Web Application Testing Report

Target Application

Hack Yourself First (https://hack-yourself-first.com)

Prepared

By

Kalyani Patil

Contents

1. Executive Summary

This report documents the results of a manual web application security assessment conducted on the **Hack Yourself First** website. The assessment was performed as part of a learning and evaluation task to identify common security weaknesses in a web application.

The testing was limited to publicly accessible features and was carried out using basic manual techniques such as URL manipulation, form input testing, error observation, and HTML source review. No automated tools or destructive testing methods were used.

During the assessment, **16 security issues** were identified across areas including access control, authentication, input validation, error handling, and security configuration. Two high-severity issues related to **broken access control** and **user enumeration** were observed, along with several medium and low-severity findings.

Although the application is intentionally insecure for educational purposes, the identified issues represent real-world security risks commonly found in production web applications.

2. Objective and Scope

2.1 Objective

The objective of this security assessment was to identify common web application security weaknesses using manual testing techniques. The assessment aimed to demonstrate basic security testing skills, understand insecure application behavior, and document findings in a structured and professional manner.

2.2 Scope of Testing

The scope of this assessment included testing of **publicly accessible functionality** of the web application. The following activities were performed:

- Manual browsing of the application
- URL parameter manipulation
- Form input testing
- Error message observation
- HTML page source review

2.3 Out of Scope

The following activities were not included in this assessment:

- Automated vulnerability scanning
- Denial-of-service testing
- Brute-force or password attacks
- Source code review
- Backend or infrastructure testing

3. Methodology

where no prior knowledge of the application's internal implementation or source code was available. The following methodology was used during the assessment:

- Manual navigation of the web application to understand functionality
- URL manipulation to test access control and authorization
- Form input testing to observe input handling and validation
- Error message analysis to identify information disclosure
- Review of HTML page source to identify exposed information

All testing was performed using a standard web browser without the use of automated tools. The assessment focused on identifying observable security weaknesses without performing any destructive or intrusive actions.

4. Findings Summary

During the manual security assessment of the **Hack Yourself First** web application, a total of **16 security issues** were identified. These findings include a combination of high, medium, and low severity vulnerabilities across different areas of the application.

The most critical issues were related to **broken access control** and **authentication handling**, which could allow unauthorized access to application data and enable user enumeration. Several medium-severity issues were identified in input validation, error handling, and missing security controls. Low-severity issues were primarily related to security configuration and information disclosure.

Severity	Number of Issues
High	2
Medium	8
Low	6
Total	**16**

Table 4.1 Summary of Findings by Severity

This summary provides an overview of the identified security weaknesses, while detailed descriptions and evidence for each issue are provided in the subsequent chapter.

5. Detailed Findings

5.1 High Severity Findings

Finding 1: Insecure Direct Object Reference (IDOR)

Severity: High

Location: /Supercar/{id}, /Make/{id}

Description:

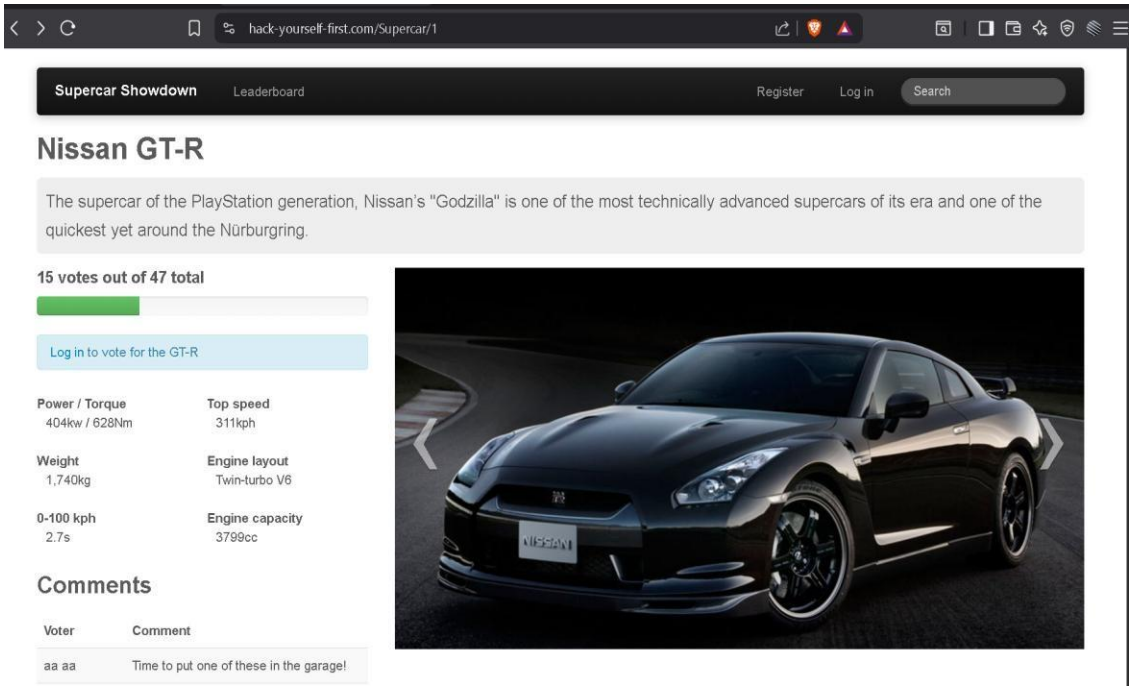
The application exposes internal object identifiers directly in the URL. By modifying the numeric ID value, different records can be accessed without proper authorization.

Observation:

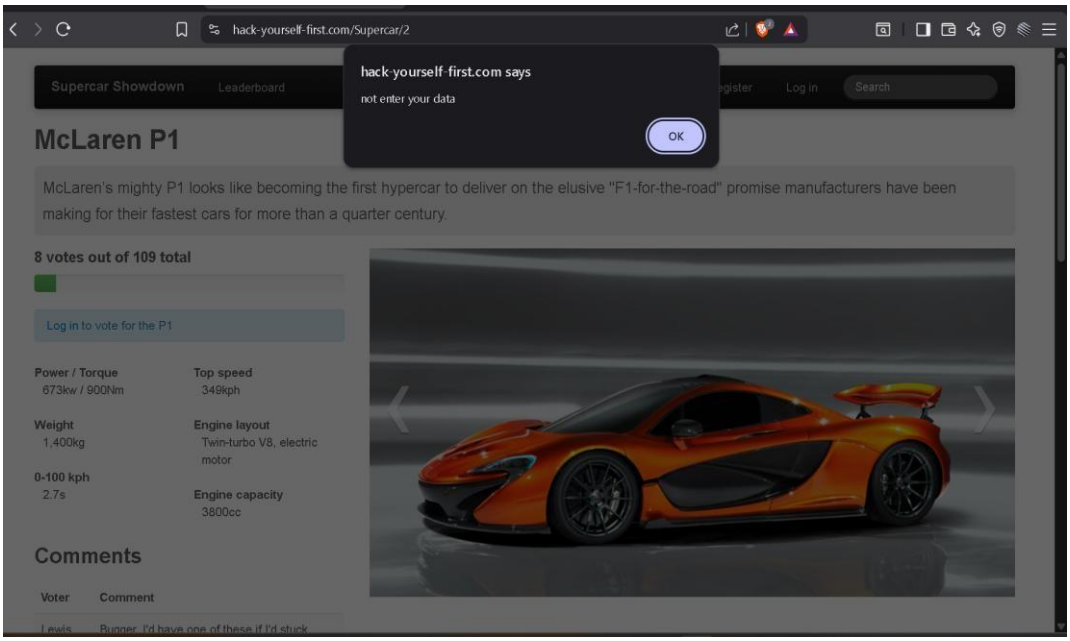
Changing the URL from /Supercar/1 to /Supercar/2 loads another supercar page successfully. The application displays only a client-side alert message but does not block access on the server side.

Impact:

Unauthorized users can access and enumerate application data, leading to confidentiality breaches.



Screenshot 5.1 Original access to /Supercar/1 before URL manipulation.



Screenshot 5.2 IDOR vulnerability demonstrated by changing the URL from

Finding 2: User Enumeration via Registration Error Message

Severity: High

Location: /Account/Register

Description:

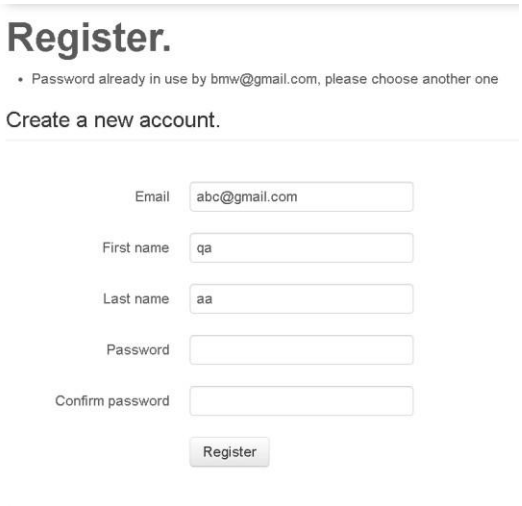
The registration functionality reveals sensitive information through detailed error messages.

Observation:

When registering with a password already used, the application displays an error message disclosing that the password is already in use by a specific email address.

Impact:

Attackers can confirm the existence of registered users, enabling targeted phishing, credential stuffing, and social engineering attacks.



Screenshot 5.3: Registration error message disclosing that the password is already in use by an existing email address.

5.2 Medium Severity Findings

Finding 3: Improper Input Handling in Search Functionality

Severity: Medium

Location: /Search

Description:

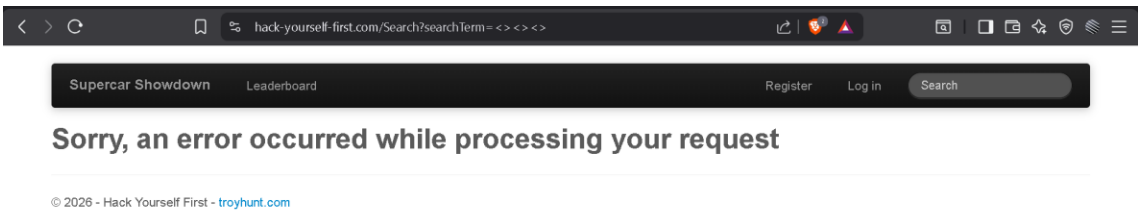
The application does not consistently validate user input in the search functionality.

Observation:

Submitting malformed input results in inconsistent behavior, including generic error pages or reflected input.

Impact:

Inconsistent input handling increases the risk of injection vulnerabilities and application instability.



Screenshot 5.4 Registration error message disclosing that the password is already in use by an existing email address.

Finding 4: Reflected User Input in Search Results

Severity: Medium

Location: /Search

Description:

User-provided input is reflected back in the response without consistent output encoding.

Observation:

Search terms entered by the user are displayed directly in the results page.

Impact:

If output encoding fails in future changes, this could lead to reflected cross-site scripting (XSS).

Finding 5: Missing Authentication on Internal Pages

Severity: Medium

Description:

The application allows unauthenticated users to access internal application pages.

Observation:

Pages such as supercar details, manufacturers, and leaderboards are accessible without logging in.

Impact:

Sensitive application data is exposed to unauthorized users.

Finding 6: Resource Enumeration via Predictable URLs

Severity: Medium

Description:

Application resources are identified using predictable sequential numeric IDs.

Observation:

Resources can be accessed sequentially by incrementing ID values in the URL.

Impact:

Attackers can enumerate and harvest large amounts of application data.

Finding 7: Absence of CAPTCHA or Rate Limiting

Severity: Medium

Location: Login and Registration pages

Description:

The application does not implement protections against automated requests.

Observation:

Login and registration endpoints accept unlimited requests without CAPTCHA or throttling.

Impact:

Enables brute-force attacks and automated abuse.

Finding 8: Missing CSRF Protection

Severity: Medium

Description:

The application does not visibly implement protection against cross-site request forgery attacks.

Observation:

Forms do not include anti-CSRF tokens or similar protection mechanisms.

Impact:

Users could be tricked into performing unintended actions.

Finding 9: Missing HTTP Security Headers

Severity: Medium

Description:

Important HTTP security headers are not configured.

Observation:

Headers such as Content-Security-Policy, X-Frame-Options, and Strict-Transport-Security are absent.

Impact:

Increases exposure to clickjacking, XSS, and protocol downgrade attacks.

Finding 10: Inconsistent Error Handling

Severity: Medium

Description:

The application does not handle errors consistently.

Observation:

Unexpected input sometimes results in a generic error page instead of graceful handling.

Impact:

May lead to application instability and denial-of-service scenarios.

5.3 LOW SEVERITY FINDINGS

Finding 11: Sensitive Input Passed via GET Method

Severity: Low

Description:

User input is transmitted via URL parameters.

Observation:

Search input is sent using the GET method.

Impact:

Data may be logged, cached, or exposed unintentionally.

Finding 12: Application Logic Controlled by URL Parameters

Severity: Low

Description:

Application logic relies on user-controlled URL parameters.

Observation:

Sorting behavior is controlled via GET parameters.

Impact:

May allow parameter tampering.

Finding 13: No Input Length Validation

Severity: Low

Description:

Input fields do not enforce maximum length restrictions.

Observation:

Search and registration inputs accept long strings without restriction.

Impact:

Could impact performance or cause denial-of-service conditions.

Finding 14: Technology Stack Disclosure

Severity: Low

Description:

Frontend technologies are exposed through page source.

Observation:

Libraries such as jQuery and Bootstrap are visible.

Impact:

Helps attackers fingerprint the application.

Finding 15: Reliance on Third-Party Scripts

Severity: Low

Description:

The application loads multiple third-party scripts.

Observation:

External analytics and tracking scripts are included.

Impact:

Introduces supply-chain security risks.

Finding 16: Weak Password Policy Enforcement

Severity: Low

Description:

The application does not visibly enforce strong password policies.

Observation:

No password complexity or strength requirements are shown during registration.

Impact:

Encourages weak passwords and increases risk of account compromise.

6. Conclusion

This security assessment identified **16 security issues** in the *Hack Yourself First* web application through manual testing of publicly accessible functionality. The findings include **high, medium, and low severity issues**, primarily related to broken access control, authentication weaknesses, improper input handling, and missing security controls.

The most critical issues, such as **Insecure Direct Object Reference (IDOR)** and **user enumeration through registration error messages**, could lead to unauthorized data access and disclosure of sensitive user information. Several medium and low severity issues further increase the application's attack surface.

Although the application is intentionally insecure for educational purposes, the identified issues represent **real-world security risks** commonly found in production web applications.