



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE ENGINEERING AND INFORMATION SYSTEMS

Programme: B. Tech (IT)

Course Code: BITE413L

Course Name: Cyber Security

SLOT A1 + TA1

Cyber Forensics 4 VIT(CF4(VIT))

Vidya Ambatipudi – 21BIT0710 (Team Leader)

Team Members: (with register numbers)

Vidya Ambatipudi – 21BIT0710

Roshan – 21BIT0619

Poli Vardhini Reddy – 21BIT0384

P. Chaitanya Reddy – 21BIT0394

R. Vinay Kumar – 21BIT0484

Respective Faculty Name: **Prof. Gitanjali J**

CONTENTS

Chapter No	Title	Page No
1.	Introduction	3
2.	Identified Vulnerabilities with Solutions Screenshots a) Methodology Assessment Techniques: Describe the tools and methods used for identifying vulnerabilities in the system. b) Identified Vulnerabilities Overview: Summarize all identified vulnerabilities. Screenshots	3
3.	Discussion about Step-by-step process (SOP)	23
4.	Recommendations Actionable Steps and Justification	26
5.	Conclusion Summarize the key points of the report.	27
6.	Report summarization involves condensing the entire content of a report	28
7.	Add your Group Photo with Team members	29

Introduction:

In today's digital era, data breaches and cybersecurity leaks pose significant challenges, particularly for applications handling sensitive information. Hackathons like this provide an opportunity to address these pressing concerns by fostering innovative solutions to protect user data and strengthen application security. This hackathon project focuses on tackling vulnerabilities commonly seen in web and mobile applications that compromise data security. Existing solutions, like those from ManageEngine, offer basic monitoring for data leaks and web application vulnerabilities, but a more comprehensive approach is needed to build software that proactively prevents breaches across different platforms.

The project identifies and aims to mitigate key vulnerabilities, including plaintext transmission of credentials, insecure cookie settings, outdated libraries, and missing security headers. By implementing measures like encrypted transmission, secure cookie configurations, timely updates of libraries, and necessary security headers, the hackathon project seeks to create a robust application that resists common cyber threats and protects user data on web and mobile platforms.

Through this hackathon, we aim to build secure systems that not only enhance user trust but also contribute to the broader cybersecurity landscape.

Identified Vulnerabilities with Solutions

Vulnerabilities after Logging in:

Sensitive Information Transmission in Plain Text:

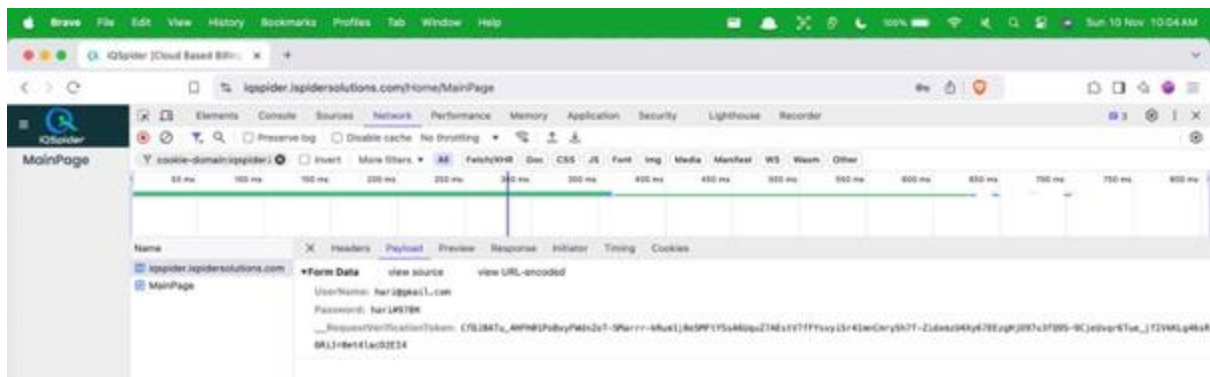
Sensitive information (username and password) is being transmitted in plain text and is visible in the Network tab of the browser's Inspect tool. Specifically, user credentials appear directly within the request payload and are unencrypted or improperly secured.

Description: Sensitive information, including username and password, is being transmitted in plain text and is visible in the Network tab of the browser's Inspect tool. This allows user credentials to be viewed directly within the request payload, as they are unencrypted or improperly secured.

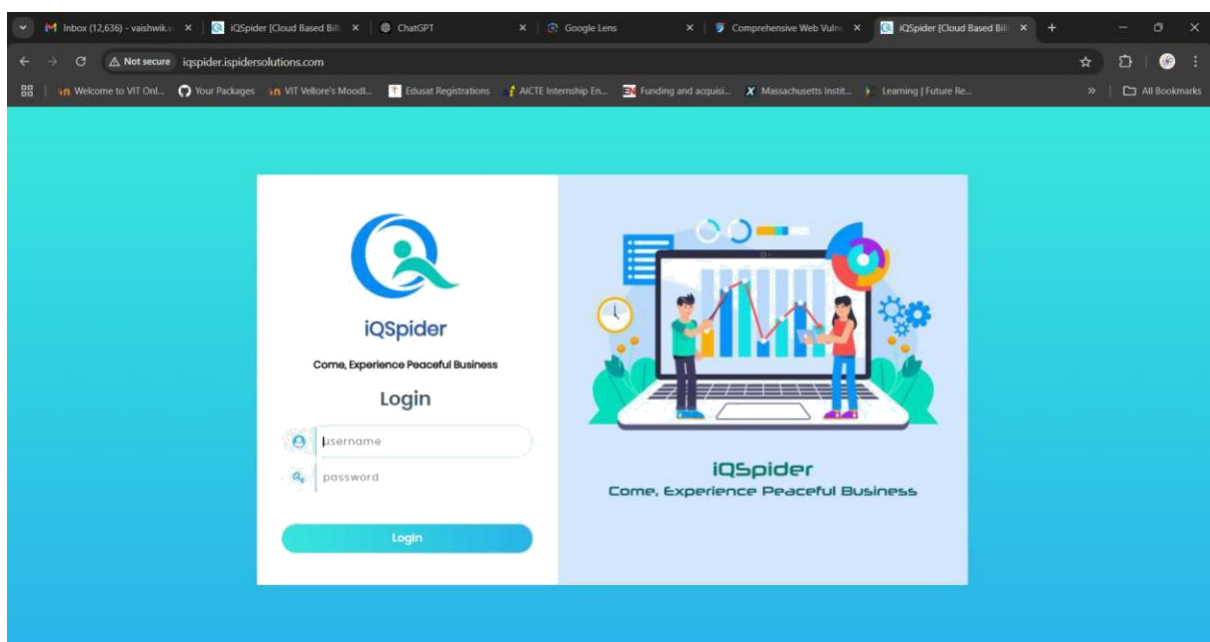
Solution:

- Implement HTTPS to encrypt all data in transit, ensuring that credentials are transmitted securely.
- Utilize Secure Sockets Layer (SSL) or Transport Layer Security (TLS) to prevent attackers from intercepting sensitive information.
- Consider using OAuth or other secure authentication methods to avoid transmitting raw credentials.

- Screenshot:



Vulnerabilities on Login page



Insecure Communication: The website uses unencrypted HTTP, making it susceptible to interception and data modification by attackers.

- Recommendation: Reconfigure the server to use HTTPS to ensure encrypted communication.
- Classification: CWE-311, OWASP Top 10 (2017: A3 - Sensitive Data Exposure, 2021: A4 - Insecure Design).

Insecure Cookie Setting (Missing Secure Flag): Cookies do not have the Secure flag, allowing potential interception and unauthorized access to sessions.

- Recommendation: Enable the Secure flag on cookies containing sensitive information.
- Classification: CWE-614, OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

Missing Security Header - Referrer-Policy: Absence of Referrer-Policy header risks inadvertent information leakage through referrer headers.

- Recommendation: Set the Referrer-Policy header to protect user privacy.
- Classification: CWE-693, OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

Missing Security Header - Content-Security-Policy: Lack of Content-Security-Policy header increases vulnerability to XSS attacks.

- Recommendation: Implement a Content-Security-Policy header for controlling resources that the page is allowed to load.
- Classification: CWE-693, OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

Missing Security Header - X-Content-Type-Options: Absence of X-Content-Type-Options header could allow content-type sniffing, leading to XSS or phishing risks.

- Recommendation: Set the X-Content-Type-Options header to nosniff.
- Classification: CWE-693, OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

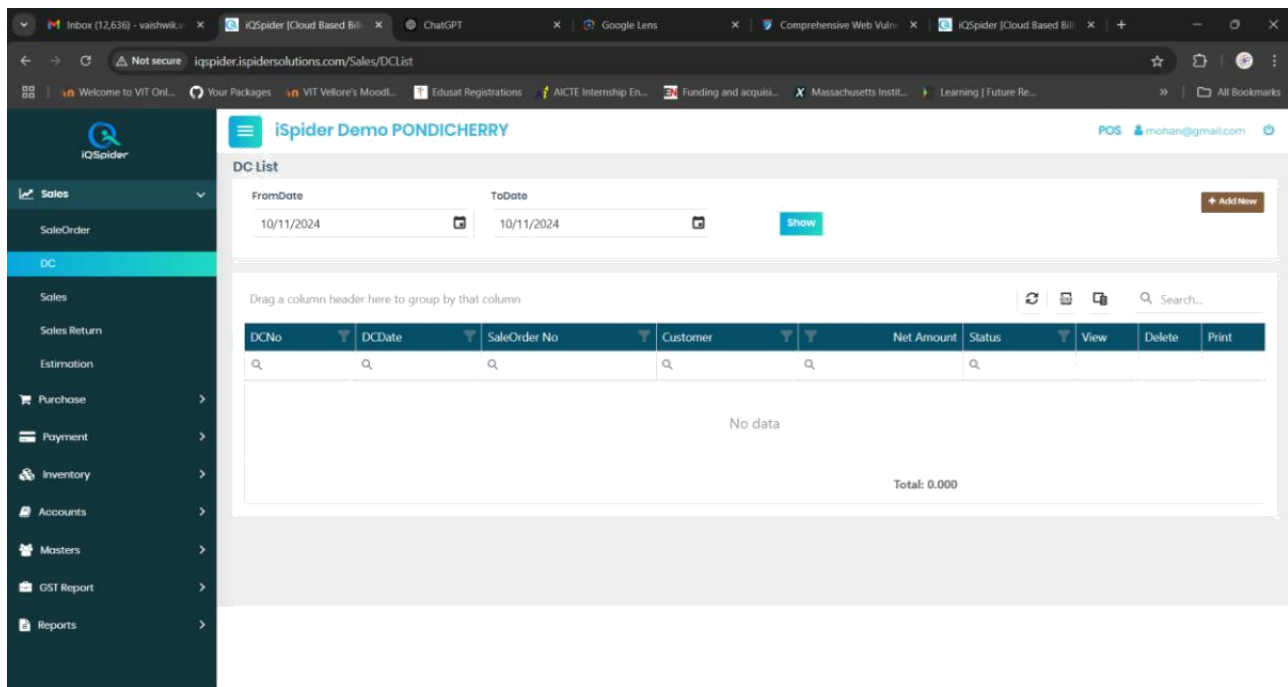
Server Software Disclosure: Information about server software and technologies (e.g., Kestrel, ASP.NET) is exposed, increasing the risk of targeted attacks.

- Recommendation: Mask server software information in HTTP headers and meta data.
- Classification: OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

Missing Security.txt File: Lack of a security.txt file, which is recommended for providing a channel for vulnerability reporting.

- Recommendation: Create a security.txt file to enable secure communication about vulnerabilities.
- Classification: OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

Vulnerabilities on DC page



Insecure Cookie Setting - Missing Secure Flag

- Description: Cookies lack the Secure flag, potentially allowing attackers to intercept user sessions.
- Recommendation: Set the Secure flag on sensitive cookies to ensure encrypted transmission.
- Classification: CWE-614, OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

Missing Security Header - X-Content-Type-Options

- Description: Lack of this header could expose the site to Cross-Site Scripting (XSS) or phishing attacks due to content-type sniffing.
- Recommendation: Set the X-Content-Type-Options header to nosniff.
- Classification: CWE-693, OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

Missing Security Header - Content-Security-Policy

- Description: Without a Content-Security-Policy header, the site is vulnerable to XSS attacks if other vulnerabilities are present.
- Recommendation: Implement a Content-Security-Policy header to restrict resources that can be loaded by the site.

- Classification: CWE-693, OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

Missing Security Header - Referrer-Policy

- Description: Absence of this header could result in sensitive URL information leakage through the Referrer header.
- Recommendation: Set the Referrer-Policy header to prevent tracking and information leakage.
- Classification: CWE-693, OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

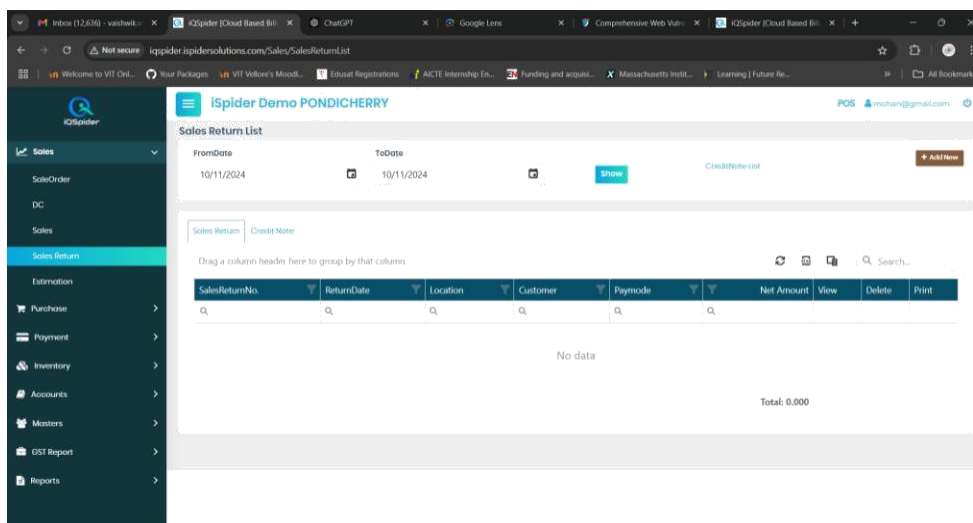
Server Software and Technology Disclosure

- Description: Server software and technology details (e.g., Kestrel, ASP.NET, jQuery) are exposed, making the site vulnerable to targeted attacks.
- Recommendation: Hide server software and technology details in HTTP headers and HTML meta tags.
- Classification: OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration). Missing

Security.txt File

- Description: Absence of a security.txt file, which is helpful for providing a standard method for reporting security issues.
- Recommendation: Implement security.txt according to standards to facilitate responsible vulnerability reporting.
- Classification: OWASP Top 10 (2017: A6, 2021: A5 - Security Misconfiguration).

Vulnerabilities on Sales Return page



Communication is Not Secure

- Description: Communication is made over unencrypted HTTP, allowing an attacker to intercept and manipulate data.
- Recommendation: Configure the server to use HTTPS for secure, encrypted communication.
- Classification: CWE-311, OWASP Top 10 (2017: A3 - Sensitive Data Exposure, 2021: A4 - Insecure Design).

Insecure Cookie Setting - Missing Secure Flag

- Description: Cookies lack the Secure flag, exposing them to interception when transmitted over HTTP.
- Recommendation: Enable the Secure flag on sensitive cookies to ensure they are only sent over HTTPS.
- Classification: CWE-614, OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Missing Security Header - X-Content-Type-Options

- Description: The absence of this header could allow content-type sniffing, leading to Cross-Site Scripting (XSS) or phishing attacks.
- Recommendation: Set the X-Content-Type-Options header to nosniff to prevent content-type sniffing.
- Classification: CWE-693, OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Missing Security Header - Referrer-Policy

- Description: Lack of Referrer-Policy header could lead to sensitive information leakage in the Referer header.
- Recommendation: Implement the Referrer-Policy header to control referrer information sent to third parties.
- Classification: CWE-693, OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Missing Security Header - Content-Security-Policy

- Description: Without a Content-Security-Policy header, the site is more vulnerable to XSS attacks.
- Recommendation: Implement a Content-Security-Policy header to control resources that can be loaded.
- Classification: CWE-693, OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Server Software and Technology Disclosure

- Description: Server software and technology details (e.g., Kestrel, jQuery) are exposed, making the site vulnerable to targeted attacks.
- Recommendation: Mask server information in HTTP headers and HTML meta tags.
- Classification: OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Missing Security.txt File

- Description: Absence of a security.txt file, which provides a standard method for reporting security issues.
- Recommendation: Implement a security.txt file to provide contact information for reporting vulnerabilities.
- Classification: OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Vulnerabilities on Purchase Order page-

The screenshot displays the 'iSpider Demo PONDICHERRY' interface for the 'Purchase Order List'. The page includes a sidebar with navigation options like Sales, Purchase, Payment, Inventory, Accounts, Masters, GST Report, and Reports. The main content area shows a 'Purchase Order List' with a filter section for 'FromDate' (10/11/2024) and 'ToDate' (10/11/2024). Below the filter is a table with the following data:

PONo	PODate	Supplier	Purchase Mode	Net Amount	Total	Status	View	Recurring PO	Delete	Print
PO0012/2025	11/10/2024	vit	Local	₹ 18,390.300	18390.3	Created				
PO0013/2025	11/10/2024	Dhruv	Local	₹ 604.800	604.8	Created				

At the bottom of the table, it shows 'TotalCount: 2' and 'Total: INR 18,995.100'.

Insecure Cookie Setting - Missing Secure Flag

- Description: Cookies lack the Secure flag, which exposes them to interception if transmitted over unencrypted HTTP.
- Recommendation: Set the Secure flag on sensitive cookies to ensure they are only sent over HTTPS.
- Classification: CWE-614, OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Unsecure Communication

- Description: Communication over HTTP is unencrypted, allowing attackers to intercept and manipulate sensitive data.
- Recommendation: Configure the server to use HTTPS to secure data transmission.
- Classification: CWE-311, OWASP Top 10 (2017: A3 - Sensitive Data Exposure, 2021: A4 - Insecure Design).

Vulnerabilities in Server-Side Software (jQuery)

- Description: jQuery version 3.1.0 is used, which has known vulnerabilities (e.g., CVE-2019-11358, CVE-2020-11023, CVE-2020-11022) allowing potential code execution.
- Recommendation: Update to the latest version of jQuery to resolve these vulnerabilities.
- Classification: CWE-1026, OWASP Top 10 (2017: A9 - Using Components with Known Vulnerabilities, 2021: A6 - Vulnerable and Outdated Components).

Missing Security Header - X-Content-Type-Options

- Description: The absence of this header could allow content-type sniffing, increasing risks of XSS or phishing attacks.
- Recommendation: Set the X-Content-Type-Options header to nosniff to prevent content-type sniffing.
- Classification: CWE-693, OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Missing Security Header - Referrer-Policy

- Description: Lack of Referrer-Policy header could result in sensitive information leakage through the Referer header.
- Recommendation: Implement the Referrer-Policy header to control referrer information sent to third parties.

- Classification: CWE-693, OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Missing Security Header - Content-Security-Policy

- Description: Without a Content-Security-Policy header, the site is more vulnerable to XSS attacks.
- Recommendation: Implement a Content-Security-Policy header to restrict resources that can be loaded by the page.
- Classification: CWE-693, OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

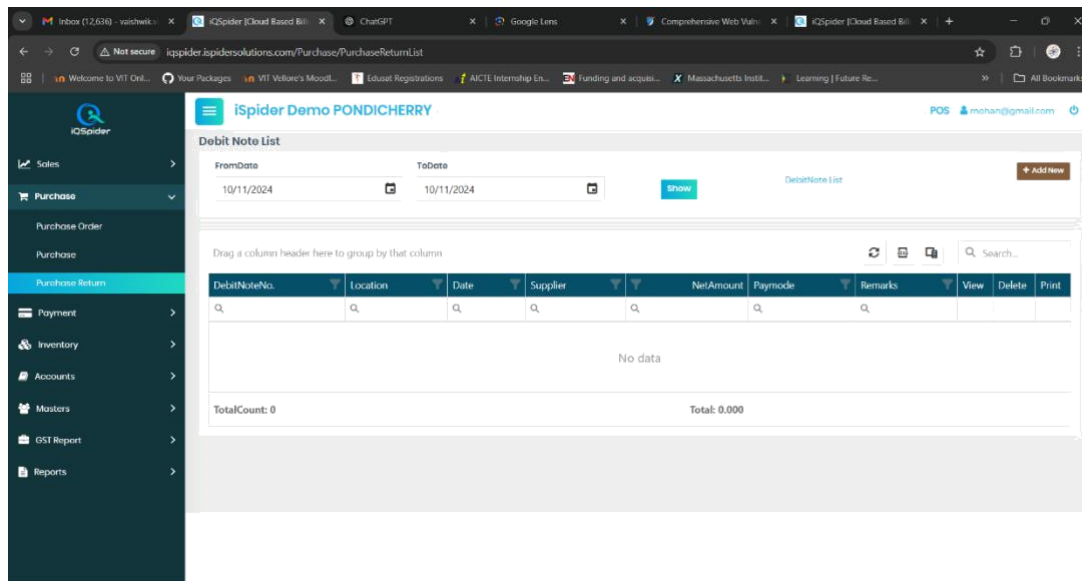
Server Software and Technology Disclosure

- Description: The server discloses software and technology details (e.g., Kestrel, jQuery), increasing risk of targeted attacks.
- Recommendation: Mask server software information in HTTP headers and HTML meta tags.
- Classification: OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Missing Security.txt File

- Description: Absence of a security.txt file, which provides a standard method for reporting security issues.
- Recommendation: Implement a security.txt file to provide contact information for reporting vulnerabilities.
- Classification: OWASP Top 10 (2017: A6 - Security Misconfiguration, 2021: A5 - Security Misconfiguration).

Vulnerabilities on Purchase Return page-



Insecure Cookie Setting: Missing Secure Flag

- Description: An attacker could intercept unencrypted communication and steal cookies, potentially gaining unauthorized access.
- Evidence: Missing Secure flag on .AspNetCore.Antiforgery.bzhrDaCLct0 cookie.
- Recommendation: Ensure cookies with sensitive data use the Secure flag to enforce encryption.
- References: OWASP
- Description: Communication over unencrypted HTTP allows attackers to intercept and modify data.
- Recommendation: Switch to HTTPS to secure data in transit.
- References: CWE-311, OWASP Top 10 2017: A3 - Sensitive Data Exposure, 2021: A4 - Insecure Design.
- Description: Lack of CSP header makes the application more vulnerable to Cross-Site Scripting (XSS) attacks.
- Recommendation: Implement a Content-Security-Policy header for each HTTP response.
- References: [OWASP Content Security Policy Cheat Sheet](#)

Missing Security Header: X-Content-Type-Options

- Description: Absence of this header increases the risk of MIME-type-based attacks.
- Recommendation: Set X-Content-Type-Options: nosniff to prevent MIME-type sniffing.
- References: [Mozilla Documentation](#)

Missing Security Header: Referrer-Policy

- Description: Without a Referrer-Policy, sensitive information could be shared in the HTTP Referer header.
- Recommendation: Set the Referrer-Policy header to no-referrer or another restrictive value.
- References: [Mozilla Documentation](#)

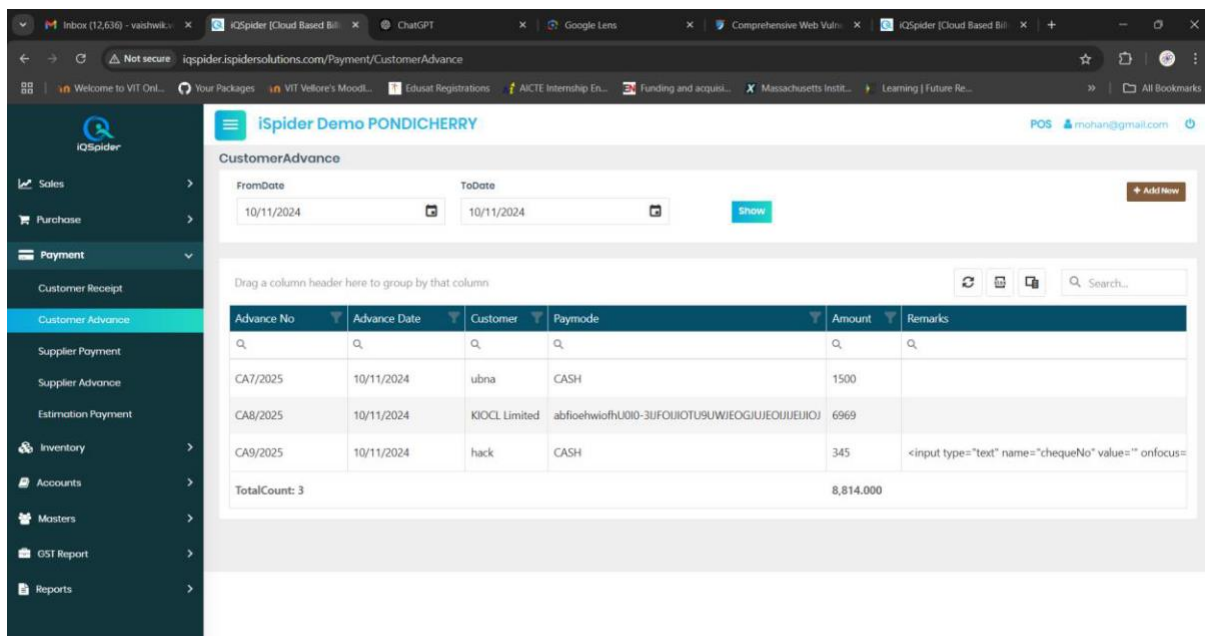
Server Software and Technology Disclosure

- Description: The server reveals details about software used (e.g., Kestrel, Bootstrap, jQuery), which attackers could leverage.
- Recommendation: Hide HTTP server headers and metadata that disclose platform and technology.
- References: OWASP

Missing Security.txt File

- Description: A missing security.txt file means there's no defined channel for reporting vulnerabilities.
- Recommendation: Create a security.txt file to improve the vulnerability reporting process.
- References: [securitytxt.org](#)

Vulnerabilities on Customer Advance page



The screenshot shows a web application interface for "iSpider Demo PONDICHERRY". The page is titled "CustomerAdvance" and features a sidebar with navigation options: Sales, Purchase, Payment (selected), Inventory, Accounts, Masters, GST Report, and Reports. The main content area displays a table of customer advances with columns: Advance No, Advance Date, Customer, Paymode, Amount, and Remarks. The table contains three rows of data and a total count of 3 advances with a total amount of 8,814.000. A search bar is visible at the top right of the table.

Advance No	Advance Date	Customer	Paymode	Amount	Remarks
CA7/2025	10/11/2024	ubna	CASH	1500	
CA8/2025	10/11/2024	KIOCL Limited	abfioehwiofhU0i0-3UFOUJOTUSUWJIEOGIUJECUJUEJIOJ	6969	
CA9/2025	10/11/2024	hack	CASH	345	<input type="text" name="chequeNo" value="" onfocus=
TotalCount: 3				8,814.000	

Detailed Findings

Insecure Cookie Setting: Missing Secure Flag

URL: <https://iqspider.ispidersolutions.com/>

Description: The Secure flag is missing from the .AspNetCore.Antiforgery cookie, making it susceptible to interception in clear-text transmission.

Recommendation: Ensure the Secure flag is set for session and sensitive cookies.

References: OWASP Guide on Cookies Attributes

Classification: CWE-614; OWASP Top 10 2017 & 2021: Security Misconfiguration

Missing Security Header: X-Content-Type-Options

URL: <https://iqspider.ispidersolutions.com/>

Description: Lack of this header can lead to MIME type security vulnerabilities like XSS or phishing attacks.

Recommendation: Set X-Content-Type-Options: nosniff to prevent browser MIME-type sniffing.

References: X-Content-Type-Options Header

Classification: CWE-693; OWASP Top 10 2017 & 2021: Security Misconfiguration

Missing Security Header: Referrer-Policy

URL: <https://iqspider.ispidersolutions.com/>

Description: Absence of Referrer-Policy header allows for potential information leakage via the Referrer header.

Recommendation: Set a Referrer-Policy header, such as no-referrer, to avoid user tracking.

References: Referrer Policy Header

Classification: CWE-693; OWASP Top 10 2017 & 2021: Security Misconfiguration

Missing Security Header: Content-Security-Policy (CSP)

URL: <https://iqspider.ispidersolutions.com/>

Description: Lack of CSP header increases vulnerability to Cross-Site Scripting (XSS) attacks.

Recommendation: Configure the Content-Security-Policy header with each HTTP response to specify allowed content sources.

References: Content-Security Policy Cheat Sheet

Classification: CWE-693; OWASP Top 10 2017 & 2021: Security Misconfiguration Server Software and Technology Information Disclosure

Details:

Technologies Detected: Kestrel (web server), Font Awesome, Bootstrap, jQuery, Popper, Sectigo (SSL), Microsoft ASP.NET, HSTS

Description: The presence of this information could help attackers exploit specific vulnerabilities related to identified software.

Recommendation: Hide software and version details in server headers and HTML meta information.

References: OWASP Guide on Server Fingerprinting

Classification: OWASP Top 10 2017 & 2021: Security Misconfiguration Missing security.txt File

URL: <https://iqspider.ispidersolutions.com/.well-known/security.txt>

Description: Lack of a security.txt file can hinder communication of vulnerabilities to the website administrators.

Recommendation: Implement a security.txt file according to standards to allow vulnerability reporting.

References: Security.txt Standard

Classification: OWASP Top 10 2017 & 2021: Security Misconfiguration

Additional Observations

No vulnerabilities were found in the following areas:

Server-side software vulnerabilities

Client access policies

robots.txt file

Untrusted certificates

HTTP debug and OPTIONS methods

Secure communication protocols

Directory listing

HTTP header Strict-Transport-Security

Domain settings for cookies

HttpOnly flag of cookies

Unsafe HTTP header CSP

Vulnerabilities on Supplier Advance page

The screenshot displays the 'Supplier Advance' page in the iSpider Demo PONDICHERRY application. The page features a sidebar with navigation options like Sales, Purchase, Payment, Inventory, Accounts, Masters, GST Report, and Reports. The main content area shows a table of supplier advances. The table has columns for Advance No, Advance Date, Supplier, Paymode, Cheque No, Amount, Remarks, View, Delete, and Print. The data rows show advances for dates 10/11/2024 with various suppliers (vit, karthiks, RAM) and amounts (123, 1000, 69, 10). The total count is 4 advances for a total amount of 1,202,000. The browser's address bar shows the URL 'iqspider.ispidersolutions.com/Payment/SupplierAdvance' and a 'Not secure' warning.

Advance No	Advance Date	Supplier	Paymode	Cheque No	Amount	Remarks	View	Delete	Print
SA2/2025	10/11/2024	vit	45	Select ChequeNo	123				
SA3/2025	10/11/2024	karthiks	CASH	Select ChequeNo	1000				
SA4/2025	10/11/2024	vit	CASH	Select ChequeNo	69	NO			
SA5/2025	10/11/2024	RAM	45	Select ChequeNo	10	NA			
TotalCount: 4					1,202,000				

1. Insecure Cookie Setting: Missing Secure Flag URL:

<https://iqspider.ispidersolutions.com/>

Cookie Name: .AspNetCore.Antiforgery.bzhrDaCLct0

Risk Description: Cookies lacking the Secure flag can be intercepted if sent over an unencrypted connection. If intercepted, this cookie could allow unauthorized access to user sessions.

Recommendation: Ensure that cookies containing sensitive information are sent with the Secure flag to enforce encrypted (HTTPS) transmission.

2. Missing Security Header: X-Content-Type-Options URL:

<https://iqspider.ispidersolutions.com/>

Risk Description: Absence of this header can lead to certain attacks, such as Cross-Site Scripting (XSS) or phishing in Internet Explorer.

Recommendation: Set the X-Content-Type-Options header to nosniff to prevent browsers from interpreting files as a different MIME type.

3. Missing Security Header: Content-Security-Policy (CSP) URL:

<https://iqspider.ispidersolutions.com/>

Risk Description: Without a CSP, the application is more vulnerable to Cross-Site Scripting (XSS) and data injection attacks.

Recommendation: Configure and apply a Content-Security-Policy header to specify which sources are considered safe for content loading.

4. Missing Security Header: Referrer-Policy URL:

<https://iqspider.ispidersolutions.com/>

Risk Description: Without a Referrer-Policy header, sensitive URL data may be inadvertently leaked to third-party sites.

Recommendation: Configure the Referrer-Policy header to no-referrer or a similar secure setting to protect against information leakage.

5. Server Software and Technology Disclosure

Software/Technologies Identified:

Kestrel (Web server) jQuery

(JavaScript library)

Bootstrap, Font Awesome (UI frameworks) Popper

(JavaScript library)

Microsoft ASP.NET (Web framework) Sectigo

(SSL/TLS certificate authority)

HSTS (Security)

Risk Description: Knowledge of server software and technologies used can aid attackers in crafting targeted attacks.

Recommendation: Conceal software platform details where possible by removing HTTP headers and meta information that reveal server, framework, and library details.

6. Missing security.txt File

Missing File: <https://iqspider.ispidersolutions.com/.well-known/security.txt>

Risk Description: While not a direct vulnerability, the absence of a security.txt file limits the ability of researchers or users to report potential security issues, reducing responsiveness to vulnerabilities.

Recommendation: Implement a security.txt file to provide a designated contact point for reporting security issues, improving defensive measures.

a) Methodology

Assessment Techniques

To identify vulnerabilities in the system, a combination of manual and automated assessment techniques were employed. The process involved using browser developer tools, automated scanners, and third-party security analysis tools to uncover security weaknesses within the application. Key methods included:

- Manual Inspection: Using the browser's Inspect tool to analyze network requests, cookies, and headers for insecure data transmissions.
- Automated Scanning: Running vulnerability scans to detect outdated libraries, missing headers, and other potential issues.
- Security Audit Frameworks: Utilizing frameworks like OWASP Top Ten as a reference to ensure all critical areas of web application security were covered.

b) Identified Vulnerabilities

Overview

Several vulnerabilities were identified that could expose sensitive information or allow unauthorized access. The following is a summary of these vulnerabilities along with solutions to address them:

1. Sensitive Information Transmission in Plain Text:

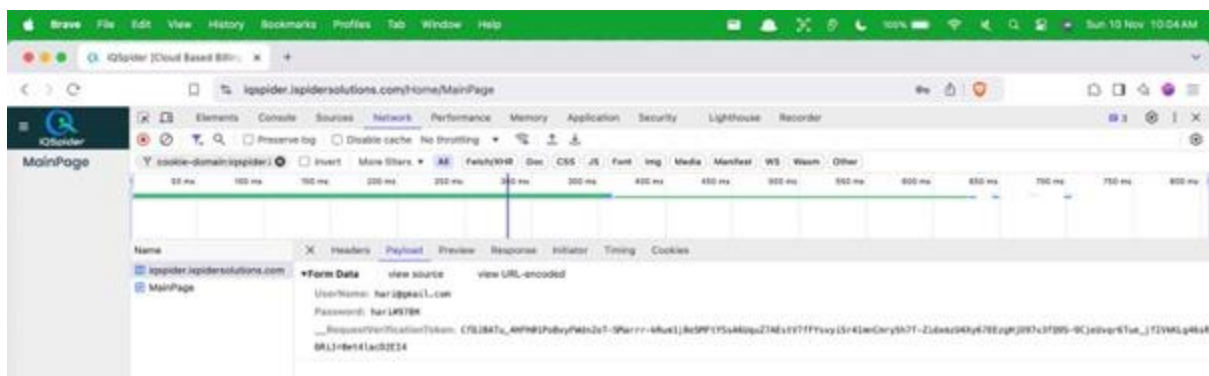
Sensitive information (username and password) is being transmitted in plain text and is visible in the Network tab of the browser's Inspect tool. Specifically, user credentials appear directly within the request payload and are unencrypted or improperly secured.

Description: Sensitive information, including username and password, is being transmitted in plain text and is visible in the Network tab of the browser's Inspect tool. This allows user credentials to be viewed directly within the request payload, as they are unencrypted or improperly secured.

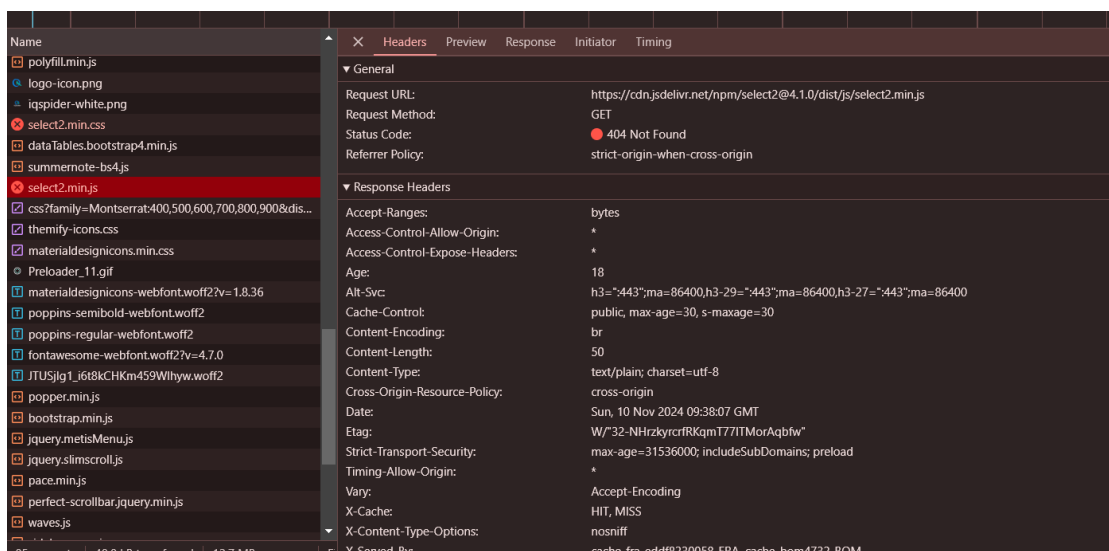
Solution:

- Implement **HTTPS** to encrypt all data in transit, ensuring that credentials are transmitted securely.
- Utilize **Secure Sockets Layer (SSL)** or **Transport Layer Security (TLS)** to prevent attackers from intercepting sensitive information.
- Consider using **OAuth** or other secure authentication methods to avoid transmitting raw credentials.

- Screenshot:



2.



This screenshot shows an HTTP 404 error for the select2.min.js file, which was requested from a CDN (Content Delivery Network) URL. The 404 error indicates that the requested file does not exist at the specified URL. This issue is not necessarily a security vulnerability by itself, but rather a resource-loading error that can lead to broken functionality in your application.

Here are potential issues and fixes:

1. Broken Functionality:

If select2.min.js is critical for certain features (such as form handling, dropdowns, etc.), its absence may break those features on your site.

Solution:

Update the URL: Check the Select2 library's latest CDN URL on their official website or GitHub repository. The file path might have changed, or the version might be outdated.

Use a Local Copy: Download select2.min.js and serve it from your local server to avoid dependency on external CDNs.

2. Security Implications:

A missing file might leave certain parts of your application vulnerable if those parts depend on select2.min.js for sanitization or validation. In some cases, attackers could exploit missing resources if they result in broken security controls.

Solution:

Regularly check for resource loading issues and ensure that all critical dependencies are properly loaded.

Use a Content Security Policy (CSP) to control which external sources your site is allowed to load resources from, reducing the risk of malicious script injection.

3. Outdated jQuery Versions (CVE-2019-11358, CVE-2020-11023, CVE-2020-11022)

Description: The application uses outdated versions of jQuery that contain vulnerabilities. These versions allow attackers to inject and execute untrusted code through DOM manipulation methods like ``.html()`` and ``.append()``, increasing the risk of Cross-Site Scripting (XSS) attacks.

Solution:

- **Update jQuery** to version 3.5.0 or later, where these vulnerabilities are addressed.
- **Regularly review and update** third-party libraries to ensure they include the latest security patches.
- If updating is not possible, sanitize untrusted data before passing it to DOM manipulation methods.

4. Insecure Cookie Setting (Missing Secure Flag)

Description: Cookies without the Secure flag can be transmitted over unencrypted HTTP connections, making them vulnerable to Man-in-the-Middle (MitM) attacks, potentially leading to unauthorized access.

Solution:

- **Set the Secure flag** on cookies containing sensitive information, so they are only sent over HTTPS.
- Configure this setting server-side, either within application settings (e.g., ASP.NET, Django) or in the server's configuration file.

5. Exposed Server Software and Technology Details

Description: The server discloses software details (such as ASP.NET, Bootstrap) in HTTP headers or HTML metadata, making it easier for attackers to target specific vulnerabilities.

Solution:

- Remove or hide server details from HTTP headers and HTML metadata.
- Configure the server to display minimal information about its software, such as setting `ServerTokens Prod` and `ServerSignature Off` in Apache.
- Use a **Web Application Firewall (WAF)** to obscure these details and block suspicious requests.

6. Missing Content-Security-Policy (CSP) Header

Description: Without a CSP header, the application is vulnerable to Cross-Site Scripting (XSS) attacks, where attackers could inject malicious JavaScript to steal user data or manipulate site behavior.

Solution:

- Add a **Content-Security-Policy (CSP) header** to your server's HTTP responses, specifying only trusted sources for scripts, styles, and other resources.
- Start with a **Report-Only CSP** to test the policy without affecting site functionality.

7. Missing Referrer-Policy Header

Description: The absence of a Referrer-Policy header means that browsers may send the full URL as the referrer, potentially leaking sensitive data when users navigate to other sites.

Solution:

- Add a **Referrer-Policy header** to control the amount of referrer information shared. Recommended settings include `no-referrer-when-downgrade` to prevent HTTPS to HTTP leaks.

8. Missing X-Content-Type-Options Header

Description: Without the X-Content-Type-Options header, some browsers might MIME-sniff content and misinterpret it, increasing the risk of XSS and phishing attacks.

Solution:

- Add the **X-Content-Type-Options header** with the value `nosniff`, ensuring browsers respect the declared content type.

9. Missing Security.txt File

Description: Lack of a security.txt file makes it difficult for security researchers to report vulnerabilities, potentially leading to missed opportunities for responsible disclosure.

Solution:

- Create a **security.txt** file at `.well-known/security.txt`, specifying contact information for reporting vulnerabilities.

Vulnerabilities found for server-side software

Risk Level	CVSS	CVE	Summary	Affected software
●	7.5	CVE-2022-31147	ReDoS vulnerability in url and URL2 validation. More details at: https://github.com/advisories/GHSA-ffmh-x56j-9rc3 https://github.com/jquery-validation/jquery-validation/commit/5bbd80d27f106c89522051a9fb0dd	Jquery-validation 1.15.1
●	6.4	CVE-2024-6531	Bootstrap Cross-Site Scripting (XSS) vulnerability. More details at: https://github.com/advisories/GHSA-vc8w-jr9v-vj7f https://nvd.nist.gov/vuln/detail/CVE-2024-6531 https://github.com/rubysec/ruby-advisory-db/blob/master/gems/bootstrap/CVE-2024-6531.yml https://github.com/twbs/bootstrap https://www.herodevs.com/vulnerability-directory/cve-2024-6531	Bootstrap 4.1.3
●	5	CVE-2021-21252	Regular Expression Denial of Service vulnerability. More details at: https://github.com/jquery-validation/jquery-validation/blob/master/changelog.md#1193--2021-01-09	Jquery-validation 1.15.1
●	5	CVE-2021-43306	ReDoS vulnerability in URL2 validation. More details at: https://github.com/jquery-validation/jquery-validation/blob/master/changelog.md#1194--2022-05-19	Jquery-validation 1.15.1
●	4.3	CVE-2019-11358	jQuery before 3.4.0, as used in Drupal, Backdrop CMS, and other products, mishandles jQuery.extend(true, {}, ...) because of Object.prototype pollution. More details at: https://blog.jquery.com/2019/04/10/jquery-3-4-0-released/ https://github.com/jquery/jquery/commit/753d591aea698e57d6db58c9f722cd0808619b1b https://nvd.nist.gov/vuln/detail/CVE-2019-11358	Jquery 3.1.0
●	4.3	CVE-2020-11023	passing HTML containing <option> elements from untrusted sources - even after sanitizing it - to one of jQuery's DOM manipulation methods (i.e. .html(), .append(), and others) may execute untrusted code. More details at: https://blog.jquery.com/2020/04/10/jquery-3-5-0-released/	Jquery 3.1.0
●	4.3	CVE-2020-11022	Regex in its jQuery.htmlPrefilter sometimes may introduce XSS. More details at: https://blog.jquery.com/2020/04/10/jquery-3-5-0-released/	Jquery 3.1.0
●	4.3	CVE-2019-8331	XSS in data-template, data-content and data-title properties of tooltip/popover. More details at: https://github.com/advisories/GHSA-9v3m-8fp8-mj99 https://github.com/twbs/bootstrap/issues/28236	Bootstrap 4.1.3
●	N/A	N/A	Potential XSS via showLabel. More details at: https://github.com/jquery-validation/jquery-validation/blob/master/changelog.md#1200--2023-10-10	Jquery-validation 1.15.1

Risk description:

The risk is that an attacker could search for an appropriate exploit (or create one himself) for any of these vulnerabilities and use it to attack the system.

Recommendation:

To eliminate the risk of these vulnerabilities, we recommend you check the installed software version and upgrade to the latest version.

Discussion: Step-by-Step Process (Standard Operating Procedure - SOP)

The following SOP outlines a systematic approach to identifying and addressing vulnerabilities in web and mobile applications. This step-by-step process ensures thorough assessment, mitigation, and ongoing monitoring of security risks to prevent data breaches and cybersecurity leaks.

Step 1: Initial Security Assessment

1. Identify Critical Components:

- Begin by mapping out critical parts of the application, especially those that handle sensitive information, such as login, registration, and data storage modules.
- Prioritize areas that impact user data and authentication processes.

2. Gather Tools and Resources:

- Use tools such as browser developer tools, automated scanners (e.g., OWASP ZAP, Burp Suite), and library checkers to assist in identifying vulnerabilities.
- Refer to security guidelines like the **OWASP Top Ten** to ensure comprehensive coverage.

3. Document Current Configuration:

- Take note of the current versions of libraries and frameworks in use, cookie settings, HTTP headers, and other security-related configurations.
- This documentation serves as a baseline for identifying changes needed to improve security.

Step 2: Vulnerability Detection

1. Inspect Network Traffic:

- Use the **Network tab in the browser's developer tools** to inspect data transmission. Check for unencrypted data, such as credentials sent in plaintext.
- Screenshot evidence, like the plaintext transmission screenshot, can help illustrate specific vulnerabilities.

2. Analyze Cookies and Security Headers:

- Check for the presence of essential headers such as **Content-Security-Policy (CSP)**, **X-Content-Type-Options**, and **Referrer-Policy**.
- Evaluate cookie settings to ensure that sensitive cookies (e.g., session tokens) are marked as `Secure` and `HttpOnly` to prevent unauthorized access.

3. Run Automated Scans:

- Use automated vulnerability scanners to detect outdated libraries, such as jQuery versions, and identify other known issues.
- Address flagged vulnerabilities, especially those that have been assigned Common Vulnerabilities and Exposures (CVEs), which indicate known weaknesses.

Step 3: Vulnerability Analysis and Documentation

1. Review Vulnerabilities:

- Evaluate each detected vulnerability and determine its impact on data security and user privacy. For example, understand how missing security headers or insecure cookies could lead to attacks like **Cross-Site Scripting (XSS)** or **Man-in-the-Middle (MitM)** attacks.

2. Document Vulnerabilities:

- Create a detailed report of each identified vulnerability, including its description, risk level, and potential consequences.
- Attach screenshots for critical vulnerabilities (like unencrypted credentials) to provide visual evidence.

3. Prioritize Vulnerabilities:

- Assign priority levels based on the severity and impact of each vulnerability, focusing on those that could lead to data breaches or unauthorized access.

Step 4: Implement Security Solutions

1. Apply Encryption and Secure Transmission:

- Implement **HTTPS** to encrypt all data transmissions, including user credentials. This step prevents sensitive data from being exposed in plaintext.
- Configure **SSL/TLS certificates** to secure the connection and ensure that all data exchanged between the server and client is encrypted.

2. Update Outdated Libraries:

- Replace outdated versions of libraries (like jQuery) with secure, patched versions to prevent vulnerabilities related to untrusted code execution.
- Regularly review and update third-party libraries to keep the application secure against known CVEs.

3. Configure Secure Headers and Cookies:

- Add security headers such as **Content-Security-Policy (CSP)**, **X-Content-Type-Options**, and **Referrer-Policy** to mitigate risks of XSS attacks and data leakage.
- Set cookies with `'Secure'`, `'HttpOnly'`, and `'SameSite'` attributes, especially for session cookies, to protect them from MitM attacks and unauthorized access.

4. Implement Additional Security Measures:

- Create a **security.txt** file to allow responsible reporting of vulnerabilities by security researchers.
- Configure the server to obscure version and technology details, which reduces the risk of targeted attacks.

Step 5: Testing and Verification

1. Conduct Security Testing:

- After implementing solutions, perform **penetration testing** to validate the security measures. Test each fixed vulnerability to confirm it has been adequately addressed.
- Use tools like **Burp Suite** or **OWASP ZAP** to simulate attacks and ensure that the fixes hold up under simulated threats.

2. Validate Security Headers and Configurations:

- Use online tools like **securityheaders.com** to verify that all necessary security headers are set and correctly configured.
- Double-check that cookies with sensitive information are now being sent only over HTTPS.

3. Document Changes and Results:

- Update documentation to reflect all changes made, including the upgraded libraries, new headers, and security settings.
- Capture screenshots of the updated configurations and successful test results for reference.

Step 6: Ongoing Monitoring and Maintenance

1. Set Up Continuous Monitoring:

- Implement monitoring tools to track security logs and receive alerts for suspicious activities, such as repeated failed login attempts or unusual data access patterns.
- Regularly audit network traffic and cookie configurations to ensure secure data handling.

2. Schedule Regular Security Audits:

- Perform periodic security audits to identify new vulnerabilities or outdated libraries, ensuring the application remains secure over time.
- Re-run vulnerability assessments as new CVEs and security standards emerge.

3. Update Security Policies and Training:

- Update internal security policies to reflect best practices, such as encryption requirements, secure cookie settings, and version control for libraries.
- Educate the development team about new vulnerabilities and preventive measures to ensure a proactive approach to security.

Recommendations Actionable Steps and Justification

1. Encrypt Sensitive Data in Transit

- **Action:** Implement HTTPS across the application to ensure all data, especially sensitive information like usernames and passwords, is encrypted during transmission.
- **Justification:** Transmitting sensitive information in plain text makes it vulnerable to interception. Using HTTPS prevents attackers from reading data in transit, safeguarding user credentials and other private data from Man-in-the-Middle (MitM) attacks.

2. Update Outdated Libraries and Frameworks

- **Action:** Upgrade libraries, particularly jQuery, to the latest stable versions. Regularly review and update all third-party libraries in use.
- **Justification:** Outdated libraries may contain vulnerabilities (such as Cross-Site Scripting in jQuery) that attackers can exploit. Updating libraries ensures that these known vulnerabilities are patched, reducing the attack surface.

3. Implement Secure Cookie Settings

- **Action:** Set cookies with the Secure, HttpOnly, and SameSite attributes. Ensure that cookies containing sensitive information, such as session identifiers, are transmitted over HTTPS only.
- **Justification:** Cookies without these flags are more susceptible to MitM attacks and can be accessed by client-side scripts. Properly configured cookies prevent unauthorized access and reduce the risk of session hijacking.

4. Add and Configure Security Headers

- **Action:** Set security headers, including **Content-Security-Policy (CSP)**, **X-Content-Type-Options**, **Referrer-Policy**, and **X-Frame-Options** to restrict resource loading, prevent MIME-type sniffing, control referrer data, and avoid clickjacking.
- **Justification:** Security headers help enforce rules in the browser, limiting exposure to XSS, data leakage, and other vulnerabilities. By adding these headers, the application can mitigate several client-side security risks effectively.

5. Limit Exposure of Server and Technology Details

- **Action:** Hide software versions and technology details in HTTP headers and HTML metadata to avoid giving attackers useful information.
- **Justification:** Revealing server details, such as the software version, can make it easier for attackers to exploit specific vulnerabilities. Concealing this information helps prevent targeted attacks based on known vulnerabilities.

6. Implement Content-Security-Policy (CSP) Gradually

- **Action:** Start with the Content-Security-Policy-Report-Only header to test and monitor CSP rules before enforcing them with Content-Security-Policy.
- **Justification:** Gradually implementing CSP prevents disruptions to legitimate resources while allowing time to refine the policy. Once tested, enforcing CSP helps protect against XSS and data injection attacks.

7. Create a security.txt File for Responsible Disclosure

- **Action:** Create a security.txt file with contact information for reporting security vulnerabilities and place it in the .well-known/ directory.
- **Justification:** A security.txt file encourages responsible disclosure, providing security researchers with a clear channel to report vulnerabilities. This step helps improve application security by facilitating proactive vulnerability reporting.

8. Conduct Regular Security Audits and Vulnerability Scanning

- **Action:** Set up a routine for security audits, vulnerability scanning, and penetration testing to identify new risks and ensure ongoing compliance with security best practices.
- **Justification:** Regular audits are essential for maintaining security as new vulnerabilities emerge. Continuous monitoring and scanning can detect issues before they become major threats, ensuring the application remains secure over time.

Conclusion Summarize the key points of the report

This report has examined critical security vulnerabilities identified in the application and provided actionable recommendations to address them. The key vulnerabilities, including transmission of sensitive information in plain text, outdated libraries, insecure cookie settings, and missing security headers, pose significant risks of data breaches and unauthorized access.

To mitigate these risks, implementing HTTPS for secure data transmission, updating outdated libraries, configuring secure cookie settings, and adding essential security headers such as Content-Security-Policy and X-Content-Type-Options are crucial steps. Furthermore, concealing server details, gradually enforcing CSP, creating a 'security.txt' file for responsible disclosure, and conducting regular security audits will strengthen the application's resilience against cyber threats.

By applying these recommendations, the application will be better prepared to prevent data breaches, protect user privacy, and meet industry security standards, ultimately fostering a safer digital environment for all users.

Report summarization involves condensing the entire content of a report

This report provides an in-depth security assessment of an application, identifying key vulnerabilities and proposing targeted solutions to strengthen cybersecurity defenses. Major vulnerabilities include the unencrypted transmission of sensitive information, outdated libraries prone to security risks, insecure cookie configurations, lack of critical security headers, and exposure of server and technology details that attackers could exploit.

To address these issues, the report recommends:

1. **Encrypting Data in Transit:** Enforce HTTPS to secure the transmission of sensitive data like usernames and passwords.
2. **Updating Libraries:** Ensure that third-party libraries, such as jQuery, are current to mitigate known vulnerabilities.
3. **Enhancing Cookie Security:** Use Secure, HttpOnly, and SameSite attributes to protect cookies from interception and client-side access.
4. **Adding Security Headers:** Configure headers such as Content-Security-Policy (CSP), X-Content-Type-Options, Referrer-Policy, and X-Frame-Options to prevent Cross-Site Scripting (XSS), data leakage, and other attacks.
5. **Limiting Server Information Exposure:** Conceal server and technology details to reduce targeted attack risks.
6. **Implementing a Gradual CSP:** Introduce the CSP policy in reporting mode initially to avoid blocking legitimate resources, then enforce it once refined.
7. **Creating a security.txt File:** Provide a contact point for responsible disclosure to facilitate vulnerability reporting.
8. **Conducting Regular Security Audits:** Establish a routine for vulnerability scanning and penetration testing to detect and address new risks.

The report concludes that by following these recommendations, the application will be better equipped to prevent data breaches, safeguard sensitive user information, and enhance its overall cybersecurity posture, reducing risks for both users and stakeholders.

Add your Group Photo with Team members

