Web Application Agentic Penetration Testing Report - SAMPLE

Agentic AI Penetration Testing — Representative Findings from Actual Customer Engagements Assessment Period: Q4 2025

Customer Name: [REDACTED - Enterprise Web Application]
Report Type: Sample Report — Compilation of Representative Findings
Testing Date Range: October 2025

Report Status: Completed

Table of Contents

Scoping

Penti conducts its tests following the best practices and compliance regulations for each Industry sector and follows the roadmap below.

Assessment Type: Authenticated Web Application & API Penetration Testing — Agentic AI-Assisted
Frameworks Applied:

- OWASP Top 10 2021
- OWASP ASVS (Application Security Verification Standard)
- NIST Cybersecurity Framework
- SOC 2 Trust Services Criteria

Testing Phases:

1. **Reconnaissance & Information Gathering**
2. **Agentic AI Automated Vulnerability Scanning**
3. **Supplementary Scanner Analysis (Burpsuite, ZAP, Nuclei, OpenVAS, Headers)**
4. **Manual Validation & Exploitation**
5. **Documentation & Reporting**
6. **Remediation Support & Re-testing**

Methodology

Our penetration testing methodology follows industry best practices and incorporates techniques from:

- **OWASP Testing Guide** - Web application and API security
- **NIST SP 800-115** - Technical guide to information security testing
- **PTES (Penetration Testing Execution Standard)** - Comprehensive testing methodology

All testing is performed by certified penetration testers augmented by Penti's proprietary Agentic AI engine, which autonomously executes attack chains, validates findings, and triages results — backed by human review at every stage.

Scanner Factors

All findings in our scanners are assigned two measurement factors:

Severity

Severity indicates the impact of the findings on technical and business operations. Findings are classified according to severity as **Critical**, **High**, **Medium**, **Low**, or **Informational**. This reflects the likely impact of each issue for a typical organization.

Confidence

Findings are also classified according to confidence as **Certain**, **Firm**, or **Tentative**. This reflects the inherent reliability of the technique that was used to identify the issue.

Note: A Tentative confidence vulnerability finding has a very low chance of real-world existence and manual assessments will most likely prove the findings to be unharmful based on the application's functionality and business logic.

Risk Rating Methodology

Penti follows the OWASP approach in risk analysis, which uses standard methodologies and is customized for application security. Risk is prioritized based on calculated scores produced by multiplying the estimated scores of **Likelihood** (confidence) and **Impact** (severity) by each other.

Risk Level Classification

- **Critical (9-10):** Immediate action required — severe business impact
- **High (7-8.9):** Should be resolved as soon as possible
- **Medium (4-6.9):** Should be resolved in a reasonable timeframe
- **Low (1-3.9):** Should be resolved when resources permit

Remediation Proposal Methodology

After risks are classified by Risk Rating Methodology, Penti produces a prioritized list of proposed remediations. As a general rule, the most severe risks should be fixed first; however, Penti helps prioritize and balance high-impact security findings with lower-impact ones to ensure your overall security posture improves continuously over time.

Executive Summary

[REDACTED] engaged Penti to perform a comprehensive security assessment of their Web Application and API using Penti's Agentic AI penetration testing platform. Penti's Agentic AI autonomously executed multi-step attack chains across the agreed target scope — autonomously discovering, validating, and triaging vulnerabilities — with certified human penetration testers reviewing, escalating, and contextualizing every finding through a Human-in-the-Loop (HIL) process. Supplementary traditional scanners (Burp Suite Pro, Nuclei, OpenVAS, ZAP, and a dedicated headers scanner) were layered on top to maximize coverage. All testing followed NIST and OWASP guidelines.

The engagement commenced in **October 2025** on the agreed scope. Penti's Agentic AI runs continuously against all [REDACTED] instances and API endpoints — both authenticated and publicly reachable — autonomously re-testing the attack surface on a scheduled basis. Findings are triaged by the AI, validated by human analysts, and surfaced directly to [REDACTED] for immediate remediation action. Traditional vulnerability scanners run in parallel on a quarterly cadence as a complementary coverage layer, with results consolidated into this unified report.

Key Findings Summary

Scanner	Critical	High	Medium	Low	Pass
Agentic AI	2	1	1	0	3
Burpsuite	0	0	1	0	0
Headers	0	0	2	1	6
Nuclei	0	0	0	0	8
Openvas	0	0	0	0	10
Zap	1	0	2	0	0
Total	**3**	**1**	**6**	**1**	**27**

Most Significant Findings

1. **Insecure Direct Object Reference (IDOR) — Unauthorized User Data Access** - Critical (Agentic AI) Broken access controls allow any authenticated user to access other users' private account data by manipulating object identifiers.
2. **Reflected Cross-Site Scripting (XSS)** - Critical (Agentic AI) Unvalidated user input is reflected in HTTP responses, enabling attackers to execute malicious scripts in victims' browsers and steal session credentials.
3. **Vulnerable JS Library** - Critical (Zap) JavaScript library with known, publicly disclosed CVEs in active use, directly exploitable by attackers without intermediate steps.
4. **Vulnerable and Outdated Components** - High (Agentic AI) Multiple application components running outdated versions with documented vulnerabilities available in public exploit databases.
5. **Email Spoofing — Missing Anti-Spoofing Controls** - Medium (Agentic AI) Absent or misconfigured SPF/DKIM/DMARC records allow attackers to send emails impersonating [REDACTED]'s domain, enabling phishing and business email compromise.

Impact Assessment

The combination of findings discovered during this assessment reveals a **HIGH** overall risk posture. An attacker exploiting the confirmed critical vulnerabilities could:

- Access and exfiltrate other users' private account data (IDOR)
- Hijack authenticated sessions and impersonate users (XSS)
- Exploit known CVEs in vulnerable JS libraries for client-side code execution
- Leverage outdated components as pivot points for deeper compromise
- Launch targeted phishing campaigns impersonating [REDACTED]'s domain (email spoofing)

Positive Security Controls Observed

- CSRF protection validated as passing across all tested endpoints
- Firebase database properly secured — no unauthorized data extraction possible
- Misconfigured Firebase scenario tested and confirmed not exploitable
- No SQL injection vectors identified
- Core authentication mechanisms validated as sound

Recommendations Priority

1. **Immediate (24-48 hours):**
    - Implement server-side authorization checks on all object-referencing endpoints (IDOR)
    - Deploy output encoding and Content Security Policy to mitigate XSS
    - Upgrade or replace the identified vulnerable JavaScript library
2. **Short-term (1-2 weeks):**
    - Audit and upgrade all outdated application components
    - Configure SPF, DKIM, and DMARC records to prevent email spoofing
    - Harden Content Security Policy headers across all routes
3. **Medium-term (1-3 months):**
    - Implement a Software Composition Analysis (SCA) pipeline to continuously monitor dependencies
    - Security code review focused on authorization logic
    - Developer training on OWASP Top 10

Scope

Production

N/A

Staging

Web Applications:

- `https://[REDACTED]`

Development

N/A

Our Tools

Penti Agentic AI

- **Penti Agentic AI** - Autonomous AI agent that executes multi-step attack chains, validates exploitability, and triages findings — with certified human penetration testers reviewing every result through a Human-in-the-Loop (HIL) process

Automated Scanners

- **Burp Suite Pro** - Web application proxy and scanner
- **OWASP ZAP** - Open-source web application security scanner
- **Nuclei** - Fast vulnerability scanner with community templates
- **OpenVAS** - Open-source vulnerability assessment system
- **Securily Headers Scanner** - HTTP security header analysis

Manual Tools (not exhaustive)

- Burp Suite Pro
- Ffuf
- Sqlmap
- 403Bypasser
- Ghauri
- Custom Scripts
-

Tool Assessment Results

Agentic AI Pentest Results

Penti's Agentic AI scanner identified **2 Critical, 1 High, 1 Medium and 0 Low** risk vulnerabilities, along with 3 passing findings. The Agentic AI autonomously executed multi-step attack chains, validated exploitability, and performed AI triage on all findings prior to human review.

Total Results

Critical	High	Medium	Low	Pass
2	1	1	0	3

Insecure Direct Object Reference (IDOR) — Unauthorized User Data Access

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** Critical

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025
- Time Since it was Found: 0 days

Description:

The application exposes internal object references — such as user account IDs — in API endpoints and URL parameters without verifying that the requesting user is the authorized owner of the referenced object. An authenticated attacker can enumerate or manipulate these identifiers to access the private account data of any other user.

Problem: Imagine your school gives every student a mailbox, and your mailbox number is 101. To get your mail, you just tell the front desk "I'm here for mailbox 101." But what if the person at the desk doesn't ask for your ID to prove you're the owner of mailbox 101? You could then say "I'm here for mailbox 102," and they'd just hand you your friend's mail. An Insecure Direct Object Reference (IDOR) is like that for a website. It fails to check if you are the real owner of the data you are asking for, allowing you to potentially access someone else's private account information just by changing an ID number in the website's address bar.

Events:

- **Agentic AI confirmed exploitability** — The AI engine autonomously modified object references across authenticated API calls and successfully retrieved data belonging to other user accounts. Exploitation confirmed with Certain confidence.
- **AI Triage: VALID (99% confidence)** — Server returned HTTP 200 with distinct user PII for each modified object reference, confirming absence of server-side authorization enforcement.

Compliance:

Type	Category	Title	ID
SOC 2	Security (Common Criteria)	Logical Access Controls	CC6.1

Justification: An IDOR vulnerability is a direct violation of SOC 2 CC6.1, which requires the entity to implement logical access controls to restrict access to data. An IDOR finding demonstrates a failure of these controls by allowing a user to access data objects they are not authorized to view, directly impacting the security and confidentiality of the system's data.

Remediation:

1. Enforce server-side ownership checks on every endpoint that accepts an object reference
2. Replace sequential numeric IDs with non-guessable UUIDs or opaque tokens in all public-facing APIs
3. Implement a centralized authorization layer (e.g., RBAC/ABAC middleware) applied consistently across all routes
4. Add automated IDOR regression tests to the CI/CD pipeline

Reflected Cross-Site Scripting (XSS)

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** Critical

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025
- Time Since it was Found: 0 days

Description:

User-supplied input submitted to one or more application parameters is reflected directly in the HTTP response without proper encoding or sanitization. An attacker can craft a malicious URL containing JavaScript payloads, which — when visited by a victim — executes in the victim's browser within the full security context of the application.

Problem: Imagine a website is like a talking mirror that shows you whatever you ask it to. For example, if you search for "cats," the mirror shows you the words "You searched for cats." A trickster realizes they can ask the mirror to show a special, invisible command instead of just words. They create a special link and send it to you. When you click it, you ask the mirror to show that tricky command. The command runs in your web browser and can be used to steal your information, like your password or cookies, without you even knowing it. This is called a Reflected Cross-Site Scripting (XSS) attack because the website "reflects" the attacker's malicious command back to you.

Events:

- **Agentic AI autonomous exploitation confirmed** — The AI engine injected XSS payloads across input vectors, identified reflected execution in the HTTP response body, and validated the attack chain end-to-end including session cookie exfiltration simulation.
- **AI Triage: VALID (98% confidence)** — Payload `<script>alert(document.domain)</script>` reflected unencoded in server response with Content-Type `text/html`, confirming exploitability.

Compliance:

Type	Category	Title	ID
SOC 2	Security (Common Criteria)	Vulnerability Detection and Management	CC7.1

Justification: A Reflected XSS vulnerability is a direct threat to system security. SOC 2 Common Criterion CC7.1 requires an entity to use detection procedures to identify system vulnerabilities. This finding indicates a gap in the controls designed to prevent or detect such security flaws, impacting the entity's security posture under the SOC 2 framework.

Remediation:

1. Apply context-aware output encoding to all user-supplied data reflected in HTML responses (use framework-native escaping — never roll your own)
2. Implement a strict Content Security Policy: `default-src 'self'; script-src 'self'; object-src 'none'`
3. Set `HttpOnly` and `Secure` flags on all session cookies to limit the impact of XSS exploitation
4. Conduct a full audit of all user-input reflection points across the application

---

Vulnerable and Outdated Components Detection

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** High

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025
- Time Since it was Found: 0 days

Description:

The application relies on one or more third-party libraries or framework components that are running versions with known, documented vulnerabilities. These vulnerabilities are publicly catalogued with

available proof-of-concept exploit code, significantly lowering the bar for exploitation.

Problem: Imagine your application is built with digital LEGO bricks. Some of these bricks are old and have known weaknesses that hackers have discovered. The security scanner found that you are using some of these weak, outdated bricks. An attacker could use these known flaws to break into your application, steal data, or cause damage. Fixing this means swapping out the old, vulnerable bricks for new, stronger ones that have the weaknesses fixed.

Events:

- **AI Triage: NEEDS_INFO (95% confidence)** — The Agentic AI flagged the finding and confirmed component version fingerprinting. Specific CVE identifiers require additional evidence from the vendor advisory to finalize remediation scope. Manual follow-up recommended.

Compliance:

Type	Category	Title	ID
SOC 2	Security (Common Criteria)	Vulnerability Management	CC7.1

Justification: The 'Vulnerable and Outdated Components' test directly addresses SOC 2 CC7.1, which requires the entity to use detection procedures to identify susceptibilities to vulnerabilities. This test is a core component of a vulnerability management program necessary for compliance.

Remediation:

1. Enumerate all third-party dependencies using an SCA tool (e.g., Snyk, Dependabot, OWASP Dependency-Check)
2. Prioritize upgrades for components with known CVEs and available patches
3. Establish a dependency update policy with automated alerts for newly disclosed CVEs
4. Remove unused or deprecated libraries from the project entirely

---

Email Spoofing — Missing Anti-Spoofing Controls

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** Medium

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025
- Time Since it was Found: 0 days

Description:

The organization's domain lacks properly configured anti-spoofing DNS records (SPF, DKIM, DMARC). This allows an attacker to send emails that appear to originate from [REDACTED]'s legitimate domain, enabling business email compromise (BEC), phishing attacks targeting employees or customers, and fraudulent communications that appear entirely authentic to recipients.

Problem: Imagine someone can send letters to your friends and sign your name at the bottom, making it look like the letter came from you. They could write mean things or ask for money, and your friends would think it was you. Email spoofing is the digital version of this. Attackers can send emails that look like they came from your company's official address, even though they didn't. They use this to trick your employees or customers into clicking dangerous links or revealing secret information because the email seems trustworthy.

Events:

- **AI Triage: VALID (90% confidence)** — DNS record analysis confirmed the domain either lacks a DMARC policy or has it configured in monitoring-only mode (`p=none`), which provides no active protection against spoofing.

Compliance:

Type	Category	Title	ID
SOC 2	Security (Common Criteria)	Fraud Risk Assessment from Email Spoofing	CC5.3

Justification: The Email Spoofing test directly assesses a primary vector for fraud such as business email compromise (BEC) and phishing attacks. This aligns with SOC 2 CC5.3, which requires an entity to consider the potential for fraud when assessing risks to its objectives. A successful exploit

plan indicates a weakness in controls designed to prevent fraudulent communication and protect sensitive information.

Remediation:

1. Publish a valid SPF record: `v=spf1 include:[REDACTED-mailprovider] -all`
2. Configure DKIM signing for all outbound email streams
3. Deploy DMARC with at minimum `p=quarantine` — progress to `p=reject` after validating legitimate mail flows
4. Use a DMARC monitoring service (e.g., Dmarcian, Valimail) to identify unauthorized senders before enforcement

---

Passing Findings

The following Agentic AI tests completed successfully with no exploitable vulnerabilities identified:

Finding	Status	Risk
Cross-Site Request Forgery (CSRF) Vulnerability Exploit	PASS	Critical
Using Components with Known Vulnerabilities — Automated Discovery	PASS	Critical
Misconfigured Firebase Database User Data Extraction	PASS	Critical

---

Burpsuite Scan Results

Burpsuite scan identified **0 Critical, 0 High, 1 Medium and 0 Low** risk vulnerabilities, along with 0 passing findings.

Total Results

Critical	High	Medium	Low	Pass
0	0	1	0	0

Content Security Policy: Allows Untrusted Style Execution

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** Medium

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025
- Time Since it was Found: 0 days

Description:

The Content Security Policy fails to prevent untrusted style execution. The policy contains `unsafe-inline` in the `style-src` directive and allows global wildcard URLs, permitting arbitrary styles to be loaded and executed from any origin.

Problem: Imagine your website as a house, and its style (fonts, colors, layout) as the furniture. A Content Security Policy (CSP) is like a lock on the door, controlling what furniture can be brought inside. This security test found that your "lock" is faulty, allowing anyone to bring in their own furniture, even if it's dangerous. This "dangerous furniture" could be malicious code disguised as style instructions, which could trick your website into doing bad things like redirecting visitors to a fake login page or stealing their information.

Events:

- **AI Triage: VALID (95% confidence)** — The policy's `style-src` directive contains `unsafe-inline` and a global wildcard. These are insecure configurations that explicitly allow loading styles from untrusted sources, undermining protection against XSS and data exfiltration.
- **Remediation classification:** Tier 2 — A missing or poorly configured CSP violates best practices and compliance standards like OWASP and PCI DSS. While not directly exploitable in isolation, it weakens a critical security control and lowers the barrier for XSS exploitation.

Compliance:

Type	Category	Title	ID
SOC 2	Security	Content Security Policy (CSP)	CC6.1

Remediation:

1. Remove `unsafe-inline` from `style-src` — replace with nonce-based or hash-based policy
2. Remove global wildcard (`*`) allowances and replace with explicit trusted origins
3. Test CSP changes against production content using the CSP Evaluator tool before deployment

Headers Scan Results

Headers scan identified **0 Critical, 0 High, 2 Medium and 1 Low** risk vulnerabilities, along with 6 passing findings.

Total Results

Critical	High	Medium	Low	Pass
0	0	2	1	6

Feature-Policy

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** Low

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025
- Time Since it was Found: 0 days

Description:

The `Feature-Policy` (now superseded by `Permissions-Policy`) HTTP response header is absent. Without this header, the application cannot restrict which browser APIs and device features are accessible to page content and embedded third parties.

Events:

- **AI Triage: VALID (85% confidence)** — The header's absence is a confirmed unsafe configuration representing a missing defense-in-depth control.
- **Remediation classification:** Tier 2 — Absence requires an additional vulnerability (e.g., XSS) to be leveraged. Classified as regulatory compliance due to OWASP ASVS requirements.

Compliance:

Type	Category	Title	ID
SOC 2	Security (Common Criteria)	Vulnerability Identification and Remediation	CC7.1

Remediation:

Add the following header to all HTTP responses:

```
None
Permissions-Policy: geolocation=(), camera=(), microphone=(), payment=()
```

Cross-Origin-Opener-Policy

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** Medium

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025
- Time Since it was Found: 0 days

Description:

The `Cross-Origin-Opener-Policy` (COOP) HTTP security header is missing. Without this header, a malicious page opened from the application can retain a window reference back to the application, enabling cross-origin attacks such as XS-Leaks.

Problem: Imagine your website is like a private room. If someone from your website opens a link to

another website in a new window, that new window might be able to peek back into your private room. The COOP header is like a special rule for your room that says, "Any new windows opened from here cannot look back at me."

Events:

- **AI Triage: VALID (95% confidence)** — Confirmed header is absent from all tested responses. The application's window can be referenced by cross-origin pages, exposing it to XS-Leak attack patterns.

Compliance:

Type	Category	Title	ID
SOC 2	Security (Common Criteria)	Uses Detection and Monitoring Procedures	CC7.1

Remediation:

Add the following header to all HTTP responses:

```
None
Cross-Origin-Opener-Policy: same-origin
```

Permissions-Policy

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** Medium

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025

- Time Since it was Found: 0 days

Description:

The `Permissions-Policy` header is not set, leaving browser feature access unrestricted for page content and third-party scripts.

Events:

- **AI Triage: VALID (85% confidence)** — Confirmed header is missing across all scanned responses.

Compliance:

Type	Category	Title	ID
SOC 2	Security	Logical Access Control	CC6.1

Remediation:

Define and deploy a `Permissions-Policy` header scoped to only the browser features the application legitimately requires. Deny all others explicitly.

Nuclei Scan Results

Nuclei scan identified **0 Critical, 0 High, 0 Medium and 0 Low** risk vulnerabilities, along with 8 passing findings.

Total Results

Critical	High	Medium	Low	Pass
0	0	0	0	8

Openvas Scan Results

Openvas scan identified **0 Critical, 0 High, 0 Medium and 0 Low** risk vulnerabilities, along with 10 passing findings.

Total Results

Critical	High	Medium	Low	Pass
0	0	0	0	10

Zap Scan Results

Zap scan identified **1 Critical, 0 High, 2 Medium and 0 Low** risk vulnerabilities, along with 0 passing findings.

Total Results

Critical	High	Medium	Low	Pass
1	0	2	0	0

CSP: Wildcard Directive

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** Medium

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025
- Time Since it was Found: 0 days

Description:

The Content Security Policy contains wildcard (`*`) directives that allow resources to be loaded from any origin, significantly undermining the protective value of the policy.

Events:

- **AI Triage: NEEDS_INFO (85% confidence)** — The specific `Content-Security-Policy` header value was not captured in scanner evidence, preventing confirmation of which directive contains the wildcard. Manual verification recommended.
- **Remediation classification:** Tier 2 — Wildcard CSP is a misconfiguration that increases risk without being directly exploitable in isolation.

Compliance:

Type	Category	Title	ID
SOC 2	Security	Logical Access	CC6.1

Remediation:

Audit the full CSP and replace all `*` wildcards with explicit allowlists of trusted origins.

Cross-Domain JavaScript Source File Inclusion

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** Medium

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025
- Time Since it was Found: 0 days

Description:

The application loads JavaScript files from external domains. If any of those domains are compromised, the loaded scripts execute in the context of the application with full access to the DOM, cookies, and session data.

Events:

- **AI Triage: NEEDS_INFO (95% confidence)** — Scanner did not capture which external domain or script file triggered the finding. Without identifying the specific third-party domain, the trustworthiness of the source cannot be assessed. Manual review of all `<script src="...">` tags referencing external origins is required.

Compliance:

Type	Category	Title	ID
SOC 2	Security	Logical Access Control	CC6.1

Remediation:

1. Audit all externally loaded JavaScript files
2. Apply Subresource Integrity (SRI) hashes to every third-party script tag:

```
None
<script src="https://cdn.example.com/lib.js"

        integrity="sha384-[HASH]"

        crossorigin="anonymous"></script>
```

3. Restrict allowed external script origins in the Content Security Policy `script-src` directive

---

Vulnerable JS Library

Category: Web Applications

Target: `https://[REDACTED]` **Host:** `https://[REDACTED]`

Remediation: To do **Status:** FAIL **Risk:** Critical

Time Frame:

- Created Date: 10/30/2025
- Last Updated Date: 10/30/2025
- Time Since it was Found: 0 days

Description:

ZAP identified client-side use of a JavaScript library with a known, publicly disclosed vulnerability. The library version matches signatures in the scanner's vulnerability database, and public proof-of-concept exploits are available for the associated CVE(s).

Problem: Imagine your application is built with digital LEGO bricks. Some of these bricks are old and have known weaknesses that hackers have discovered. An attacker can use these known flaws to break into your application, steal data, or cause damage — without needing to find new weaknesses themselves.

Events:

- **AI Triage: NEEDS_INFO (95% confidence)** — The finding does not specify which library is vulnerable, the version fingerprinted, or the associated CVE(s). Without this evidence, the

specific remediation path cannot be confirmed. Manual inspection of loaded client-side libraries is required.

- **Remediation classification:** Tier 1 — Use of components with known, publicly available exploits creates a direct exploitation risk. Successful exploitation can result in session hijacking, sensitive data theft, or arbitrary client-side code execution without requiring intermediate steps.

Compliance:

Type	Category	Title	ID
SOC 2	Common Criteria	Monitors for Vulnerabilities	CC7.1

Remediation:

1. Use browser developer tools or a tool like Retire.js to identify all client-side library versions
2. Cross-reference identified versions against the NVD or Snyk vulnerability database
3. Upgrade to the patched version of the identified library
4. Integrate automated SCA scanning into the CI/CD pipeline to prevent future regressions

Manual Assessment Results

We have researched and confirmed the highest priority findings from both the manual and automated assessments:

#	Title of Finding	Status	Risk
1	Insecure Direct Object Reference (IDOR) — Unauthorized User Data Access	Active	**Critical**
2	Reflected Cross-Site Scripting (XSS)	Active	**Critical**
3	Vulnerable JS Library	Active	**Critical**
4	Vulnerable and Outdated Components	Active	**High**
5	Email Spoofing — Missing Anti-Spoofing Controls	Active	**Medium**

Prioritized Remediation

Tier 1: Critical — Immediate Action Required (24-48 hours)

Finding	Scanner	Risk	Business Impact
IDOR — Unauthorized User Data Access	Agentic AI	Critical	Unauthorized access to all user accounts
Reflected XSS	Agentic AI	Critical	Session hijacking, account takeover
Vulnerable JS Library	Zap	Critical	Direct client-side exploitation via known CVEs

Recommended Actions:

1. Deploy server-side authorization checks on all object-referencing API endpoints immediately
2. Apply output encoding to all reflected user input — audit the entire application surface
3. Identify and upgrade the flagged JavaScript library — use Retire.js or equivalent

Tier 2: High Priority — Short Term (1-2 weeks)

Finding	Scanner	Risk	Business Impact
Vulnerable and Outdated Components	Agentic AI	High	Known exploit vectors against framework dependencies
Content Security Policy: Untrusted Style Execution	Burpsuite	Medium	Weakened XSS defenses
Cross-Origin-Opener-Policy Missing	Headers	Medium	Exposure to XS-Leak cross-origin attacks
Permissions-Policy Missing	Headers	Medium	Unrestricted browser feature access
CSP: Wildcard Directive	Zap	Medium	Weakened content injection protection
Cross-Domain JS Source Inclusion	Zap	Medium	Supply chain script execution risk

Recommended Actions:

1. Run a full SCA audit — upgrade all outdated components with known CVEs
2. Harden the CSP: remove `unsafe-inline`, remove wildcards, add SRI to external scripts
3. Deploy missing security headers: `Cross-Origin-Opener-Policy`, `Permissions-Policy`

Tier 3: Lower Priority — Medium Term (1-3 months)

Finding	Scanner	Risk	Business Impact
Email Spoofing — Missing Anti-Spoofing Controls	Agentic AI	Medium	BEC and phishing campaigns impersonating the domain
Feature-Policy Missing	Headers	Low	Missing browser feature restriction

Recommended Actions:

1. Configure and validate SPF, DKIM, and DMARC records — progress to `p=reject`
2. Deploy `Permissions-Policy` header (this also satisfies the Feature-Policy finding)

Please don't hesitate to raise any questions on this topic — we're here to assist in any way we can.

Re-testing

The goal of our penetration test is to ensure that the findings we are addressing have been assessed, remediated, and confirmed.

After finishing the remediation stage, we will retest the findings to ensure that their mitigation is complete.

Retest Timeline

- **Tier 1 Findings:** Retest within 1 week of remediation deployment
- **Tier 2 Findings:** Retest within 2 weeks of remediation deployment
- **Tier 3 Findings:** Retest within 1 month of remediation deployment

Pending Retests

The following findings are pending remediation and retest:

- Insecure Direct Object Reference (IDOR)
- Reflected Cross-Site Scripting (XSS)
- Vulnerable JS Library
- Vulnerable and Outdated Components
- Email Spoofing — Missing Anti-Spoofing Controls
- Content Security Policy: Allows Untrusted Style Execution
- Cross-Origin-Opener-Policy
- Permissions-Policy
- CSP: Wildcard Directive
- Cross-Domain JavaScript Source File Inclusion
- Feature-Policy

Retest Credits: Included in the original engagement scope at no additional cost.

Disclosure

Penti uses the best practices for security scanning and detecting the security holes in an application and/or infrastructure. Please note that new findings and security holes are constantly discovered and patches to software and platforms are conducted momentarily. Under development applications are especially more prone to frequent findings introduced. As a result, Penti is unable to guarantee that your application or infrastructure is completely safe from every form of attacks that are undiscovered at the time.

Privacy Protection: All URLs, customer-identifying information, and application-specific details have been redacted to protect client confidentiality. Real customer domains, endpoints, and infrastructure details are not disclosed in this sample report.

Scope Limitations: Testing was limited to the agreed-upon scope. Systems, applications, or network segments outside the scope were not tested and may contain vulnerabilities.

Evolving Threat Landscape: Cyber threats evolve rapidly. New attack techniques, vulnerabilities, and exploits are discovered daily. Regular periodic testing is strongly recommended as part of a comprehensive security program.

Our Certifications

Penti's penetration testing team holds the following industry-recognized certifications:

Offensive Security Certifications

- **OSCP+** (OffSec Certified Professional Plus)
- **OSCP** (OffSec Certified Professional)
- **CPTS** (Hack The Box Certified Penetration Testing Specialist)
- **eCPPTv2** (eLearnSecurity Certified Professional Penetration Tester v2)
- **eJPTv2** (eLearnSecurity Junior Penetration Tester v2)
- **CEH** (Certified Ethical Hacker - Practical)

Cloud Security Certifications

- **AWS Certified Security - Specialty**
- **AWS Solutions Architect Associate**
- **Microsoft Azure Security Engineer Associate**
- **Microsoft Azure Administrator Associate**
- **Google Cloud Professional Security Engineer**

Blue Team & Defense Certifications

- **BTL1** (Blue Team Level 1)
- **CCSE** (Certified Container Security Expert)
- **CompTIA PenTest+**
- **CompTIA Security+**

Specialized Certifications

- **CREST** - Elected Member, Pentest Focus Group Subcommittee
- **Metasploit Pro Certified Specialist**
- **PentesterLab** - PCAP, Unix, Introduction Badges
- **Hack The Box** - Dante Pro Lab, 130+ machines completed

Contact Information

Penti - Agentic AI Penetration Testing Boca Raton, Florida www.penti.ai

Report Prepared By: Penti Penetration Testing Team Lead Penetration Testers: [REDACTED]

Report Date: October 2025 **Report Version:** 1.0 — Sample **Report Type:** Agentic AI Web Application Penetration Testing Report **Classification:** Sample — For Demonstration Purposes Only

This report contains sensitive security information. Distribution should be limited to authorized personnel only. Unauthorized disclosure may increase security risks.