# WEB APPLICATION PENETRATION TESTING REPORT

Target Application: mypentest.me
URL: http://mypentest.me
Assessment Type: Grey Box
Assessment Window: 2 February 2026 – 3 February 2026
Environment: Production
Assessor: Rakib Mahmud Nadir

## Table of Contents

## Executive Summary

A Grey Box web application penetration test was conducted against http://mypentest.me, a website owned and maintained by the assessor. The assessment was performed between 2 February 2026 and 3 February 2026 and was limited to the scope, procedures, and information available during this timeframe. Any changes made to the application after this period may affect the accuracy of the findings presented in this report.

The objective of this engagement was to identify exploitable vulnerabilities and security misconfigurations within the web application by simulating realistic attack scenarios with partial knowledge of the system. Valid application credentials were provided prior to testing, allowing assessment of both authenticated and unauthenticated attack surfaces.

The testing methodology followed industry-recognized standards and best practices, including the OWASP Top 10, OWASP Application Security Verification Standard (ASVS), OWASP Testing Guide, NIST, SANS, and the Penetration Testing Execution Standard (PTES). This assessment was not intended to identify every possible vulnerability, but rather to provide a practical, risk-focused evaluation of the application's security posture.

A combination of automated tools and manual testing techniques was used during the engagement. Manual testing was emphasized to validate exploitability, identify business logic flaws, and reduce false positives. The tools and techniques used during the assessment are documented later in this report.

The findings described herein represent the state of the application at a single point in time. Changes to application logic, configuration, or infrastructure may impact on the relevance or severity of the identified issues.

## Background

This exercise was conducted to perform a Grey Box penetration test of the web application in scope to determine its susceptibility to attacks and exploitation. The testing primarily consisted of manual techniques to identify and validate security vulnerabilities.

The assessment aimed to identify security weaknesses within the application and provide recommendations to remediate the identified issues, thereby improving the overall security posture of the application.

The security assessment was performed by the assessor against http://mypentest.me between 2 February 2026 and 3 February 2026.

## Objectives

The objectives of the penetration testing activities were to:

i. Identify potential security vulnerabilities and gaps within the web application.
ii. Assess the identified vulnerabilities to determine the associated risk.
iii. Provide recommendations to remediate the identified vulnerabilities and security gaps.

This assessment report contains:

i. Technical details of the vulnerabilities identified, including evidence and proof-of-concept where applicable.
ii. Risk mitigation recommendations are required to reduce the risks arising from the discovered vulnerabilities.

## Scope of Assessment

The web application mypentest.me and its supporting infrastructure were subjected to a security-focused Grey Box penetration test.

The scope of the assessment included the following asset:

| URL | IP Address |
|---|---|
| http://mypentest.me | Not Public-facing (Internet-Inaccessible) |

## Out of Scope

The following activities were considered out of scope for this engagement:

i. Social engineering
ii. Denial of Service (DoS) attacks
iii. Vulnerability remediation or fixes
iv. Intrusive or destructive exploitation

As the assessment was conducted against a production environment, the following tests were not performed to avoid potential service disruption or data impact:

i. Denial of Service (DoS) attacks
ii. Brute force attacks
iii. Directory and content fuzzing
iv. Exploits with known destructive consequences
v. Data manipulation or deletion
vi. Excessive traffic generation
vii. Use of unverified third-party tools or scripts
viii. Social engineering attacks
ix. Exhaustive automated vulnerability scanning
x. Security testing without proper authorization

## Tools Used

The assessment was performed using a combination of manual testing techniques and widely used open-source and security testing tools. Manual testing was prioritized to validate findings and reduce false positives.

The following tools were used during the assessment:

i. Burp Suite
ii. Mozilla Firefox Developer Tools and Plugins
iii. Kali Linux
iv. jwt_tool

## Engagement Summary

A Grey Box web application penetration test was conducted against mypentest.me between 2 February 2026 and 3 February 2026. The objective of this engagement was to assess the security posture of the application by identifying vulnerabilities that could be exploited by an attacker with partial knowledge of the system. The assessment focused on both authenticated and unauthenticated attack surfaces and was performed using a combination of manual testing techniques and industry-standard security tools. Testing activities were aligned with recognized best practices, including OWASP and PTES methodologies. The engagement identified multiple security weaknesses across authentication, input handling, token management, and transport security. Several of these findings pose a risk of unauthorized access and account compromise if left unaddressed. All findings were validated during the assessment period and documented with supporting evidence. The results of this engagement provide a clear view of the application’s current security posture and highlight areas requiring remediation. The recommendations included in this report are intended to support effective risk reduction and improve the overall resilience of the application against real-world attack scenarios.

| FEBRUARY | | | | | | |
|---|---|---|---|---|---|---|
| M | T | W | T | F | S | S |
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | | | | | | |

## Methodology

The web application security assessment was conducted using a structured and systematic testing methodology aligned with industry-recognized standards and best practices, including the OWASP Testing Guide, OWASP Top 10, OWASP Application Security Verification Standard (ASVS), and the Penetration Testing Execution Standard (PTES).

The engagement followed a Grey Box testing approach, with valid application credentials provided prior to testing. This allowed for assessment of both authenticated and unauthenticated application functionality, simulating realistic attacker scenarios with partial knowledge of the system.

Testing activities primarily focused on manual analysis, supported by industry-standard security testing tools. Manual testing was emphasized to identify complex vulnerabilities, validate exploitability, and reduce false positives that automated tools may overlook.

The assessment covered key areas of the application, including but not limited to authentication and authorization controls, session management, input validation, business logic, token handling, and transport security. Identified issues were validated, analyzed for impact and likelihood, and documented with supporting evidence.

All testing was performed within the defined scope and timeframe and was conducted in a controlled manner to avoid service disruption or data integrity issues.

Discovery & Analysis
Testing & Validation
Risk Assessment & Reporting

## Summary of Findings

The Grey Box Web Application Penetration Testing conducted against mypentest.me identified multiple security vulnerabilities that could be exploited to compromise user accounts and application security. This section provides a high-level summary of the key findings identified during the assessment period.

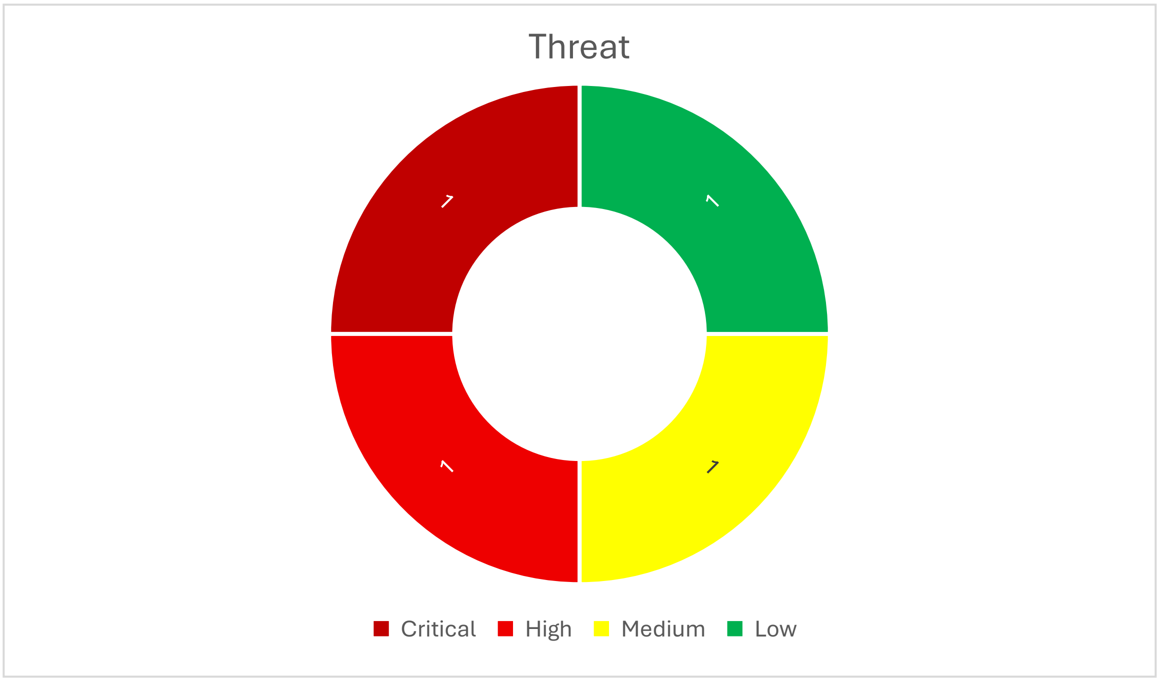
A total of 4 security vulnerabilities were discovered with the following severity distribution:

i. Critical: 1
ii. High: 1
iii. Medium: 1
iv. Low: 1

All identified vulnerabilities were unresolved at the time of assessment.

Findings Overview

| S. No. | Vulnerability Name | Severity | Impact | Status |
|---|---|---|---|---|
| 1 | Weak JWT secret leading to account takeover | Critical | Account takeover | Unresolved |
| 2 | Cross-Site Scripting (XSS) | High | Session hijacking / client-side code execution | Unresolved |
| 3 | Authentication weakness | Medium | Unauthorized access | Unresolved |
| 4 | Unencrypted communication (Missing HTTPS / insecure transport) | Low | Data exposure in transit | Unresolved |



## Vulnerability Details

### 1.1 Weak JWT secret leading to account takeover

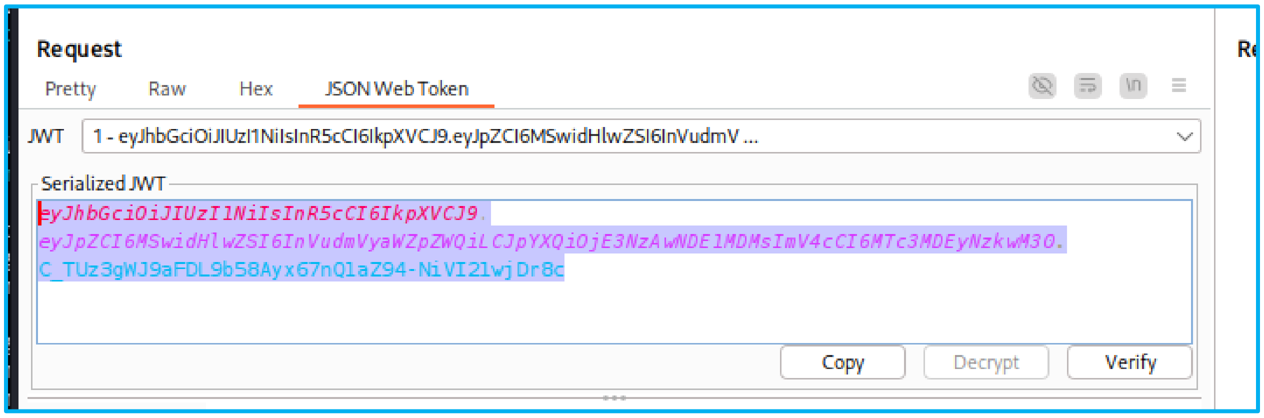
| Parameter | Description |
|---|---|
| Severity | Critical |
| Impact | Critical |
| Risk Score | 9.1 |
| Affected URL | http://mypentest.me |
| Threat | A weak or predictable JWT signing secret allows an attacker to forge valid tokens and impersonate legitimate users. Successful exploitation can result in:<br>1. Account takeover<br>2. Privilege escalation<br>3. Unauthorized access to sensitive data |

JSON Web Token (JWT) vulnerabilities arise when authentication tokens are improperly implemented, particularly when weak, predictable, or misconfigured signing secrets are used. JWTs are designed to securely transmit user identity and authorization claims; however, improper configuration can allow attackers to manipulate or forge tokens.

During the assessment, it was observed that the application utilized a weak JWT signing secret. An attacker could exploit this weakness to generate valid forged tokens and impersonate legitimate users. Successful exploitation could result in account takeover, privilege escalation, and unauthorized access to sensitive application data, all without requiring valid user credentials.

## PROOF OF CONCEPT

1. Getting a Jwt token in the request

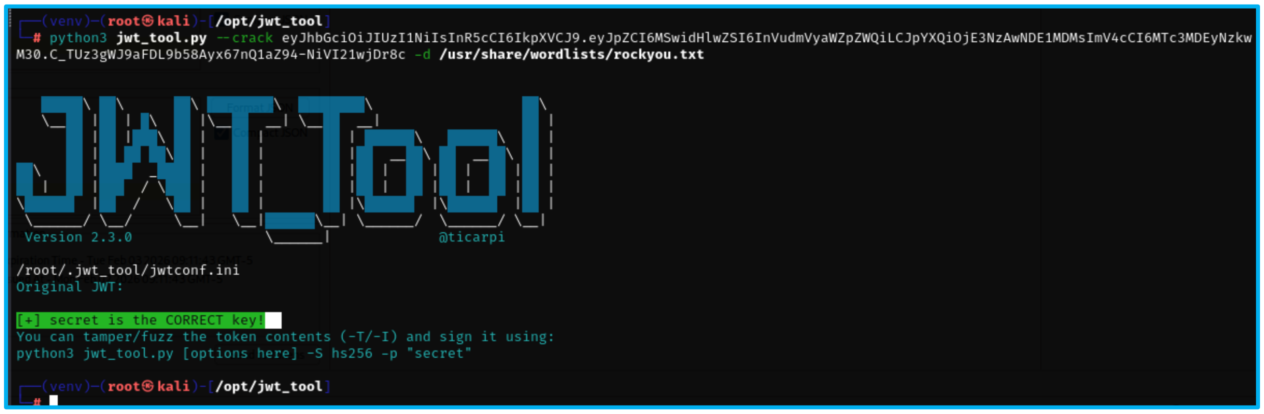


JWT 1 - eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwidHlwZSI6InVudmV ...

Serialized JWT

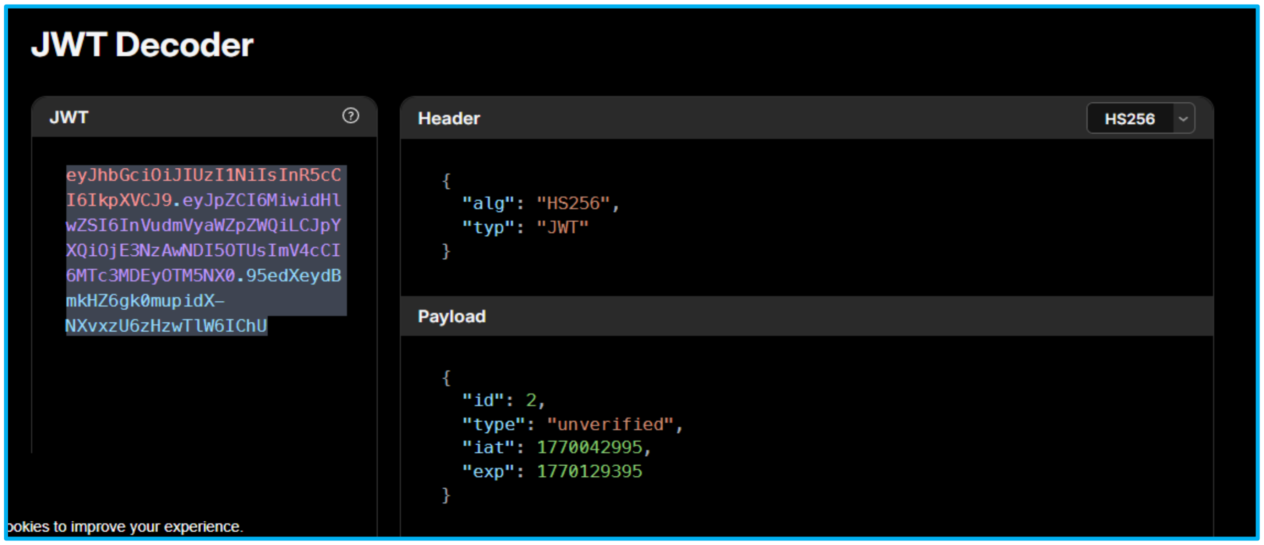
```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6MSwidHlwZSI6InVudmVyaWZpZWQiLCJpYXQiOjE3NzAwNDE1MDMsImV4cCI6MTc3MDEyNzkwM30.
C_TUz3gWJ9aFDL9b58Ayx67nQ1aZ94-NiVI21wjDr8c
```

2. Tried To crack the secret with the jwt_tool using the rockyou.txt Wordlist.



```
┌──(venv)─(root㉿kali)-[/opt/jwt_tool]
└─# python3 jwt_tool.py --crack eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwidHlwZSI6InVudmVyaWZpZWQiLCJpYXQiOjE3NzAwNDE1MDMsImV4cCI6MTc3MDEyNzkw
M30.C_TUz3gWJ9aFDL9b58Ayx67nQ1aZ94-NiVI21wjDr8c -d /usr/share/wordlists/rockyou.txt

 Version 2.3.0                                         @ticarpi

/root/.jwt_tool/jwtconf.ini
Original JWT:

[+] secret is the CORRECT key!
You can tamper/fuzz the token contents (-T/-I) and sign it using:
python3 jwt_tool.py [options here] -S hs256 -p "secret"

┌──(venv)─(root㉿kali)-[/opt/jwt_tool]
└─#
```

3. Created a jwt token from the Valid secret we found earlier

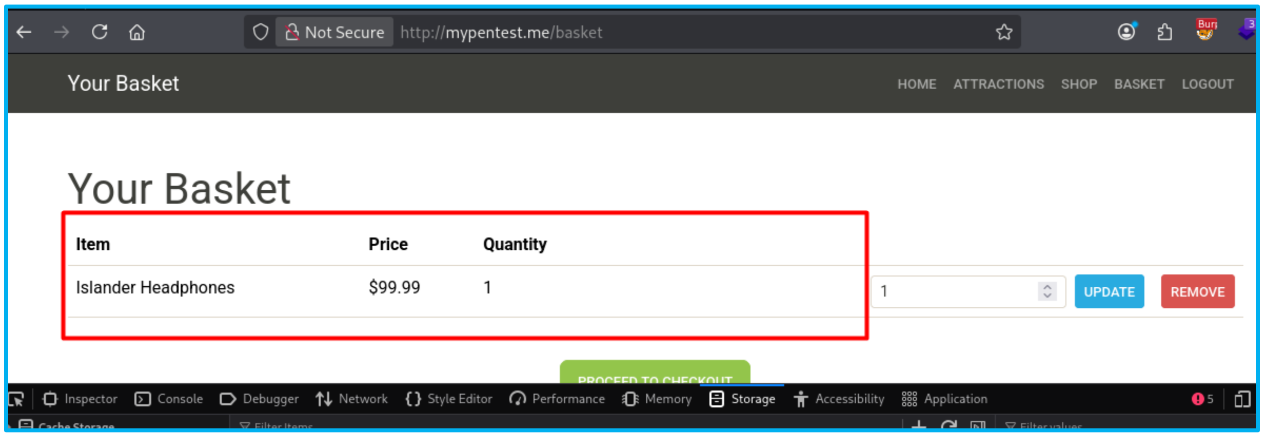


```
eyJhbGciOiJIUzI1NiIsInR5cC
I6IkpXVCJ9.eyJpZCI6MiwidHl
wZSI6InVudmVyaWZpZWQiLCJpY
XQiOjE3NzAwNDI5OTUsImV4cCI
6MTc3MDEyOTM5NX0.95edXeydB
mkHZ6gk0mupidX-
NXvxzU6zHzwTlW6IChU
```

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

```
{
  "id": 2,
  "type": "unverified",
  "iat": 1770042995,
  "exp": 1770129395
}
```

4. Logged In as Another user . The browser doesn't shows the accurate username so I add unique items from the other user to identify newly logged in user.



Your Basket

| Item | Price | Quantity |
|---|---|---|
| Islander Headphones | $99.99 | 1 |

## RESOLUTION OF VULNERABILITY

To mitigate the risks associated with weak or improperly implemented JSON Web Tokens (JWT), the application's authentication and token handling mechanisms should be strengthened in line with industry best practices.

First, JWT signing keys must be sufficiently strong and securely managed. Weak or predictable secrets should be replaced with cryptographically strong keys generated using an approved random source. Signing keys should be stored securely using environment variables or a centralized secrets management solution and rotated periodically to limit the impact of potential key compromise.

Second, the application should strictly enforce the use of secure signing algorithms. The allowed algorithm must be explicitly defined and validated during token verification, and insecure or unused algorithms must be disabled. Where feasible, asymmetric signing algorithms such as RS256 should be preferred to improve key separation and reduce the risk of token forgery.

Third, comprehensive validation of JWT claims should be implemented. The application should validate mandatory claims including issuer (iss), audience (aud), expiration (exp), and issued-at (iat). Tokens that are expired, malformed, or missing required claims should be rejected. Short token lifetimes should be enforced to reduce the window of abuse in the event of token leakage.

Finally, JWTs must be protected during transmission and storage. Tokens should only be transmitted over secure HTTPS connections to prevent interception. Storage mechanisms should be carefully chosen to minimize exposure to client-side attacks, with secure, HttpOnly cookies preferred where applicable. In addition, implementing token revocation or monitoring mechanisms can help detect and respond to suspicious token usage.

## REFERENCE

https://portswigger.net/web-security/jwt

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/10-Testing_JSON_Web_Tokens

https://www.acunetix.com/blog/articles/json-web-token-jwt-attacks-vulnerabilities/

## 1.2 Cross-Site Scripting (XSS)

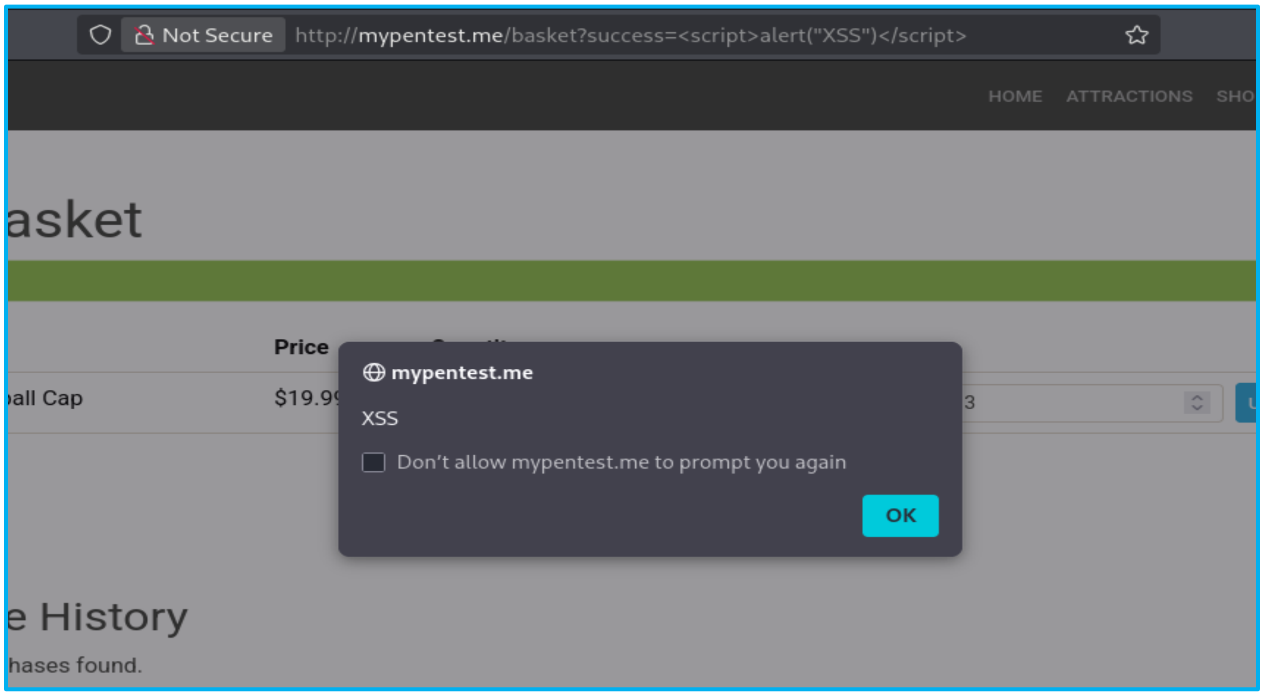
| Parameter | Description |
|---|---|
| Severity | High |
| Impact | High |
| Risk Score | 7.1 |
| Affected URL | http://mypentest.me |
| Threat | Improper input validation allows an attacker to inject and execute malicious JavaScript in a victim's browser. Successful exploitation may result in:<br>1. Session hijacking<br>2. Account compromise<br>3. Execution of unauthorized actions on behalf of the user<br>4. Delivery of malicious content |

Cross-Site Scripting (XSS) is a web security vulnerability that occurs when an application fails to properly validate or sanitize user-supplied input before rendering it in a web page. This allows an attacker to inject and execute malicious client-side scripts in the browser of unsuspecting users.

During the assessment, it was observed that user-controlled input could be injected into the application and executed as JavaScript in the victim's browser. Successful exploitation of this vulnerability could allow an attacker to steal session tokens, impersonate users, perform actions on behalf of victims, or deliver malicious content, potentially leading to session hijacking or account compromise.

### PROOF OF CONCEPT

1. Executed the JavaScript Payload <script>alert("XSS")</script>



Not Secure http://mypentest.me/basket?success=<script>alert("XSS")</script>

mypentest.me

XSS

Don't allow mypentest.me to prompt you again

## RESOLUTION OF THE VULNERABILITY

To remediate the identified Cross-Site Scripting (XSS) vulnerability, the application must ensure that all user-controlled input is handled securely throughout its lifecycle, from ingestion to rendering.

The application should implement context-aware output encoding all dynamic content rendered in the browser. User-supplied input, including URL parameters such as success parameter, must be properly encoded based on the context in which it is displayed (e.g., HTML body, HTML attributes, or JavaScript context). This prevents injected scripts from being interpreted and executed by the browser.

In addition, server-side input validation should be enforced to restrict parameters to expected formats and values. Parameters intended to display status messages or flags should be validated against a strict allowlist and rejected if unexpected input is received. Client-side validation alone is insufficient and should not be relied upon as a primary defense.

The application should also implement a strong Content Security Policy (CSP) to limit the execution of unauthorized scripts. A well-defined CSP can significantly reduce the impact of XSS vulnerabilities by restricting script sources and disallowing inline script execution.

Finally, development teams should adopt secure coding practices and conduct regular security testing, including code reviews and penetration testing, to identify and remediate XSS vulnerabilities early in the development lifecycle.

## REFERENCE

**https://owasp.org/www-community/attacks/xss/**

**https://portswigger.net/web-security/cross-site-scripting**

**https://developer.mozilla.org/en-US/docs/Web/Security/Attacks/XSS**

## 1.3 Insufficient Authentication Mechanism

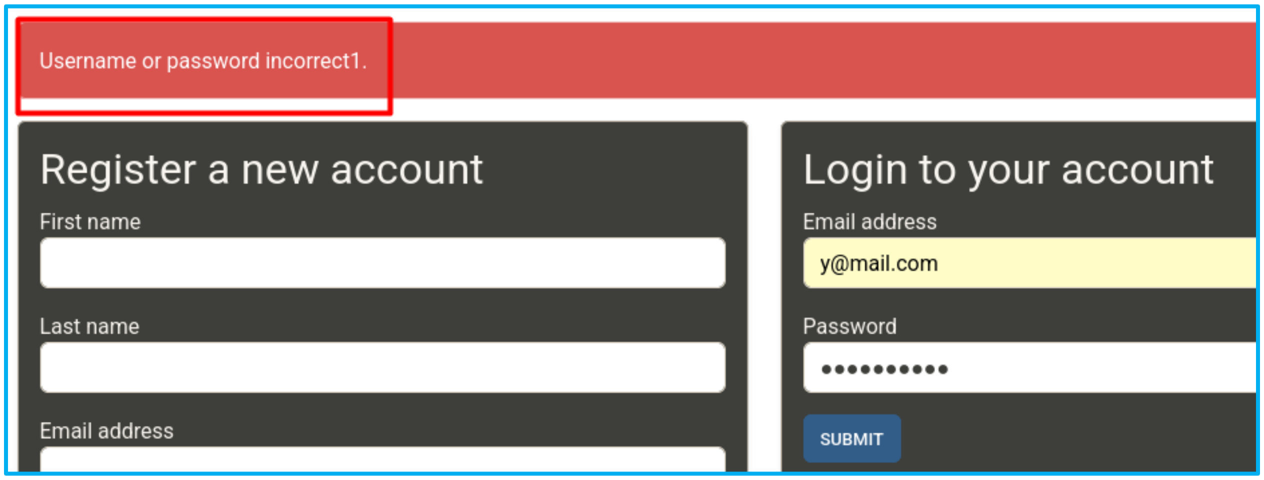
| Parameter | Description |
|---|---|
| Severity | Medium |
| Impact | Unauthorized access to user accounts |
| Risk Score | 6.5 |
| Affected URL | http://mypentest.me |
| Affected Component | Authentication / Login mechanism |

Authentication vulnerabilities occur when an application fails to properly verify the identity of users or enforce adequate controls around the login process. Weak authentication mechanisms can allow attackers to bypass login protections, brute-force credentials, reuse stolen credentials, or gain unauthorized access to user accounts.

During the assessment, it was observed that the application’s authentication controls were insufficiently implemented, allowing an attacker to abuse weaknesses in the login process. Successful exploitation of this vulnerability could result in unauthorized access, account compromise, and potential escalation of privileges within the application.

## PROOF OF CONCEPT

1. Username can be enumerated via the response


Username or password incorrect1.

Register a new account

First name

Last name

Email address

Login to your account

Email address

y@mail.com

Password

••••••••••

## RESOLUTION OF THE VULNERABILITY

To remediate the identified authentication vulnerability, the application's authentication mechanisms should be strengthened in accordance with industry's best practices.

The application should ensure that all user credentials are handled securely. Passwords must be stored using strong, adaptive hashing algorithms such as bcrypt, Argon2, or PBKDF2, and plaintext storage or transmission of credentials must be strictly avoided. Secure password policies should be enforced to require adequate complexity and length.

To prevent brute-force and credential-stuffing attacks, rate limiting and account lockout controls should be implemented on authentication endpoints. Repeated failed login attempts should trigger temporary lockouts or progressive delays to reduce the feasibility of automated attacks.

Authentication error handling should be reviewed to ensure that generic error messages are returned for failed login attempts. The application should not disclose whether a username or password is valid, as this could enable user enumeration.

Where applicable, the use of multi-factor authentication (MFA) should be considered, particularly for privileged accounts or sensitive functionality, to provide an additional layer of security beyond passwords alone.

Finally, authentication-related events should be logged and actively monitored to detect suspicious behavior and support timely incident response.

## REFERENCE

https://portswigger.net/web-security/authentication

https://owasp.org/Top10/2025/A07_2025-Authentication_Failures/

https://www.strongdm.com/blog/authentication-vulnerabilities

## 1.4 Unencrypted Communication

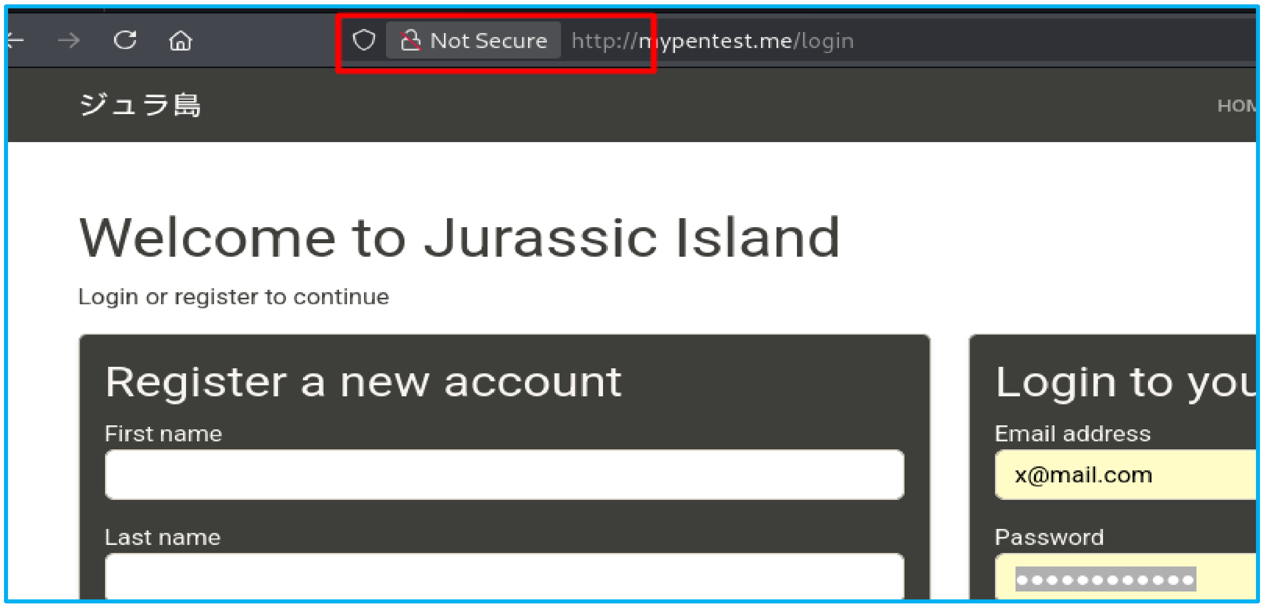
| Parameter | Description |
|---|---|
| Severity | Low |
| Impact | Exposure of sensitive data in transit |
| Risk Score | 3.0 |
| Affected URL | http://mypentest.me |
| Affected Component | Network communication / Transport layer |

Unencrypted communication occurs when an application transmits sensitive data over an insecure channel, such as HTTP, instead of using encrypted transport mechanisms like HTTPS. Without encryption, data transmitted between the client and server can be intercepted, modified, or replayed by an attacker positioned on the network.

During the assessment, it was observed that the application allowed communication over an unencrypted channel. This exposes authentication tokens, session identifiers, and user-supplied data to interception by attackers performing man-in-the-middle (MITM) attacks.

### PROOF OF CONCEPT

1. Request going through http



### RESOLUTION OF THE VULNERABILITY

To remediate this issue, the application must enforce secure communication across all endpoints.

All HTTP traffic should be redirected to HTTPS using a valid TLS configuration. The server should be configured to disable insecure protocols and ensure that modern TLS versions are used.

The application should also implement HTTP Strict Transport Security (HSTS) to instruct browsers to communicate only over HTTPS, preventing protocol downgrade attacks.

Sensitive cookies must be configured with the Secure and HttpOnly flags to ensure they are transmitted only over encrypted connections and are inaccessible to client-side scripts.

Regular reviews of TLS configuration and certificate management should be conducted to ensure ongoing compliance with security best practices.

### REFERENCE

**https://portswigger.net/kb/issues/01000200_unencrypted-communications**

**https://beaglesecurity.com/blog/vulnerability/information-sent-using-unencrypted-channels.html**

**https://www.hacksplaining.com/lessons/unencrypted-communication/start**

## Conclusion

The Grey Box web application penetration test conducted against mypentest.me provided a detailed assessment of the application's current security posture. The engagement identified multiple security weaknesses spanning authentication controls, input validation, session and token management, and transport-layer security. While some findings were categorized as low risk, several issues demonstrated realistic attack paths that could lead to unauthorized access, account takeover, and exposure of sensitive information if exploited by a motivated attacker.

The assessment highlights the importance of implementing consistent and defense-in-depth security controls across the application. Weaknesses in authentication and token handling represent a higher level of risk, as they directly impact user identity and access control. The presence of client-side vulnerabilities, such as Cross-Site Scripting (XSS), further increases the potential impact by enabling chained attacks when combined with other weaknesses.

Overall, the application demonstrates a functional security baseline; however, improvements are required to align it with industry best practices and reduce exposure to common web application attack techniques. Addressing the identified vulnerabilities will significantly improve the application's resilience and reduce the likelihood of successful exploitation.

## Remediation Summary

Remediation efforts should be prioritized based on the severity and potential impact of the identified vulnerabilities. Critical and high-severity findings, particularly those related to authentication, JWT implementation, and account takeover scenarios, should be addressed immediately. Medium- and low-severity issues should be remediated in a timely manner to further reduce the overall attack surface.

Key remediation actions include strengthening authentication mechanisms, enforcing secure password handling, implementing robust token validation and signing practices, and ensuring all user input is properly validated and output encoded. Transport-layer security should be consistently enforced across the application by mandating HTTPS, configuring secure cookies, and implementing HTTP Strict Transport Security (HSTS) where applicable.

In addition to technical fixes, it is recommended that secure development practices be integrated into the application lifecycle. This includes regular code reviews, dependency management, and periodic security testing to identify and address vulnerabilities early. Once remediation activities are complete, a follow-up penetration test or targeted re-assessment should be conducted to validate the effectiveness of the implemented controls and confirm that no new security issues have been introduced.