Technical Penetration Test Report
(White-box and Black-Box)

Table of content:

Risk rating.

Each identified weakness is evaluated using the CVSS v3.1 Base Score (Common Vulnerability Scoring System v3.1).The resulting score is used to classify the risk level according to the predefined severity scale (Low, Medium, High, Critical).

0.0 – 3.9:	Low
4.0 – 6.9:	Medium
7.0 – 8.9:	High
9.0 – 10.0:	Critical

Risk Dating Methodology

The identified weakness was rated using CVSS v3.1. Based on the calculated base score of **6.1**, the issue falls into the **Medium** risk category according to the adopted scoring scale. No High or Critical risk vulnerabilities were identified during the assessment.

1. Engagement Metadata

- **Target:** Production WordPress-based web application
- **Test Type:** Combined Black-Box and White-Box Penetration Test
- **Access Level:** No administrative accounts, no SSH access, no direct database access.
- **Testing Period:** January 2026
- **Methodology:** OWASP Web Security Testing Guide (WSTG) OWASP top 10
- **Testing Nature:** Non-destructive, no brute force, np. denialof-service

2. Scope Definition

2.1 In Scope

- Web application (HTTP/HTTPS)
- WordPress backend and REST/AJAX endpoints
- Reservation form business logic
- Input validation and workflow enfocement
- TLS configuration
- Network and backend exposure

2.2 Out of Scope

- Social engineering
- Zero-day exploitation
- Destructive testing
- Full source code audit outside reservation logic

3. Tools and Environment

- Kali Linus (virtual environment)
- Burp Suite (HTTP proxy and request analysis)

- Nmap (network exposure testing)
- testssl.sh (TLS configuration analysis)
- ffuf (controlled path and file discovery)

4. Executive Technical Summary

The tested WordPress application demonstrates a high level of security maturity. No critical or high-risk vulnerabilities were identified that would allow system compromise, unauthorized database access, or administrative account takeover.

The application correctly enforces backend business logic, does not trust client-side input for security decisions, and effectively prevents abuse of the reservation workflow.

The only identified risk relates to semantic input validation, which may impact data integrity and introduce secondary risks in administrative contexts.

5. Technical Findings

5.1 Application Architecture

- CMS: WordPress
- Backend entry point: POST /wp-admin/admin-ajax.php
- Custom AJAX action: zapisz_rezerwacje
- Frontend uses fetch() stristy as a transport layer
- no dynamic backend selection possible from UI or JavaScirpt

5.2 admin-ajax.php Security Behavior

- Requests without the `action` parameter return HTTP 400 / `0`
- No fallback handlers or default execution paths

- No possibility to enumerate registered AJAX actions **Assessment:**

  Secure, hardened behavior consistent with WordPress best

  practices.

5.3 Backend Validation and Workflow Enforcement

Confirmed behaviors:

- Required fields validated server-side
- Business objects validated via database lookup
- Invalid submissions do not result in database writes
- Replayed requests reach the backend but do not persist data

Conclusion: Backend enforces workflow/state integrity (nonce/state-lock/sequence control).

5.4 Mass Assignment Analysis

- Logical and privileged fields are ignored when supplied by the client
- Sensitive values are assigned exclusively server-side

Conclusion: Explicit field whitelisting or mapping is in place. No mass assignment vulnerability identified.

5.5 Identified Finding: Semantic Input Validation

Description:

The backend accepts syntactically valid but semantically incorrect values, including:

- Excessively long strings (e.g., name fields)
- Illogical formats (e.g., phone numbers)
- Arbitrary values for enum-like parameters

If business rules are satisfied, such data is persisted to the database.

Impact:

- Reduced data integrity
- Potential secondary risks (stored XSS in admin panels, reports, exports or notification emails)

Classification: Improper Input Validation / business Logic Weakness
Severity: Medium
CVSS V3.1: 6.1 (AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N)

5.6 Explicitly Excluded Vulnerabilities

The following ussues were tested for and not identified:

- SQL Injection (classic or blind)
- Insecure Direct Object References (DIOR)
- Remote Code Execution (RCE)
- Workflow bypass
- forces database writes

6. Technical Recommendations

- **Server-side input length enforcement**
  Define and enforce strict maximum length limits for all user-controlled input fields before data persistence.
- **Format validation for structured fields**
  Apply explicit server-side format validation for structured data types (e.g. phone numbers, email addresses) using allowlisted patterns.

- **Allowlisting for enum-like parameters**
  Restrict enum-like parameters to a predefined set of allowed values and reject any unexpected input.
- **Input normalization prior to persistence**
  Normalize all input values (e.g. trimming, canonicalization) before storing them in the database.
- **Context-aware output encoding**
  Ensure proper output encoding and sanitization when rendering stored data in administrative panels, reports, exports, and email notifications to prevent secondary injection risks.

Remediation Status

All recommended remediation actions have been implemented. The identified issue related to data quality and secondary risk vectors has been remediated through improved server-side validation without requiring architectural changes.

7. Summary

No vulnderabilities were identified that would enable system compromise, privilege escalation, or unathorized data access. The application demonstates well-designed backend security model with effective workflow enforcement.
The identified issue related to data quality and secondary risk vectors has been remediated through improved server-side input validation and does not require any architectural changes.

Author: Patryk Chwalik