infiltra.ai

Penetration Testing Report for

Visitormanager.acme.com

Vulnerability Assessment Date

Saturday, February 21st, 2026

Table of CONTENTS

1. EXECUTIVE SUMMARY

Introduction

This report highlights potential security vulnerabilities identified during a recent scan of the application. Addressing these issues will significantly improve the application's security posture and reduce the risk of potential attacks. The vulnerabilities identified range from missing security headers to the presence of vulnerable JavaScript libraries and exposed hidden files. Immediate action is recommended to remediate these findings, focusing on the high-risk issues first.

The objective of this assessment is to evaluate the overall security posture of **Visitormanager.acme.com -** https://visitormanager.acme.com while identifying technical risks that the application and the API endpoints may be exposed to. The testing was done in a time-boxed manner and adopted a sample-based approach for the testing process.

A globally accepted Web Application Security Framework - The OWASP Top 10 (Open Web Application Security Project) was utilized as the underlying testing framework for the security assessment.

Assessment Methodology

- The application was tested from an external system as a a grey box test with customer provided credentials, using Infiltra.ai’s automated framework and AI agents

Key Findings

The security assessment of the web application identified a total of 13 vulnerabilities, categorized into 3 High, 4 Medium, and 6 Low risk issues. Critical findings indicate significant weaknesses in the application's authorization mechanisms (Vertical Privilege Escalation) and file upload validation, potentially exposing the Java-based backend to Remote Code Execution (RCE) and unauthorized administrative access. Additionally, the React frontend utilizes vulnerable JavaScript libraries, increasing the attack surface. Several security misconfigurations were also noted in the Nginx web server, specifically regarding missing HTTP security headers (CSP, HSTS, Clickjacking protection). Immediate remediation is recommended for the High-risk vulnerabilities to prevent data breaches and system compromise.

#	Vulnerability Details	Risk	Description
1	Access Control Issue - Improper Authorization	High	A Vertical Privilege Escalation vulnerability exists where a standard user (e.g., 'readOnly' role) can access administrative functions. The Java backend fails to verify user roles effectively before granting access to sensitive endpoints like /admin/dashboard.
2	Unrestricted Upload of File with Dangerous Type	High	The application allows file uploads without sufficient validation of the file content. Attackers can bypass extension checks to upload malicious scripts (e.g., web shells), leading to Remote Code Execution.
3	Vulnerable JS Library	High	The React frontend is using JavaScript libraries with known security vulnerabilities (CVE-2025-27152, CVE-2025-58754, etc.). These outdated components can be exploited to compromise client-side security.
4	Access Control Issue - Improper Authorization	Medium	An unauthenticated user can access restricted resources. The application fails to enforce authentication checks on specific directories or functions, allowing direct browsing to sensitive URLs.

#	Vulnerability Details	Risk	Description
5	Content Security Policy (CSP) Header Not Set	Medium	The 'Content-Security-Policy' HTTP header is missing. This header is crucial for preventing Cross-Site Scripting (XSS) and data injection attacks by defining which dynamic resources are allowed to load.
6	Missing Anti-clickjacking Header	Medium	The application response is missing 'X-Frame-Options' or CSP 'frame-ancestors' directives. This allows the site to be framed by attackers, facilitating Clickjacking attacks.
7	Sub Resource Integrity Attribute Missing	Medium	External scripts (likely from CDNs or Google Fonts) are loaded without the 'integrity' attribute. If the external source is compromised, malicious code could be injected into the application.
8	Application Error Disclosure	Low	The application displays detailed error messages or stack traces to the user. This may reveal internal file paths, library versions, or logic flaws.
9	Private IP Disclosure	Low	The HTTP response body contains private IP addresses or internal hostnames (e.g., 10.x.x.x). This often happens in debug comments or default server error pages.
10	Server Leaks Version Information via "Server" HTTP Response Header Field	Low	The web server reveals its specific version in the 'Server' HTTP header (e.g., Nginx/1.18.0). This helps attackers identify specific vulnerabilities associated with that version.
11	Strict-Transport-Security Header Not Set	Low	The HSTS header is missing, meaning the browser is not forced to use HTTPS for future requests. Although the tech stack claims HSTS is implemented, the scan

#	Vulnerability Details	Risk	Description
			indicates it is missing on specific responses.
12	Timestamp Disclosure - Unix	Low	A Unix timestamp was found in the response. While often benign, precise timestamps can sometimes be used to predict random number generation seeds or map server-side operations.
13	X-Content-Type-Options Header Missing	Low	The 'X-Content-Type-Options' header is not set to 'nosniff'. This allows browsers to MIME-sniff the response, potentially executing non-executable files as scripts.

Recommendations

Test Recommendation. Holistic security is achieved through the combination of a well-designed application architecture, secure coding practices, appropriate configuration of underlying infrastructure and multiple layers of defense (Defense-In-Depth). New attack techniques are developed on a regular basis that will affect the security of all applications. Hence it is recommended to schedule regular security assessments to cover such new attacks and vulnerabilities.

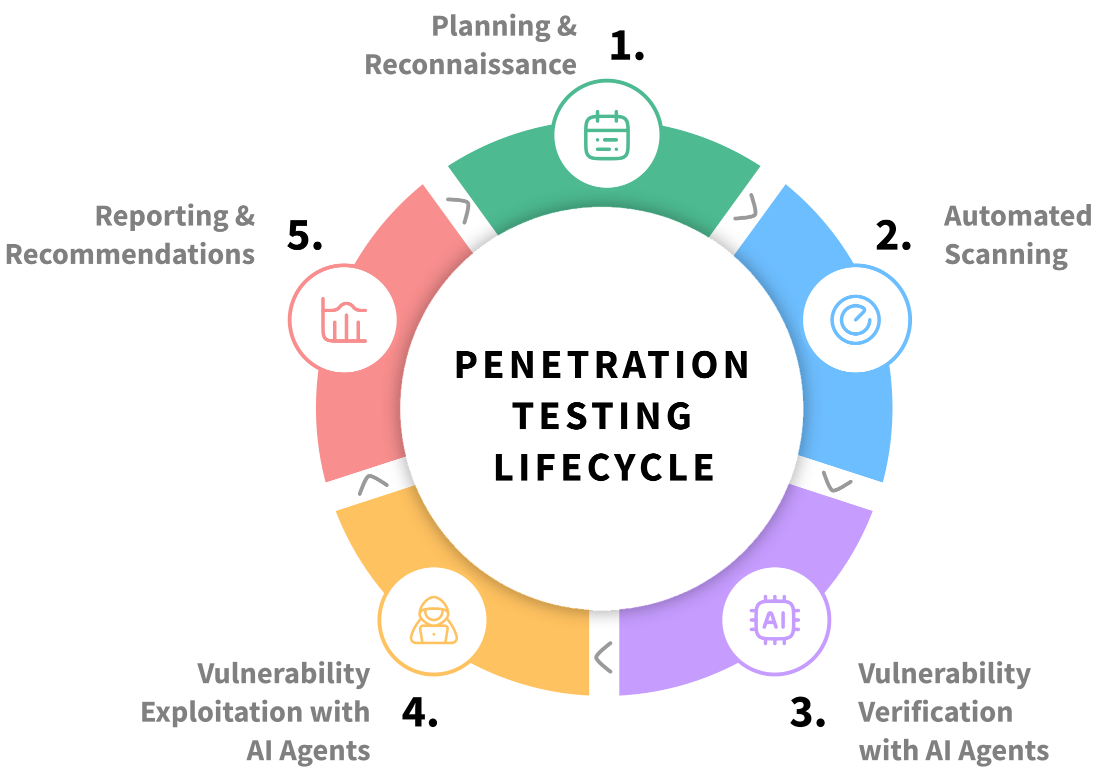
The detailed findings from the penetration testing that are documented within {Appendix A} include corresponding technical recommendations to remediate the identified issues pertaining to the web application and APIs.

Based on the vulnerabilities discovered as part of this security assessment, Infiltra.ai strongly recommends that **Infiltra**

- Prioritise the identified vulnerabilities using a risk-based approach to reduce the security exposure.
- Remediate all the findings in the report and schedule a re-test of the remediated vulnerabilities before deploying the change into production.

2. TESTING APPROACH

Infiltra.ai uses an agentic approach outlined below for the automated security assessment


PENETRATION TESTING LIFECYCLE
1. Planning & Reconnaissance
2. Automated Scanning
3. Vulnerability Verification with AI Agents
4. Vulnerability Exploitation with AI Agents
5. Reporting & Recommendations

3. RISK RATING

The risk rating for each finding in this report is based on the Impact and Exploit vector of the vulnerability. Here is a guide to interpreting the risk rating.

Technical Severity	CVSS v4.0 Score *	Explanation
Critical	9.0 – 10	Vulnerability was discovered that has been rated as critical. It is recommended that corrective actions are implemented urgently. This category of risk should be monitored closely by management.
High	7.0 - 8.9	Vulnerability was discovered that has been rated as important. It is recommended that corrective actions must be implemented within a short term.
Medium	4.0 - 6.9	Vulnerability was discovered that has been rated as of medium criticality. It is recommended that corrective actions should be part of on-going security maintenance of the system.
Low	1.0 - 3.9	Vulnerability was discovered that has been rated as of low criticality. Owners should consider whether to apply corrective measures as part of routine maintenance tasks or to accept the risk.
Info	0 - 0.9	A finding was discovered that has been rated as of informational value which should be addressed to meet industry best practice.

(*) For "vulnerabilities" identified as part of this assessment, Infiltra has adopted the Common Vulnerability Scoring System (version 4.0). CVSS is a vendor independent, industry open standard. It is designed to convey vulnerability severity; help determine urgency and priority of response. The Table 2 above gives a key to the icons and symbols used through this report to provide a clear and concise risk scoring system.

For additional information, please refer:

https://www.first.org/cvss/specification-document

4. SUMMARY OF FINDINGS

Executive summary

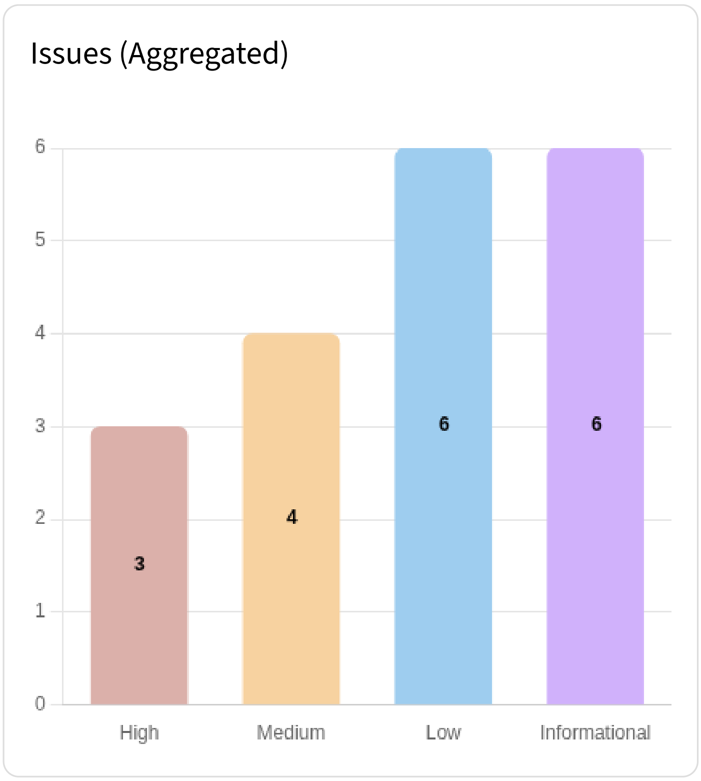
The security assessment of the web application identified a total of 13 vulnerabilities, categorized into 3 High, 4 Medium, and 6 Low risk issues. Critical findings indicate significant weaknesses in the application's authorization mechanisms (Vertical Privilege Escalation) and file upload validation, potentially exposing the Java-based backend to Remote Code Execution (RCE) and unauthorized administrative access. Additionally, the React frontend utilizes vulnerable JavaScript libraries, increasing the attack surface. Several security misconfigurations were also noted in the Nginx web server, specifically regarding missing HTTP security headers (CSP, HSTS, Clickjacking protection). Immediate remediation is recommended for the High-risk vulnerabilities to prevent data breaches and system compromise.

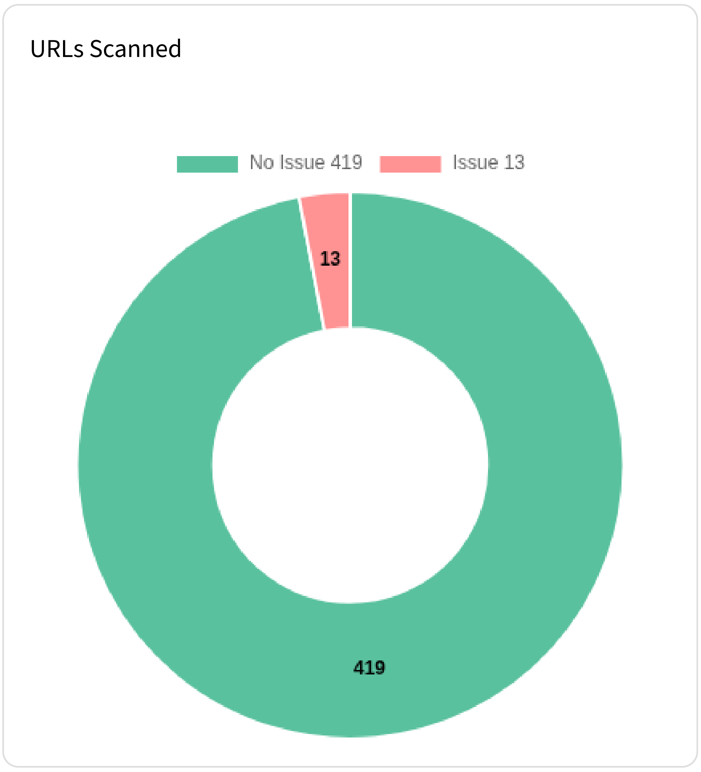
Alerts Summary

- **High:** 3
- **Medium:** 4
- **Low:** 6

Summary of Issues:

The following graph depicts the total number of findings discovered in the application and/or API endpoints during the security assessment





List of Reported Alerts:

#	FINDINGS	SEVERITY

Vulnerability Details and Recommendations

1. Access Control Issue - Improper Authorization

High

Description:A Vertical Privilege Escalation vulnerability exists where a standard user (e.g., 'readOnly' role) can access administrative functions. The Java backend fails to verify user roles effectively before granting access to sensitive endpoints like /admin/dashboard.

Business Impact: High. Unauthorized users can perform administrative actions, potentially leading to data manipulation, theft of sensitive customer information, or full system takeover.

Recommendation:

1. Implement Role-Based Access Control (RBAC) on all backend endpoints using Java security filters (e.g., Spring Security).
2. Verify permissions on the server-side for every request, ensuring the user's session token matches the required role for the requested resource.
3. Deny access by default and explicitly grant permissions to specific roles.

Remediation Effort: 3-5 Days. Requires auditing controller endpoints and implementing a security filter chain.

Relevant Links:

https://spring.io/guides/topicals/spring-security-architecture
https://www.baeldung.com/role-and-privilege-for-spring-security-registration

OWASP Codes: OWASP_2021_A01, CWE-306, OWASP_2017_A05

2. Unrestricted Upload of File with Dangerous Type

High

Description: The application allows file uploads without sufficient validation of the file content. Attackers can bypass extension checks to upload malicious scripts (e.g., web shells), leading to Remote Code Execution.

Business Impact: High. An attacker could execute arbitrary code on the server, leading to a complete compromise of the application and underlying infrastructure.

Recommendation:

1. Validate file types in the Java backend by checking the 'Magic Numbers' (file signature) rather than relying on extensions or Content-Type headers.
2. Rename uploaded files to a random hash and store them outside the web root directory.
3. Configure Nginx to disable script execution in the upload directory.

Remediation Effort: 3-5 Days. Requires rewriting file upload logic and configuring Nginx locations.

Relevant Links:

https://www.baeldung.com/java-file-mime-type
https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html

OWASP Codes: CWE-434, OWASP_2021_A01, OWASP_2017_A05

3. Vulnerable JS Library High

Description: The React frontend is using JavaScript libraries with known security vulnerabilities (CVE-2025-27152, CVE-2025-58754, etc.). These outdated components can be exploited to compromise client-side security.

Business Impact: High. Exploitation of known vulnerabilities in third-party libraries can lead to Cross-Site Scripting (XSS) or other client-side attacks, compromising user sessions.

Recommendation:

1. Run 'npm audit' or 'yarn audit' to identify vulnerable packages.
2. Update the affected libraries to the patched versions specified in the audit report.
3. Implement a dependency management process to regularly check for and update outdated components.

Remediation Effort: 1-2 Days. Depends on breaking changes in updated libraries.

Relevant Links:

https://docs.npmjs.com/auditing-package-dependencies-for-security-vulnerabilities
https://react.dev/learn/add-react-to-an-existing-project#modular-codebase

OWASP Codes: OWASP_2017_A09, CVE-2025-27152, CVE-2025-58754, OWASP_2021_A06, CWE-1395, CVE-2023-45857

4. Access Control Issue - Improper Authorization Medium

Description: An unauthenticated user can access restricted resources. The application fails to enforce authentication checks on specific directories or functions, allowing direct browsing to sensitive URLs.

Business Impact: Medium. Exposure of sensitive information or functionality to unauthenticated users, undermining the application's security model.

Recommendation:

1. Implement a global authentication filter in Java to intercept all requests to protected routes.
2. Ensure that resources requiring authentication return a 401 Unauthorized or 403 Forbidden status if no valid session is present.
3. Audit the Nginx configuration to ensure it is not inadvertently exposing protected internal paths.

Remediation Effort: 2-4 Days. Requires configuring security interceptors and testing route coverage.

Relevant Links:

https://spring.io/guides/gs/securing-web/
https://www.baeldung.com/spring-security-method-security

OWASP Codes: CWE-862, OWASP_2021_A01, OWASP_2017_A05

5. Content Security Policy (CSP) Header Not Set

Medium

Description: The 'Content-Security-Policy' HTTP header is missing. This header is crucial for preventing Cross-Site Scripting (XSS) and data injection attacks by defining which dynamic resources are allowed to load.

Business Impact: Medium. Lack of CSP increases the risk of successful XSS attacks, which can lead to session hijacking and data theft.

Recommendation:

1. Configure Nginx to send the 'Content-Security-Policy' header.
2. Define a strict policy that allows scripts only from trusted domains (e.g., self and Google Fonts API).
3. Use the 'report-uri' or 'report-to' directive initially to monitor policy violations before enforcing them.

Remediation Effort: 1-2 Days. Requires tuning the policy to avoid breaking React functionality.

Relevant Links:

https://content-security-policy.com/examples/nginx/
https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

OWASP Codes: OWASP_2021_A05, SYSTEMIC, CWE-693, OWASP_2017_A06

6. Missing Anti-clickjacking Header

Medium

Description: The application response is missing 'X-Frame-Options' or CSP 'frame-ancestors' directives. This allows the site to be framed by attackers, facilitating Clickjacking attacks.

Business Impact: Medium. Attackers can trick users into performing unintended actions (e.g., clicking hidden buttons) by overlaying the application in a transparent iframe.

Recommendation:

1. Configure Nginx to add the header: 'add_header X-Frame-Options SAMEORIGIN always;'.
2. Alternatively, use the Content-Security-Policy 'frame-ancestors' directive.
3. Ensure this is applied to all server responses serving HTML content.

Remediation Effort: < 1 Day. Simple configuration change in Nginx.

Relevant Links:

https://nginx.org/en/docs/http/ngx_http_headers_module.html
https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html

OWASP Codes: OWASP_2021_A05, CWE-1021, SYSTEMIC, WSTG-v42-CLNT-09, OWASP_2017_A06

7. Sub Resource Integrity Attribute Missing

Medium

Description: External scripts (likely from CDNs or Google Fonts) are loaded without the 'integrity' attribute. If the external source is compromised, malicious code could be injected into the application.

Business Impact: Medium. Compromise of a third-party CDN could lead to the execution of malicious code within the user's browser context.

Recommendation:

1. Generate SRI hashes (sha384/sha512) for all external scripts and stylesheets.
2. Add the 'integrity' and 'crossorigin' attributes to <script> and <link> tags in the React `index.html` or build process.
3. Ensure the build pipeline automatically generates these hashes for production assets.

Remediation Effort: 1-2 Days. May require build configuration updates.

Relevant Links:

https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity
https://webpack.js.org/guides/asset-modules/

OWASP Codes: CWE-345, OWASP_2021_A05, SYSTEMIC, OWASP_2017_A06

8. Application Error Disclosure

Low

Description: The application displays detailed error messages or stack traces to the user. This may reveal internal file paths, library versions, or logic flaws.

Business Impact: Low. Information leakage aids attackers in reconnaissance, helping them craft more targeted attacks.

Recommendation:

1. Configure a global exception handler in Java to catch unhandled exceptions and return generic error messages.
2. Disable default error pages in Nginx and serve custom, static error pages (e.g., 500.html).
3. Ensure debug mode is disabled in the production environment.

Remediation Effort: 1-2 Days. Requires code changes in Java and config changes in Nginx.

Relevant Links:

https://www.baeldung.com/exception-handling-for-rest-with-spring
https://nginx.org/en/docs/http/ngx_http_core_module.html#error_page

OWASP Codes: WSTG-v42-ERRH-02, WSTG-v42-ERRH-01, OWASP_2021_A05, CWE-550, OWASP_2017_A06

9. Private IP Disclosure

Low

Description: The HTTP response body contains private IP addresses or internal hostnames (e.g., 10.x.x.x). This often happens in debug comments or default server error pages.

Business Impact: Low. Disclosed internal network information can assist attackers in mapping the internal infrastructure.

Recommendation:

1. Review application logs and error responses to ensure internal IPs are not echoed back to the client.
2. Configure Nginx to mask or remove headers that might leak upstream IP addresses.
3. Check React build artifacts (source maps) to ensure they do not contain hardcoded internal development URLs.

Remediation Effort: 1 Day. Investigation and configuration adjustment.

Relevant Links:

https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_hide_header

OWASP Codes: OWASP_2021_A01, OWASP_2017_A03, CWE-497

10. Server Leaks Version Information via "Server" HTTP Response Header Field

Low

Description: The web server reveals its specific version in the 'Server' HTTP header (e.g., Nginx/1.18.0). This helps attackers identify specific vulnerabilities associated with that version.

Business Impact: Low. Reduces the effort required for an attacker to find known exploits for the specific server version.

Recommendation:

1. Edit the Nginx configuration file (nginx.conf).
2. Set 'server_tokens off;' within the http, server, or location block.
3. Restart Nginx to apply the changes.

Remediation Effort: < 1 Hour. Simple configuration change.

Relevant Links:

https://nginx.org/en/docs/http/ngx_http_core_module.html#server_tokens

OWASP Codes: OWASP_2021_A05, SYSTEMIC, OWASP_2017_A06, WSTG-v42-INFO-02, CWE-497

11. Strict-Transport-Security Header Not Set

Low

Description: The HSTS header is missing, meaning the browser is not forced to use HTTPS for future requests. Although the tech stack claims HSTS is implemented, the scan indicates it is missing on specific responses.

Business Impact: Low. Users are susceptible to Man-in-the-Middle (MitM) attacks and protocol downgrade attacks if they access the site via HTTP.

Recommendation:

1. Verify Nginx configuration includes: 'add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;'.

2. Ensure the header is applied to the HTTPS server block.
3. Verify that no upstream application logic is stripping this header.

Remediation Effort: < 1 Hour. Configuration verification and fix.

Relevant Links:

https://nginx.org/en/docs/http/ngx_http_headers_module.html
https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

OWASP Codes: OWASP_2021_A05, SYSTEMIC, CWE-319, OWASP_2017_A06

12. Timestamp Disclosure - Unix

Low

Description:A Unix timestamp was found in the response. While often benign, precise timestamps can sometimes be used to predict random number generation seeds or map server-side operations.

Business Impact: Low. Minimal impact, but may aid in cryptographic attacks or information gathering regarding server uptime/behavior.

Recommendation:

1. Review the Java application code to see where timestamps are being serialized to the frontend.
2. Remove timestamps from API responses if they are not required for the business logic.
3. If required, ensure they do not leak sensitive timing information regarding security operations.

Remediation Effort: 1 Day. Code review and minor refactoring.

Relevant Links:

https://www.baeldung.com/jackson-serialize-dates

OWASP Codes: OWASP_2021_A01, SYSTEMIC, OWASP_2017_A03, CWE-497

13. X-Content-Type-Options Header Missing

Low

Description: The 'X-Content-Type-Options' header is not set to 'nosniff'. This allows browsers to MIME-sniff the response, potentially executing non-executable files as scripts.

Business Impact: Low. Increases the risk of Drive-by Download attacks or XSS if the application serves user-uploaded content.

Recommendation:

1. Configure Nginx to add the header: 'add_header X-Content-Type-Options nosniff always;'.
2. Apply this globally in the Nginx http or server block.
3. Ensure the application correctly sets Content-Type headers for all assets.

Remediation Effort: < 1 Hour. Simple configuration change.

Relevant Links:

https://nginx.org/en/docs/http/ngx_http_headers_module.html
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options

OWASP Codes: OWASP_2021_A05, SYSTEMIC, CWE-693, OWASP_2017_A06

SSL SCAN Summary:

The SSLScan analysis of visitormanager.acme.com reveals a secure TLS configuration. Only TLS 1.2 and TLS 1.3 are enabled, with older and vulnerable protocols (SSLv2, SSLv3, TLSv1.0, TLSv1.1) disabled. The server supports strong cipher suites for both TLS 1.2 and TLS 1.3, including AES-GCM and ChaCha20-Poly1305. There is no indication of Heartbleed vulnerability. The server also supports a range of secure Diffie-Hellman groups for TLS 1.3 and elliptic curve groups for both TLS versions. The certificate is valid, not self-signed, and issued by a trusted authority (E8), with a valid expiration date in the future. No critical vulnerabilities were detected in this scan.

Features	Value
SSLv2	False
SSLv3	False
TLSv1.0	False
TLSv1.1	False
TLSv1.2	True
TLSv1.3	True
Fallback	True
Renegotiation	False
Heartbleed	False
Certificate Signature Algorithm	ecdsa-with-SHA384
Certificate Public Key Type	EC
Certificate Public Key Curve	prime256v1
Certificate Subject	visitormanager.acme.com
Certificate Subject Alternative Names	DNS:visitormanager.acme.com
Certificate Issuer	E8
Certificate Self-Signed	False
Certificate Not Valid Before	Jan 1 00:08:03 2026 GMT
Certificate Not Yet Valid	False

Features	Value
Certificate Not Valid After	Apr 1 00:08:02 2026 GMT
Certificate Expired	False

5. APPENDIX A: OWASP TOP 10 VULNERABILITIES

The below table provides mapping of the vulnerabilities found during the penetration testing activity against the most prevalent and critical vulnerabilities identified according to OWASP Top 10.

VULNERABILITIES	REFERENCES	VERIFICATION
Broken Access Control	A1	Vulnerable
Cryptographic Failures	A2	Not Vulnerable
Injection	A3	Not Vulnerable
Insecure Design	A4	Not Vulnerable
Security Misconfiguration	A5	Vulnerable
Vulnerable and Outdated Components	A6	Vulnerable
Identification and Authentication Failure	A7	NA
Software and Data Integrity Failures	A8	Not Vulnerable
Security Logging and Monitoring Failures	A9	NA
Server-Side Request Forgery (SSRF)	A10	NA

6. TESTING LIMITATIONS

The issues raised in this report are only those that were discovered during the scan and are not necessarily a comprehensive statement of all the weaknesses that exist or improvements that might be made.

The scan is performed on a sample basis; and as such cannot, in practice, examine every activity, technical control and procedure, nor can it be a substitute for the customers responsibility to maintain adequate controls over all levels of operations and their responsibility to prevent and detect irregularities, including fraud.

Vulnerability assessments and Penetration testing (VAPT) are point in time assessments and due to the changing landscape of threats and vulnerabilities, newer vulnerabilities are identified on a regular basis. It is incumbent on the customer to follow prudent security best practices commensurate with risks, throughout the application lifecycle. This includes conducting regular security assessments on an ongoing basis.

Recommendations and suggestions for improvement should be assessed for their full commercial impact before they are implemented. The statements made in this report are accurate, but no warranty of completeness, accuracy, or reliability is given in relation to the statements and representations made by, and the information and documentation provided by **Visitormanager.acme.com**