AI-Augmented Web Application Security Assessment

Target: https://www.indusrangers.com/

Report Date: 2025-06-25

Scan Date: 2025-06-25

Report Version: 1.0

Generated by: AutoPent.AI Security Assistant

Classification: CONFIDENTIAL

Distribution: Internal Use Only

Executive Summary

This report presents the findings of an automated web application security assessment conducted using AI-augmented analysis. The assessment identified **6** potential security vulnerabilities across the target application.

Risk Distribution:
• High Risk: 1 vulnerabilities
• Medium Risk: 2 vulnerabilities
• Low Risk: 3 vulnerabilities
• Informational: 0 findings

Immediate Actions Required:
1. Address all high-risk vulnerabilities within 24-48 hours
2. Implement input validation and output encoding
3. Review and update security configurations
4. Conduct regular security assessments

Vulnerability Summary

Vulnerability Name	Risk Level	Confidence	URL	CWE ID
Missing Security Header: Content-Securit	Medium	High	https://www.indusrangers.com/	N/A
Missing Security Header: X-Frame-Options	Medium	High	https://www.indusrangers.com/	N/A
Missing Security Header: Referrer-Policy	Low	High	https://www.indusrangers.com/	N/A
Information Disclosure - Server Header	Low	High	https://www.indusrangers.com/	N/A
SSL Certificate Name Mismatch	High	High	https://www.indusrangers.com/	N/A
Inline JavaScript Detected	Low	High	https://www.indusrangers.com/	N/A

Detailed Findings

SSL Certificate Name Mismatch [High Risk]

Property	Value
Risk Level	High
Confidence	High
CWE ID	Not specified
Affected URL	https://www.indusrangers.com/
Parameter	

Description:

SSL Certificate Name Mismatch occurs when the common name specified in the SSL certificate does not match the domain it is being used for. This misconfiguration can lead to security warnings in browsers and potential interception of sensitive data.

Impact:

This vulnerability is dangerous as it can allow attackers to perform man-in-the-middle attacks, intercepting and potentially altering the communication between the user and the website. This can lead to unauthorized access to sensitive information such as login credentials, financial data, or personal details, resulting in reputation damage, financial loss, and legal implications for the affected organization.

Remediation:

To fix SSL Certificate Name Mismatch: 1. Obtain a new SSL certificate with the correct common name matching the domain. 2. Install the new SSL certificate on the web server. 3. Ensure proper configuration of the web server to use the new SSL certificate. 4. Test the SSL certificate installation using online tools to verify the common name matches the domain.

Prevention:

1. Regularly monitor SSL certificate expiration dates and ensure they are correctly configured. 2. Implement automated tools to detect misconfigured SSL certificates and domain mismatches. 3. Enforce strict policies for SSL certificate management and deployment processes. 4. Educate staff responsible for SSL certificate management on best practices to avoid misconfigurations.

Missing Security Header: Content-Security-Policy [Medium Risk]

Property	Value
Risk Level	Medium
Confidence	High
CWE ID	Not specified
Affected URL	https://www.indusrangers.com/
Parameter	

Description:

What is this vulnerability? The missing security header "Content-Security-Policy" vulnerability occurs when a web application fails to implement a Content Security Policy (CSP) header. CSP is an added layer of security that helps prevent various types of attacks, such as cross-site scripting

(XSS) and data injection. **Why is it dangerous?** Without a Content-Security-Policy header, the web application is more vulnerable to XSS attacks, data injection, clickjacking, and other malicious acti...

Impact:

Without a Content-Security-Policy header, the web application is more vulnerable to XSS attacks, data injection, clickjacking, and other malicious activities. Attackers could exploit this vulnerability to execute arbitrary code, steal sensitive data, deface the website, or redirect users to malicious sites, leading to reputational damage, data breaches, and financial loss.

Remediation:

To fix the missing "Content-Security-Policy" header vulnerability, you need to implement a Content Security Policy for the web application. Here are the steps to add a basic CSP header: 1. Open the web server configuration file. 2. Add the following Content-Security-Policy header: ``` Content-Security-Policy: default-src 'self'; ``` 3. Test the CSP header using online tools like securityheaders.com or CSP evaluator. 4. Adjust the policy directives based on the application's requirements to balance security and functionality.

Prevention:

2. **Regular Security Headers Audit:** Conduct periodic audits to ensure all necessary security headers, including CSP, are correctly implemented and configured. 3. **Use Security Headers Middleware:** Utilize security header middleware or plugins in web servers or frameworks to automate the enforcement of security headers, including CSP. 4. **Stay Informed:** Stay updated on the latest web security best practices and vulnerabilities to proactively protect your web application from emerging threats.

Missing Security Header: X-Frame-Options [Medium Risk]

Property	Value
Risk Level	Medium
Confidence	High
CWE ID	Not specified
Affected URL	https://www.indusrangers.com/
Parameter	

Description:

What is this vulnerability? The Missing Security Header: X-Frame-Options vulnerability occurs when a web application fails to include the X-Frame-Options header in its HTTP responses. This header helps prevent clickjacking attacks by specifying whether a browser should be allowed to render a page in a frame or iframe. **Why is it dangerous?** Without the X-Frame-Options header, an attacker can potentially load the vulnerable web application within a malicious frame on another site. This cou...

Impact:

Without the X-Frame-Options header, an attacker can potentially load the vulnerable web application within a malicious frame on another site. This could lead to clickjacking attacks where users unknowingly interact with the vulnerable site, enabling the attacker to perform actions on behalf of the user, such as stealing sensitive information or executing unauthorized transactions.

Remediation:

To fix this vulnerability, the web application should include the X-Frame-Options header in its HTTP responses. This header can have one of three values: DENY, SAMEORIGIN, or ALLOW-FROM uri.

Here's an example of how to set the X-Frame-Options header to DENY using Apache configuration: ``` Header always append X-Frame-Options DENY ``` For Nginx, you can set the header like this: ``` add_header X-Frame-Options DENY; ``` Ensure the header is included in all responses from the application to protect against clickjacking attacks.

Prevention:

1. Implement Content Security Policy (CSP) headers to control which resources can be loaded on your web application, further enhancing security. 2. Regularly scan your web application using security tools to identify and address missing security headers like X-Frame-Options. 3. Educate developers on secure coding practices and the importance of including security headers in web applications. 4. Stay informed about emerging threats and security best practices to continuously improve your web application's security posture.

Missing Security Header: Referrer-Policy [Low Risk]

Property	Value
Risk Level	Low
Confidence	High
CWE ID	Not specified
Affected URL	https://www.indusrangers.com/
Parameter	

Description:

The Missing Security Header: Referrer-Policy vulnerability occurs when the Referrer-Policy header is not set in the web application's HTTP response. This header controls how much information about the origin of the request is included in the Referer header.

Impact:

Without setting the Referrer-Policy header, the web application is vulnerable to information leakage, potentially exposing sensitive data in the Referer header. Attackers could exploit this to track users' browsing behavior, gather sensitive information, or launch targeted attacks based on the leaked data.

Remediation:

Remediation steps not available

Prevention:

``` Referrer-Policy: strict-origin-when-cross-origin ``` 1. Implement a Content Security Policy (CSP) to control which external resources can be loaded by the web application. 2. Regularly scan and audit HTTP response headers to ensure all necessary security headers are properly configured. 3. Educate developers on secure coding practices and the importance of setting security headers in web applications. 4. Utilize automated tools or security scanners to detect and flag missing security headers during development and deployment stages.

Information Disclosure - Server Header [Low Risk]

Property	Value
Risk Level	Low
Confidence	High
```

CWE ID	Not specified
Affected URL	https://www.indusrangers.com/
Parameter	

Description:

The Information Disclosure - Server Header vulnerability occurs when the server header in HTTP responses reveals specific information about the technology stack being used by the web server. This information can include the server software name and version, potentially aiding attackers in crafting targeted attacks against known vulnerabilities in that software.

Impact:

While this vulnerability is categorized as low risk, it can still provide valuable information to attackers. By knowing the server software and version, attackers can tailor their attacks to exploit known vulnerabilities associated with that specific software version. This could lead to unauthorized access, data breaches, or disruption of services, impacting the confidentiality, integrity, and availability of the web application.

Remediation:

To remediate the Information Disclosure - Server Header vulnerability, you can configure the web server to suppress or modify the server header information. Below are steps to mitigate this issue for Apache and Nginx servers: For Apache: 1. Edit the Apache configuration file (httpd.conf or apache2.conf). 2. Add or modify the following line to remove the server signature: ``` ServerSignature Off ServerTokens Prod ``` 3. Restart the Apache service for changes to take effect. For Nginx: 1. Edit the Nginx configuration file (nginx.conf). 2. Add or modify the following line to hide the server version: ``` server_tokens off; ``` 3. Reload the Nginx configuration for changes to be applied.

Prevention:

2. **Implement web application firewalls (WAF):** Deploy a WAF to filter and monitor HTTP traffic, helping to detect and block malicious requests targeting server headers. 3. **Security headers:** Utilize security headers like Content-Security-Policy (CSP) and X-Frame-Options to enhance overall security posture and mitigate various web application vulnerabilities. 4. **Security testing:** Conduct regular security assessments, including vulnerability scanning and penetration testing, to identify and address any security weaknesses, such as information disclosure in server headers.

Inline JavaScript Detected [Low Risk]

Property	Value
Risk Level	Low
Confidence	High
CWE ID	Not specified
Affected URL	https://www.indusrangers.com/
Parameter	

Description:

Inline JavaScript Detected is a security vulnerability where JavaScript code is embedded directly within the HTML content of a web page. This can occur when developers include JavaScript code within the HTML markup instead of using external script files or event handlers.

Impact:

This vulnerability is considered low risk because it does not directly lead to code execution or data leakage. However, having multiple inline JavaScript blocks can make the codebase harder to maintain and debug. An attacker could potentially exploit this by injecting malicious scripts into the existing JavaScript blocks, leading to cross-site scripting (XSS) attacks.

Remediation:

1. Refactor Inline JavaScript: Move all inline JavaScript code to external script files and reference them using the `` tag. 2. Use Event Handlers: Instead of inline event attributes like `onclick`, attach event handlers programmatically using JavaScript. 3. Content Security Policy (CSP): Implement a strict CSP to restrict the execution of inline scripts and mitigate the risk of XSS attacks. 4. Code Reviews and Static Analysis: Regularly review the codebase to identify and remove any remaining inline JavaScript blocks.

Prevention:

1. Separation of Concerns: Encourage developers to separate HTML, CSS, and JavaScript code for better maintainability and security. 3. Use Libraries and Frameworks: Utilize modern JavaScript frameworks and libraries that promote best practices for handling client-side code securely. 4. Security Testing: Conduct regular security assessments, including automated scans and manual reviews, to identify and address vulnerabilities proactively.

Recommendations

General Security Recommendations:

1. Implement a comprehensive input validation framework
2. Deploy proper output encoding mechanisms
3. Enable security headers (CSP, HSTS, X-Frame-Options)
4. Conduct regular security code reviews
5. Implement automated security testing in CI/CD pipeline
6. Establish an incident response plan
7. Provide security training for development team

Recommended Remediation Timeline:
- **Critical/High Risk:** Immediate action required (24-48 hours)
- **Medium Risk:** Address within 1-2 weeks
- **Low Risk:** Include in next maintenance cycle
- **Informational:** Monitor and review during next assessment