aikido

06 May 2026

# Penetration Test Report

## Security Assessment of AI Pentest Demo Application

For ACME Corp

# Table of contents

# 1. Executive Summary

## 1.1 Confidentiality statement

This document is the exclusive property of **ACME Corp** and **Aikido**. This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires the consent of ACME Corp. ACME Corp may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

## 1.2 Report Overview

On **06 May 2026**, Aikido conducted a **white box** web application penetration test for ACME Corp focused on the application(s) within the defined scope and their exposure to targeted attacks. The primary goal of this engagement was to proactively discover vulnerabilities within those scoped applications that could lead to compromise or exposure of sensitive information.

The penetration test simulated real-world attack techniques across applicable attack surfaces, including authentication, authorization, input handling, business logic, and where relevant, agentic and API-specific behaviors, to identify risks within the scoped application(s). The configured scope was limited to the following domains and associated endpoints:

- **https://7931b860-nginx-3000.aipentest.attack-me.com**

The evaluation resulted in 1 critical, 5 high, 8 medium and 5 low findings and revealed that there are several opportunities to strengthen the cybersecurity posture of the application. The environment shows a foundational commitment to security, but targeted improvements are recommended to reduce the likelihood and impact of compromise.

### Identified Improvement Points

The application presents several areas where input handling and access controls could be strengthened. User-supplied data in multiple parts of the application is not sufficiently validated or sanitized before being processed, which creates opportunities for unintended behavior. Reviewing how the application handles inputs across its endpoints and applying consistent validation practices would meaningfully reduce this exposure.

Access to sensitive data and internal functionality appears to lack adequate authentication and authorization checks in some areas. Certain endpoints may return information or perform actions without properly verifying the identity or permissions of the requester. Introducing consistent access control checks and limiting the information exposed by diagnostic or utility endpoints would help protect user data and system internals.

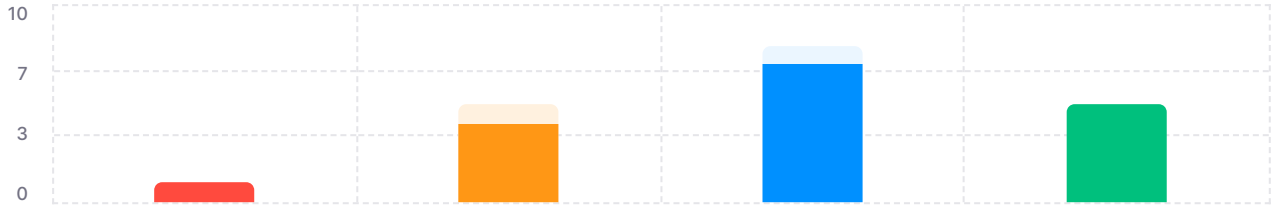
The application's supporting infrastructure, including its API layer, session management, and network configuration, has several areas where hardening measures could be applied. Strengthening password policies, tightening API query controls, and ensuring secure transport and cookie settings are in place would improve the overall security posture. These are generally low-effort improvements that can provide a meaningful layer of defense.

### Conclusion

This engagement identified 19 findings that could impact the confidentiality, integrity, and availability of the application if left unaddressed. So far, 2 of 19 have been resolved and 17 remain open, reflecting active remediation and a clear commitment to the application's security posture. We recommend completing the remaining items and validating each fix through structured retesting.

# 2. Findings

## 2.1 Vulnerability Distribution




| Critical | High | Medium | Low |
|---|---|---|---|
| 0/1 solved | 1/5 solved | 1/8 solved | 0/5 solved |

The table below outlines each identified vulnerability, categorized by severity and current remediation status. Findings highlight several critical and high-risk issues that could allow attackers to compromise application integrity, access sensitive data, or execute arbitrary code on the server.

- **Critical-severity issues** involve a command injection vulnerability in the /backup-notes endpoint that enables full remote code execution, posing an immediate and severe risk to system integrity and confidentiality.
- **High-severity issues** include SQL injection, stored and reflected XSS, path traversal/LFI, unauthenticated IDOR exposing private notes, and environment variable disclosure, all of which could lead to data theft, unauthorized access, or full account compromise.
- **Medium-severity issues** encompass reflected XSS, cache poisoning, GraphQL misconfigurations (GET queries, alias overloading, lack of depth limiting, introspection enabled, field suggestions), and weak password policies, which could enable information leakage, denial of service, or user-targeted attacks.
- **Low-severity issues** include a missing Secure flag on session cookies, full-read SSRF enabling internal network probing, DOM XSS in link preview rendering, weak TLS cipher suites, and missing or misconfigured security headers, which collectively increase the attack surface and risk of data interception or exposure.

## 2.2 Master Findings Table

| ID | Title | State | Severity |
|---|---|---|---|
| PT-1 | Command Injection in /backup-notes leads to Remote Code Execution | Unresolved | Critical |
| PT-2 | Time-based SQL Injection in /public-notes search parameter | Unresolved | High |
| PT-3 | Stored XSS in blog article rendering via unsanitized GraphQL article content | Risk Accepted | High |
| PT-4 | Path Traversal LFI in /static via non-overlapping '../' filter bypass | Unresolved | High |
| PT-5 | Unauthenticated IDOR on /api/notes/:id exposes private notes | Unresolved | High |
| PT-6 | Unauthenticated environment variable disclosure in /health debug endpoint | Unresolved | High |
| PT-7 | Reflected XSS in /blog/:id via JavaScript context injection | Fixed | Medium |
| PT-8 | Header-based cache poisoning on /static via X-Content-Type | Unresolved | Medium |
| PT-9 | GraphQL endpoint accepts GET requests for query execution | Unresolved | Medium |
| PT-10 | Registration accepts trivial weak passwords without policy enforcement | Unresolved | Medium |
| PT-11 | GraphQL endpoint allows alias overloading without limits | Unresolved | Medium |
| PT-12 | GraphQL endpoint lacks query depth limiting | Unresolved | Medium |
| PT-13 | GraphQL introspection enabled on /graphql endpoint | Unresolved | Medium |
| PT-14 | GraphQL field suggestions leak valid schema fields | Unresolved | Medium |
| PT-15 | Authentication session cookie missing Secure flag | Unresolved | Low |

| ID | Title | State | Severity |
|---|---|---|---|
| PT-16 | Full-read SSRF in /api/link-preview enables arbitrary outbound requests and internal address probing | Unresolved | Low |
| PT-17 | DOM XSS in create-note link preview rendering via unescaped fetched response body | Unresolved | Low |
| PT-18 | Weak TLS cipher suites accepted | Unresolved | Low |
| PT-19 | Missing or Misconfigured Security Headers | Unresolved | Low |

## 2.3 Detailed Findings

### 2.3.1 PT-1 - Command Injection in /backup-notes leads to Remote Code Execution

Critical

Identified on: May 6th 2026

#### Description

A command injection vulnerability was validated in the backup feature where the user-controlled `format` parameter is concatenated into a shell command. Arbitrary shell commands are executed on the server by supplying crafted input such as `cat; id 1>&2; false`, and command output is reflected in the HTTP error response. This allows authenticated users to achieve remote code execution with the application process privileges.

#### Business impact

Remote code execution enables complete compromise of the application host under the privileges of the application process. An attacker can run arbitrary OS commands, read or modify local files, pivot to other internal services, and exfiltrate sensitive information such as application secrets, database credentials, and user data. Service availability is at risk, as attackers can terminate processes, consume system resources, or alter backups. Data integrity risks include tampering with or deleting backups and user content. Given that exploitation only requires authentication and simple payloads, the likelihood of discovery and abuse is high. Depending on the data stored (e.g., personal notes or user information), unauthorized access and disclosure may trigger regulatory obligations (GDPR/CCPA) and incident reporting. The reflected error output increases the impact by directly leaking system information to the client.

#### References

**CVSS 9.4:** CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:H/VA:H/SC:H/SI:H/SA:H

### 2.3.2 PT-2 - Time-based SQL Injection in /public-notes search parameter

High

Identified on: May 6th 2026

#### Description

A SQL injection vulnerability was confirmed in the `/public-notes` endpoint where `search` is concatenated directly into a SQL string after only removing double quotes. Timing-based payloads were successful: `search=abc' OR SLEEP(0) OR '1'='1` returned quickly, while the identical payload with `SLEEP(5)` caused a significant delay, confirming SQL execution of attacker-controlled input. This allows unauthenticated manipulation of database queries and can enable unauthorized data access.

#### Business impact

Exposure of a time-based SQL injection in a publicly accessible search endpoint presents a high risk of unauthorized data access and query manipulation. An attacker can leverage boolean or timing-based techniques to infer the structure and content of the database, pivot to extract sensitive fields, or enumerate records that should not be publicly retrievable. If private notes or user-related data are stored in the same database, this can lead to confidentiality breaches and erosion of data integrity. Timing functions and heavy database workloads can be abused to degrade service availability. Repeated requests that force long-running operations can cause performance issues, increased resource consumption, and potential denial of service for legitimate users. If personal or regulated information is stored in notes or associated tables, exploitation could trigger regulatory obligations (e.g., GDPR/CCPA) due to unauthorized access. Given the endpoint is unauthenticated and the exploit is straightforward, the likelihood of discovery and abuse is high.

#### References

**CVSS 9.3:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:L/SC:H/SI:H/SA:H

### 2.3.3 PT-3 - Stored XSS in blog article rendering via unsanitized GraphQL article content

~~High~~ Risk Accepted

Identified on: May 6th 2026 | Resolved on: August 5th 2026

#### Description

A stored cross-site scripting issue was identified in the blog article page where article.content from GraphQL is inserted into innerHTML without sanitization. An unauthenticated GraphQL createArticle mutation was used to store payloads such as <img src=x onerror="console.log(424297718)">, and visiting /blog/9 executed script in the browser with a valid XSS canary. This enables persistent script execution in the application origin for users who open attacker-created article links.

#### Business impact

A successful stored XSS enables persistent execution of attacker-controlled JavaScript in the context of the application's origin. This can lead to session hijacking if cookies are accessible to JavaScript, credential or personal data exfiltration from pages within the origin, UI redress/defacement, and the ability to perform privileged actions on behalf of users by abusing their authenticated sessions. Because the payload is stored and delivered to every visitor of the affected article, the blast radius includes all users who open attacker-provided links, including administrators or moderators. If administrators view the malicious article, administrative sessions and workflows may be compromised, potentially resulting in publication of further malicious content, configuration changes, or user data exposure. The likelihood of exploitation is high due to the unauthenticated createArticle mutation and the absence of output sanitization in the client. If personal data is exposed or modified as a result of this issue, regulatory obligations (e.g., GDPR/CCPA) may be triggered, and reputational damage is likely due to site defacement or user compromise.

#### References

**CVSS 6.4:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:P/VC:N/VI:N/VA:N/SC:H/SI:H/SA:N

### 2.3.4 PT-4 - Path Traversal LFI in /static via non-overlapping '../' filter bypass

High

Identified on: May 6th 2026

#### Description

A local file inclusion vulnerability was validated in the `/static` endpoint because path traversal sanitization only removes literal `../` substrings once and can be bypassed with crafted sequences such as `..../`. This input is transformed into traversal segments during normalization, allowing file reads outside the intended `public` directory. Unauthorized access to sensitive server files was demonstrated by retrieving `/etc/passwd` over HTTP.

#### Business impact

Unauthorized file reads can disclose sensitive server data and application secrets. Typical impacts include exposure of system configuration files, application source code, environment variables, and credentials stored in configuration files. This can facilitate further compromise, including account takeover (by reading secrets) or remote code execution in chained scenarios. Because the endpoint is publicly accessible and exploitation is straightforward, the likelihood of discovery and abuse is high. Disclosure of system and application information may trigger regulatory or contractual obligations depending on the data exposed and could materially impact confidentiality and trust.

#### References

**CVSS 9.2:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:H/SI:N/SA:N

### 2.3.5 PT-5 - Unauthenticated IDOR on /api/notes/:id exposes private notes

High

Identified on: May 6th 2026

#### Description

Private note records are returned by `/api/notes/:id` without authentication, ownership, or visibility checks. It was demonstrated that a note created as private ( `is_public = 0` ) can still be fetched from an unauthenticated session by requesting its numeric ID. This allows unauthorized disclosure of cross-user note content and author metadata.

#### Business impact

Private note content and associated author metadata can be read by any unauthenticated party, resulting in a confidentiality breach. Notes may contain sensitive information such as personal data, credentials, internal references, or proprietary business content. Exposure of author names can also reveal user identities and relationships. The likelihood of exploitation is high due to simple numeric ID enumeration and a JSON endpoint that facilitates automated scraping. Attackers can iterate through IDs to harvest the entire note repository, including private entries, without detection if monitoring is insufficient. Unauthorized disclosure of user-generated content and personal data may trigger regulatory obligations (e.g., GDPR/CCPA) and erode user trust. Depending on the nature of the notes, breach notification requirements, contractual violations, or reputational damage may apply.

#### References

**CVSS 8.7:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N

### 2.3.6 PT-6 - Unauthenticated environment variable disclosure in /health debug endpoint

High

Identified on: May 6th 2026

#### Description

Environment variables were exposed through the /health endpoint without authentication when query parameters were supplied. By using `env`, arbitrary process environment values (including `DB_PASSWORD`) were returned, and `debug=true` caused the full `process.env` object to be disclosed. This enables unauthorized retrieval of sensitive configuration and credential material that can facilitate further compromise.

#### Business impact

Exposing environment variables enables immediate disclosure of sensitive secrets such as database passwords, service credentials, and internal configuration. With knowledge of the database password, an attacker may gain unauthorized access to the backing data store, leading to confidential data exposure, data tampering, or service disruption. Secrets reused across services could enable lateral movement and broader compromise of associated infrastructure. Beyond direct credential leakage, the response reveals operational details (runtime version, process memory usage, PATH, hostnames) that aid targeted exploitation, evasion, and fingerprinting. The likelihood of exploitation is high, as the issue is unauthenticated, easily discoverable, and requires minimal skill. If personal or regulated data is accessible via the exposed credentials, this exposure can trigger compliance obligations under regulations such as GDPR or CCPA due to potential data breach implications.

#### References

**CVSS 9.2:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:H/SI:N/SA:N

### 2.3.7 PT-7 - Reflected XSS in /blog/:id via JavaScript context injection

~~Medium~~ Marked As Solved

Identified on: May 6th 2026 | Resolved on: August 5th 2026

#### Description

A reflected cross-site scripting issue was identified in the blog article page where the route parameter is injected into inline JavaScript without safe JavaScript-context encoding. By requesting /blog/1%3Bconsole.log(424297718)%2F%2F, attacker-controlled code was executed in the browser and the XSS canary was triggered. This allows arbitrary script execution in the origin for any user who visits a crafted link.

#### Business impact

Arbitrary script execution in the application's origin enables an attacker to fully control the victim's browser session on the site. This includes reading and modifying page content, initiating authenticated actions via the application's APIs, and exfiltrating sensitive data present in the DOM. If the victim is logged in, the attacker's script can act with the victim's privileges, potentially accessing private notes, initiating backups, or modifying data. The likelihood of exploitation is high because this is a simple reflected XSS triggered by visiting a crafted URL, which can be embedded in emails, messages, or other pages. The impact ranges from user account misuse to data leakage and integrity loss. Depending on the data processed by the application, unauthorized access to personal data may trigger regulatory obligations under frameworks such as GDPR or CCPA. Overall severity is moderate-to-high for a public-facing application with authenticated functionality.

#### References

**CVSS 6.4:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:P/VC:N/VI:N/VA:N/SC:H/SI:H/SA:N

### 2.3.8 PT-8 - Header-based cache poisoning on /static via X-Content-Type

Medium

Identified on: May 6th 2026

#### Description

A cache poisoning issue was confirmed on `/static` because the response varies based on attacker-controlled `X-Content-Type`, while the cache key does not include request headers. A poisoned request using `X-Content-Type: %` triggered a 500 response and caused subsequent normal requests for the same URL to receive the poisoned cached response. This allows an unauthenticated attacker to poison cached static responses and disrupt content delivery to other users.

#### Business impact

Cached poisoning of static assets can degrade the user experience and availability of the application. If core assets such as CSS or JavaScript are served with an incorrect Content-Type, pages may render incorrectly, appear unstyled, or fail to execute scripts. If error responses are cached, pages can break altogether until the cache entry expires. Because shared caches affect many users, exploitation can disrupt a broad portion of the user base with minimal effort. Attackers can repeat the poisoning to maintain the impact over time. This can lead to perceived downtime, violation of uptime SLAs, increased support load, and reputational damage. Although this issue does not directly expose sensitive data, it constitutes a denial-of-service vector against content delivery. Depending on contractual obligations, prolonged disruption may have compliance or business ramifications related to service reliability and availability.

#### References

**CVSS 6.9:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:L/VA:L/SC:N/SI:N/SA:N

### 2.3.9 PT-9 - GraphQL endpoint accepts GET requests for query execution

Medium

Identified on: May 6th 2026

#### Description

The GraphQL endpoint was found to process query operations over HTTP GET. A request to `/graphql?query={__typename}` returned `200` with valid GraphQL data, confirming that queries can be executed through URL parameters. This weakens CSRF hardening and enables cross-site triggering of read operations via crafted links or embedded requests.

#### Business impact

Execution of GraphQL queries via GET increases exposure to cross-site request triggering. An attacker can coerce authenticated users to load a crafted URL, causing the server to execute attacker-chosen queries in the victim's context. Even without direct response access due to browser same-origin policy, this can facilitate targeted reconnaissance, timing-based side channels, or server-side effects (e.g., cache priming) that reveal sensitive patterns about a user's data or presence. Because queries and variables are carried in the URL, sensitive parameters may be recorded in web server logs, reverse proxy logs, CDN logs, browser history, analytics platforms, and possibly forwarded via Referer headers to third-party domains. This expands the blast radius of any sensitive data included in GraphQL variables. Likelihood is moderate, as exploitation only requires enticing a logged-in user to click a link or visit a page with an embedded resource that triggers a top-level navigation. The business impact depends on the sensitivity of data accessible through queries tied to an authenticated session; personal data exposure through logs or indirect leakage can result in privacy incidents and potential regulatory reporting obligations.

#### References

**CVSS 6.9:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N

### 2.3.10 PT-10 - Registration accepts trivial weak passwords without policy enforcement

**Medium**

Identified on: May 6th 2026

#### Description

The registration endpoint accepted the weak password `123456` and created an account successfully. A POST request to `/register` with a new username/email and that password returned `302` redirect to `/login`, indicating normal registration success instead of password-policy rejection. This confirms that password strength controls are not enforced during account creation.

#### Business impact

Permitting weak passwords substantially increases the likelihood of account takeover. Users are prone to choosing common, low-entropy passwords when not guided or restricted by policy. Adversaries can leverage credential stuffing (from other breach corpuses) and simple online guessing to compromise accounts at low cost. Compromise of individual accounts can lead to unauthorized access to personal data, notes, blog content, and any other user-scoped resources, with potential lateral movement if shared email addresses or reused passwords exist elsewhere. The issue also weakens the effectiveness of downstream controls such as rate limiting and monitoring, as attackers can succeed with fewer attempts when passwords are trivial. From a compliance and security-standards perspective, the behavior conflicts with guidance in OWASP ASVS and NIST SP 800-63B, which recommend minimum length requirements and screening against known-compromised password lists.

#### References

**CVSS 6.9:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N

### 2.3.11 PT-11 - GraphQL endpoint allows alias overloading without limits

Medium

Identified on: May 6th 2026

#### Description

The GraphQL endpoint accepted aliased batched queries without any alias-count restriction. Non-destructive tests with 10 and 30 aliases ( `__typename` ) both returned successful `200` responses containing all alias keys, with no limit-related errors. This enables request-level amplification and can be used to increase backend workload while bypassing simple per-request rate controls.

#### Business impact

Unrestricted alias overloading allows significant amplification of backend workload per request, which threatens service availability. By batching many aliased selections of heavy fields, an attacker can generate a large number of resolver and database executions with very few HTTP requests. This can degrade performance for legitimate users, cause resource contention, and in peak cases lead to partial outages or denial of service. The likelihood of exploitation is moderate to high because the technique is straightforward, requires no authentication, and is widely documented in GraphQL attack patterns. The impact is primarily on availability and operational cost rather than confidentiality or integrity; however, service-level objectives and customer experience can be adversely affected, and infrastructure costs may increase due to handling inflated workloads. Organizations with uptime commitments may face SLA implications if availability is impacted.

#### References

**CVSS 6.9:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:L/SC:N/SI:N/SA:N

### 2.3.12 PT-12 - GraphQL endpoint lacks query depth limiting

Medium

Identified on: May 6th 2026

#### Description

Deeply nested GraphQL queries were accepted and executed without any depth-limit enforcement. A recursive query chaining `users -> articles -> author -> articles` across multiple levels returned `200 OK` with full data instead of being rejected for excessive depth. This allows adversaries to submit high-cost nested queries that increase backend load and elevate denial-of-service risk.

#### Business impact

Unbounded query depth allows attackers to generate requests with exponential cost characteristics, leading to increased database I/O, CPU utilization, and memory pressure. This can manifest as slow responses, thread pool saturation, or outages of the API and dependent services. The impact extends to all users who rely on the application's GraphQL-backed features, potentially causing broad service disruption. The likelihood of exploitation is moderate to high given the endpoint is reachable without authentication and depth construction requires only knowledge of visible schema fields. In multi-tenant or shared infrastructure environments, resource exhaustion may cascade, affecting other workloads. While the issue primarily increases DoS exposure, execution of excessive resolver chains may also amplify secondary risks such as noisy log growth or operational instability. For organizations subject to availability-focused SLAs or regulatory obligations around service continuity, failure to mitigate resource exhaustion can contribute to SLA breaches and customer-impacting incidents.

#### References

**CVSS 6.9:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:L/SC:N/SI:N/SA:N

### 2.3.13 PT-13 - GraphQL introspection enabled on /graphql endpoint

**Medium**

Identified on: May 6th 2026

#### Description

GraphQL schema introspection was found to be enabled on the production endpoint. A POST request to /graphql with the __schema query returned the full schema metadata, including query and mutation root types and internal GraphQL meta types. This exposes API structure to attackers and materially lowers effort for targeted exploitation and enumeration.

#### Business impact

Exposed schema metadata materially lowers the effort required for attackers to enumerate the API and craft targeted requests. With full visibility into object types, fields, arguments, and available mutations, an attacker can quickly identify sensitive data fields (such as user emails or unpublished content), discover relationships between entities, and pinpoint potentially risky operations (e.g., mutations that create or modify records). This accelerates the discovery of authorization gaps, injection points, or other logic flaws if they exist. The likelihood of exploitation is high because introspection queries are trivial to run and require no authentication. While introspection alone does not modify data or directly expose record contents, it enables effective reconnaissance that increases the probability and speed of successful follow-on attacks. From a business perspective, this increases exposure to data leakage, integrity issues stemming from misuse of mutations, and service abuse through automated enumeration. Depending on subsequent exploitation of discovered operations, unauthorized access to personal or confidential data could create compliance implications for privacy regulations such as GDPR or CCPA.

#### References

**CVSS 6.9:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N

### 2.3.14 PT-14 - GraphQL field suggestions leak valid schema fields

Medium

Identified on: May 6th 2026

#### Description

GraphQL validation errors were found to include field-name suggestions for invalid queries. A request using an invalid field ( `userz` ) returned an error containing `Did you mean "user" or "users"?` , which discloses valid schema field names even without explicit introspection queries. This information disclosure aids endpoint and data-model enumeration for further attacks.

#### Business impact

Information leakage that reveals valid schema fields increases the likelihood of successful reconnaissance and targeted attacks against the API. Enumerated fields can be used to craft precise queries, identify sensitive data paths, and locate potentially risky mutations that could be abused if other controls are weak. The impact extends to accelerated endpoint discovery and more efficient fuzzing, which may lead to unauthorized data access if combined with missing authorization checks elsewhere. From a compliance perspective, unnecessary disclosure of internal API structure may contravene secure development and privacy-by-design principles expected in regulated environments.

#### References

**CVSS 6.9:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N

### 2.3.15 PT-15 - Authentication session cookie missing Secure flag

Low

Identified on: May 6th 2026

#### Description

The `connect.sid` session cookie used for authentication was configured without the `Secure` attribute. Provided cookie metadata and observed `Set-Cookie` headers show `HttpOnly` and `SameSite=Lax`, but no `Secure`, meaning the cookie can be transmitted over unencrypted HTTP if reached through a non-TLS path. This weakens session confidentiality and increases exposure to interception in mixed or downgraded transport scenarios.

#### Business impact

Loss of session confidentiality enables account impersonation. An attacker able to observe network traffic (e.g., on public Wi-Fi, corporate LAN, or any proxy in the path) can harvest the session identifier from an HTTP request and reuse it to access the victim's account, potentially reaching private notes, dashboards, and any stateful authenticated functionality. Administrative actions performed under a hijacked session can lead to data modification and reputational damage. The likelihood increases in environments without strict HTTPS enforcement where occasional HTTP exposure or downgrades can occur. Even if the primary application entry points are typically accessed via HTTPS, omitting Secure reduces defense-in-depth: a single mislinked http:// resource or configuration regression becomes sufficient for session exposure. Depending on jurisdiction, unauthorized access to user data through session theft can trigger notification obligations under privacy regulations (e.g., GDPR/CCPA).

#### References

**CVSS 6.9:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N

### 2.3.16 PT-16 - Full-read SSRF in /api/link-preview enables arbitrary outbound requests and internal address probing

Low

Identified on: May 6th 2026

#### Description

The /api/link-preview endpoint was found to accept user-controlled URLs and fetch them server-side without allowlisting or internal address restrictions. The fetched response body and headers are returned to the client, which enables full-read SSRF and allows probing of internal/link-local targets such as 169.254.169.254. Access to cloud metadata secrets was not demonstrated; requests to 169.254.169.254 returned non-sensitive error XML, so internal metadata compromise remains unverified.

#### Business impact

An attacker can use the application as an HTTP client to issue arbitrary outbound requests. This enables reconnaissance of internal networks and services reachable from the server (e.g., probing intranet hosts, link-local services, or metadata endpoints), potentially bypassing perimeter controls and IP-based allowlists. If sensitive internal HTTP services are exposed (databases with HTTP interfaces, admin panels, metadata endpoints), the attacker could retrieve data directly, leading to data exposure or facilitating lateral movement. The likelihood of exploitation is moderate to high because the endpoint is unauthenticated and easy to invoke. While this assessment did not retrieve cloud instance metadata or credentials, environments where the metadata service is accessible could face credential disclosure, privilege escalation, and broader compromise. Even without sensitive disclosure, the endpoint can be abused as an open proxy for malicious traffic and to map internal infrastructure, increasing operational and compliance risk.

#### References

**CVSS 8.7:** CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:L/SI:N/SA:N

### 2.3.17 PT-17 - DOM XSS in create-note link preview rendering via unescaped fetched response body

Low

Identified on: May 6th 2026

#### Description

The link preview feature on the create-note page was found to render server-returned preview content into `innerHTML` without output encoding. Because `/api/link-preview` fetches attacker-controlled URLs and returns the raw response body, HTML payloads can be injected into the preview container and executed in the browser context. JavaScript execution was confirmed with the XSS canary when previewing attacker-controlled content.

#### Business impact

Arbitrary JavaScript execution in the authenticated user's browser can lead to session hijacking (subject to cookie flags), theft of in-page data, CSRF token extraction, or unauthorized actions performed on behalf of the victim user. Sensitive note content visible to the user could be exfiltrated to an attacker-controlled endpoint. The likelihood is moderate where users can be socially engineered to preview attacker-controlled links (e.g., internal collaboration or support workflows). The impact is confined to the victim session and page context because the payload is not stored; however, successful exploitation still undermines confidentiality and integrity of user actions and data accessed during the session. Compliance concerns may arise if confidential user data can be exfiltrated via this vector.

#### References

**CVSS 6.2:** CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:A/VC:N/VI:N/VA:N/SC:H/SI:H/SA:N

### 2.3.18 PT-18 - Weak TLS cipher suites accepted

Low

Identified on: May 6th 2026

#### Description

The service accepts suboptimal TLS cipher suites or parameters that are not outright broken but weaken confidentiality and forward secrecy. Examples include AES-CBC suites under TLS 1.2, RSA key exchange (TLS_RSA, no forward secrecy), SHA-1 signatures, finite-field DH parameters smaller than 2048 bits, or legacy/weak elliptic curves. These choices increase exposure to downgrade and cross-protocol attacks and are deprecated by modern security guidance. Many Third-Party Risk Management (TPRM) platforms flag this as a medium-to-high risk that can lower external security scores and slow vendor assessments and insurance renewals.

#### Business impact

This issue can lead to data breaches, regulatory non-compliance, and erosion of customer trust.

#### Identified Configuration

```
{
  "TLS 1.2": {
    "weak_ciphers": [
      "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA",
      "TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA"
    ]
  }
}
```

### 2.3.19 PT-19 - Missing or Misconfigured Security Headers

Low

Identified on: May 6th 2026

#### Description

The application is missing important HTTP security headers or has them misconfigured. Security headers provide defense-in-depth against common web attacks including clickjacking, MIME-sniffing, and protocol downgrade attacks.

#### Identified Configuration


```
[
 {
  "rule_id": "strict_transport_security_missing",
  "header_name": "Strict-Transport-Security",
  "header_value": ""
 },
 {
  "rule_id": "server_version_disclosed",
  "header_name": "Server",
  "header_value": "nginx/1.29.8"
 },
 {
  "rule_id": "x_content_type_options_missing",
  "header_name": "X-Content-Type-Options",
  "header_value": ""
 },
 {
  "rule_id": "content_security_policy_missing",
  "header_name": "Content-Security-Policy",
  "header_value": ""
 }
]
```

# Appendices

## A. Scope & Methodology

### Methodologies

The penetration test followed a structured engagement lifecycle aligned with industry-standard methodologies, including OWASP Testing Guide v4.2, PTES, NIST SP 800-115, OWASP (M)ASVS, and the OWASP AI Testing Guide, including agentic behavior testing.

- Engagement scope and configuration were validated prior to and during testing
- Findings were reviewed for accuracy and severity classification
- Results can be escalated for further analysis upon request
- Methodology is continuously refined based on emerging threat intelligence

### Penetration test types

| | Black box | Grey box | White box |
|---|---|---|---|
| **Goal** | Assess security defences against a cyber attack | Assess security defences against a cyber attack | Assess security defences against a cyber attack |
| **Access level** | Zero access or internal information | Some internal access and internal information | Complete open access to applications and systems |
| **Pros** | Most realistic<br>Testing is performed from point of view of attacker | More efficient and complete than Black Box while saving time and money<br>Testing is performed from point of view of attacker | More comprehensive, less likely to miss a vulnerability<br>Testing is performed from point of view of attacker |
| **Cons** | Time consuming and more likely to miss a vulnerability | More hidden vulnerabilities might be missed | More action required by client to provide information and more time necessary to process documents |

# B. Vulnerability Coverage

## OWASP Top 10 Compliance

The penetration test covers the OWASP Top 10 vulnerability categories and includes agentic behavior testing aligned with the OWASP AI Testing Guide, helping identify critical web application and AI-agent behavior risks.

## Tested Vulnerability Classes

The penetration test explicitly checked for the following vulnerability types:

| Injection & Execution | Authentication & Access Control |
|---|---|
| SQL, NoSQL, XPath, & LDAP Injection | Broken or Improper Authentication |
| Cross-Site Scripting (XSS) | Missing Authentication for Critical Functions |
| Server-Side Request Forgery (SSRF) | Insecure Direct Object Reference (IDOR) |
| Server-Side Template Injection (SSTI) | Improper Access Control |
| Local File Inclusion (LFI) | Cross-Site Request Forgery (CSRF) |
| Insecure Deserialization | Excessive Authentication Attempts |

| Data Security & Logic | Configuration, Files & Cryptography |
|---|---|
| Business Logic Flaws | Unrestricted File Uploads |
| Exposure of Sensitive Information | External Control of File Name or Path |
| Information Exposure via Errors | Open Redirects |
| Directory Listing Enabled | Improper JWT Signature Verification |
| Web Cache Poisoning | Broken or Risky Cryptographic Algorithms |
| Insufficient Data Authenticity | Hard-Coded Passwords or Credentials |
| Reliance on Cookies Without Integrity | Insufficiently Protected Credentials |

## C. Glossary

| | |
|---|---|
| **Authentication** | The process of verifying the identity of a user, device, or system before granting access to resources. |
| **Authorization** | The process of determining what permissions an authenticated user has and what resources they can access. |
| **CVSS** | Common Vulnerability Scoring System - A standardized method for rating the severity of security vulnerabilities. |
| **IDOR** | Insecure Direct Object Reference - When an application provides direct access to objects based on user-supplied input without proper authorization checks. |
| **PII** | Personally Identifiable Information - Any data that could potentially identify a specific individual. |
| **SQL Injection** | A code injection technique where malicious SQL statements are inserted into application data entry points. |
| **XSS** | Cross-Site Scripting - A vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. |
| **WAF** | Web Application Firewall - A security device that monitors, filters, and blocks HTTP traffic to and from a web application. |