properly invalidated after a user changes their password, it can potentially allow an attacker to gain unauthorized access to the user's account. This is because the old password may still be valid, and an attacker can potentially use it to gain access to the application.

The impact of this vulnerability can be significant, as it can allow an attacker to access sensitive information, perform actions on behalf of the user, and potentially compromise the security of the entire system.

Recommendations

The Following are the recommendations that should be implemented.

Implementing strong, secure registration processes that properly validate user input and require the use of strong passwords.

Implementing security measures, such as access control lists and authentication protocols, to prevent unauthorized access to sensitive information and resources.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.149 Misconfigured DNS - Zone Transfer		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Upon conducting testing on the web application we found that the application was vulnerable to DNS zone transfer, also known as DNS query type AXFR, is a process by which a DNS server passes a copy of part of its database to another DNS server. The portion of the database that is replicated is known as a zone. A zone transfer uses the Transmission Control Protocol (TCP) and takes the form of a client-server transaction. The client requesting a zone transfer may be a slave server or secondary server, requesting data from a master server or a primary server.

Risk

DNS zone transfer offers no authentication. So, any client or someone posing as a client can ask a DNS server for a copy of the entire zone. This means that unless some kind of protection is introduced, anyone is capable of getting a list of all hosts for the particular domain, which gives them a lot of potential attack vectors.

Recommendations

The Following are the recommendations that should be implemented.

Only allow a zone transfer from trusted IPs.

Use Transaction Signatures (TSIG) for zone transfers.

Affected URL & Parameter

N/A

CVSS Vector

3.2 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.150 Temp Mail Allowed		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

While conducting the penetration testing of the web application, we found out the the application allow temporary mail .It will give the attacker a temporary access for the registration or uploading the email without having the work mail . This will used for the malicious activity.

Risk

This vulnerability allow an attacker to create unlimited account or updating the email for the malicious activity by creating unlimited accounts.

Recommendations

The Following are the recommendations that should be implemented.

Disable the temporary email and application should not accept the temp mail and only accepts the original mails work mails or company mails.

Affected URL & Parameter

N/A

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.151 LOGJAM		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

After concluding the testing of the application, one of our findings were the Logjam vulnerability. Logjam is a security vulnerability against a Diffie–Hellman key exchange. Key sizes range from 512 to 1024 bits. This vulnerability allows an attacker to downgrade vulnerable TLS connections using Man-In-The-Middle (MITM) attack. This also allows the attacker to read and modify any data passed over the connection. This vulnerability exists in servers which supports DHE_EXPORT ciphers, which can be easily attacked.

Risk

Logjam vulnerability allows an attacker to weaken the encryption complexity, consequently decrypting data easily without the user's knowledge.

Recommendations

The Following are the recommendations that should be implemented.

Disable support for export cipher suites and use a 2048-bit Diffie-Hellman group.

Disable EXPORT cipher suits in the web server configuration.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.152 POODLE Attack		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Upon testing the web application, we were able to conduct a POODLE attack due to the padding CBC vulnerability existing in SSL version 3.0. POODLE (Padding Oracle on Downgraded Legacy Encryption) is an attack that occurs when an attacker exploits the significant weakness in the SSL protocol of version 3 (SSLv3).

Risk

If this vulnerability exists in a system, the attacker might be able to eavesdrop on an encrypted communication. Confidential data that is transmitted on the network can be stolen by the attacker. The attacker can then use this data for impersonating as a user or admin to access the database content. Sensitive information critical to an organization might also be accessed by the attacker due to to this vulnerability.

Recommendations

It is recommended to:

Configure the server to support only TLS 1.2 and above.

Disable all SSLv2 and SSLv3.

Enable TLS_FALLBACK_SCSV. This protocol extension guarantees that during a negotiation, the protocol never falls back to earlier protocol version, hence preventing a downgrade attack.

Affected URL & Parameter

N/A

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.153 Source Code Disclosure		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

During the testing phase, it came to our notice that the source code of the web application was viewable. Through the source code, a lot of information was being disclosed. Source code intended to be kept server-side can sometimes end up being disclosed to users. Such code may contain sensitive information such as database passwords and secret keys, which may help malicious users formulate attacks against the application.

Risk

An attacker would be able to view the source code of the web application and try to understand the working of the application The attacker might also try to find any vulnerabilities in the source code and exploit it.

Recommendations

The Following are the recommendations that should be implemented.

Disable all the HTTP-headers that disclose information.

Review the source code for any sensitive information.

Remove any unnecessary comments.

Remove all the META and generator tags.

Protect the application's presentation layer using a strong and hardened reverse proxy.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.154 HTTPOXY Attack		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Upon testing of the web application, it came to our notice that the web application was vulnerable to HTTPOXY attack. HTTPOXY is a set of vulnerabilities that affect application code running in CGI, or CGI-like environments. It comes down to a simple namespace conflict, RFC 3875 (CGI) puts the HTTP Proxy header from a request into the environment variables as HTTP_PROXY
HTTP_PROXY is a popular environment variable used to configure an outgoing proxy. This leads to a remotely exploitable vulnerability.

Risk

If this vulnerability is found in any web application, attackers can potentially redirect internal requests generated by the application to a server of their choosing, and thus capture any secret data contained in the requests, leading to sensitive data leakage.

Recommendations

It is recommended to immediately block Proxy request headers as early as possible, and before they hit the web application.

Affected URL & Parameter

N/A

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.155 HEARTBLEED Attack		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Upon conducting the testing of the given web application, we found a bug known as the Heartbleed bug. This bug is a severe vulnerability present in OpenSSL cryptographic software library. This bug allows leakage of sensitive information via SSL/TLS encryption. The SSL/TLS protocol was first introduced to provide better security and privacy for web applications like VPNs (Virtual Private Network), E-mail service and many more.

Risk

If this vulnerability is found in an application, an attacker might exploit this (even remotely) and can expose sensitive data about the server including user authentication credentials and secret keys that are being used. The attacker can also use this bug to read the memory of the application with any vulnerable versions of OpenSSL.

Recommendations

The Following are the recommendations that should be implemented.

Try not to use a vulnerable version of OpenSSL (eg: OpenSSL 1.0.1) and update to the latest version immediately if using.

To prevent data leakage, a developer can also use 3rd party vendors like Amazon web services.

It is better to replace the SSL key used in the application.

If the web application just faced an attack due to heartbleed, try to recommend a password change for the application's users. This step would protect their data if there were a breach in the application.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.156 DNS Rebinding		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

DNS rebinding is a type of security vulnerability that occurs when an attacker is able to use the DNS (Domain Name System) to gain unauthorized access to a victim's device. This is done by sending the victim a link that resolves to a local IP address, such as 127.0.0.1, and then using the DNS to change the IP address to a remote address that is controlled by the attacker. The victim's device then treats the remote address as a trusted local resource, allowing the attacker to gain access to it.

Risk

With DNS rebinding, the attacker cannot use the session of the connected user (as is done with CSRF attacks); the attacker can attack the server directly anyway.

Recommendations

The Following are the recommendations that should be implemented.

Implementing DNS security measures, such as DNSSEC, to prevent attackers from manipulating DNS records.

Using firewalls and other security tools to block connections to and from local IP addresses, such as 127.0.0.1, to prevent DNS rebinding attacks.

Regularly monitoring and testing your system for potential DNS rebinding vulnerabilities, and addressing any issues that are discovered.

Affected URL & Parameter

N/A

CVSS Vector

3.0 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.157 SOAP Injection		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

SOAP injection is a type of web application vulnerability that allows attackers to send malicious SOAP (Simple Object Access Protocol) messages to a web service. This can be accomplished by manipulating the SOAP messages sent to the web service, either by injecting malicious code into the message, or by manipulating the message to bypass authentication or authorization checks.

Risk

The impact of a SOAP injection attack can be severe, as it allows attackers to gain unauthorized access to sensitive information or resources, disrupt the operation of the web service, or spread malware to other users. This can lead to data breaches, financial losses, and reputational damage.

Recommendations

It is highly recommended to:-

Use parameterized queries.

Validate user input.

Use an ORM (Object-Relational Mapping) tool.

Use input filtering.

Escape special characters.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.9 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.158 No CSRF/XSRF Tokens		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

CSRF (Cross-Site Request Forgery) is a type of attack where an attacker exploits a user's web session to send unauthorized requests to a website. One way to prevent CSRF attacks is to use CSRF tokens. CSRF tokens are randomly generated tokens that are included in web forms and HTTP requests to verify that the request is coming from a legitimate source.

The absence of CSRF tokens presents a vulnerability because it allows an attacker to bypass the CSRF protection mechanism and send unauthorized requests to the website. An attacker could create a malicious web page that includes a form that posts data to the target website, and then trick the user into submitting the form.

Without a CSRF token, the target website would not be able to verify the authenticity of the request and would process it as if it came from a legitimate source. This could result in the attacker being able to perform actions on behalf of the user, such as changing their account information, making purchases, or even deleting their account.

Risk

Unauthorized actions: Without CSRF protection, an attacker could perform actions on behalf of a user without their knowledge or consent. For example, an attacker could change the user's account information, delete their account, or make unauthorized purchases.

Data theft: An attacker could use a CSRF attack to steal sensitive data, such as login credentials, credit card information, or personal information.

Malware injection: A CSRF attack could be used to inject malware into the target website, which could then infect other users who visit the site.

Reputation damage: If a successful CSRF attack leads to unauthorized actions or data theft, it could damage the reputation of the affected website and erode trust with its users.

Recommendations

The Following are the recommendations that should be implemented.

Implement CSRF protection: The most effective way to remediate this vulnerability is to implement CSRF protection mechanisms, such as using CSRF tokens. Make sure to generate random tokens for each session and include them in all forms and HTTP requests.

Review all forms and HTTP requests: Conduct a thorough review of all forms and HTTP requests in your application to ensure that CSRF tokens are included. This includes both server-side and client-side code.

Use a web application firewall: Consider using a web application firewall (WAF) that can detect and block CSRF attacks.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

Observation ID & Title	8.159 Password Reuse		
CVSS Risk Rating	Low	Status	Not Applicable

Observation Details

Reusing passwords makes it possible for a malicious agent to hack into an account to have access to others belonging to the same user. And the more a password is reused, the greater the risk of having the credentials breached.

Risk

An attacker can able to takeover the account.

Old password can be breached and known to public.

Recommendations

It is highly advised to :

- Set Enforce password history to 24. This setting will help mitigate vulnerabilities that are caused by password reuse.
- Set Maximum password age to expire passwords between 60 and 90 days. Try to expire the passwords between major business cycles to prevent work loss.
- Configure Minimum password age so that you don't allow passwords to be changed immediately.

Affected URL & Parameter

Throughout the Application

CVSS Vector

3.5 CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

9. Tools & Reference

9.1 Tools

The following tools were used for Application Security Testing

- Burp Suite Proxy
- Mozilla Web Browser with Firebug
- SQL Map
- IBM Rational AppScan
- Some other proprietary scripts and tools were also used.

9.2 References

The following tools were used for Application Security Testing

- For application security visit www.owasp.org
  - http://cert.org/other_sources/tool_sources.html
  - http://securityfocus.com/
  - http://projects.webappsec.org/w/page/13246978/Threat Classification
- Security advisories